

*Peripheral Design  
Handbook*

intel

intel



*Peripheral Design Handbook*

*August 1981*



# PERIPHERAL DESIGN HANDBOOK

**AUGUST 1981**

Intel Corporation makes no warranty for the use of its products and assumes no responsibility for any errors which may appear in this document nor does it make a commitment to update the information contained herein.

Intel software products are copyrighted by and shall remain the property of Intel Corporation. Use, duplication or disclosure is subject to restrictions stated in Intel's software license, or as defined in ASPR 7-104.9 (a) (9). Intel Corporation assumes no responsibility for the use of any circuitry other than circuitry embodied in an Intel product. No other circuit patent licenses are implied.

No part of this document may be copied or reproduced in any form or by any means without the prior written consent of Intel Corporation.

The following are trademarks of Intel Corporation and may only be used to identify Intel products:

BXP	Intelevison	MULTIBUS
CREDIT	Intellec	MULTIMODULE
i	iSBC	Plug-A-Bubble
ICE	iSBX	PROMPT
ICS	Library Manager	Promware
i <sub>m</sub>	MCS	RMX
Insite	Megachassis	UPI
Intel	Micromainframe	μScope
	Micromap	System 2000

MDS is an ordering code only and is not used as a product name or trademark. MDS® is a registered trademark of Mohawk Data Sciences Corporation.

Additional copies of this manual or other Intel literature may be obtained from:

Intel Corporation  
Literature Department SV3-3  
3065 Bowers Avenue  
Santa Clara, CA 95051

# Table of Contents

## CHAPTER 1

### Data Sheets

#### Slave Processors

8041AH/8041AH-2/8641A/8741A Universal Peripheral Interface 8-Bit Microcomputer . . . . .	1-1
8042/8742 Universal Peripheral Interface 8-Bit Microcomputer . . . . .	1-14
8231A Arithmetic Processing Unit . . . . .	1-27
8232 Floating Point Processing Unit . . . . .	1-37
8294 Data Encryption Unit . . . . .	1-49
8295 Dot Matrix Printer Controller . . . . .	1-60

#### Memory Controllers

8202A Dynamic RAM Controller . . . . .	1-69
8203 64K Dynamic RAM Controller . . . . .	1-83
8206 Error Detection and Correction Unit . . . . .	1-98
8271/8271-6 Programmable Floppy Disk Controller . . . . .	1-117
8272 Single/Double Density Floppy Disk Controller . . . . .	1-146

#### Data Communications

8251A Programmable Communication Interface . . . . .	1-165
8256 Multifunction Universal Asynchronous Receiver-Transmitter (MUART) . . . . .	1-182
8273, 8273-4, 8273-8 Programmable HDLC/SDLC Protocol Controller . . . . .	1-191
8274 Multi-Protocol Serial Controller . . . . .	1-216
8291A GPIB Talker/Listener . . . . .	1-251
8292 GPIB Controller . . . . .	1-280
8293 GPIB Transceiver . . . . .	1-295

#### Controllers

8253/8253-5 Programmable Interval Timer . . . . .	1-307
8254 Programmable Interval Timer . . . . .	1-318
8255A/8255A-5 Programmable Peripheral Interface . . . . .	1-333
8275 Programmable CRT Controller . . . . .	1-354
8276 Small System CRT Controller . . . . .	1-378
8279/8279-5 Programmable Keyboard/Display Interface . . . . .	1-395

## CHAPTER 2

### Application Notes

#### Slave Processors

Introduction to the UPI-41A™ . . . . .	2-1
An 8741A/8041A Digital Cassette Controller . . . . .	2-45
Using the 8295 Dot Matrix Printer Controller . . . . .	2-77

#### Memory Controllers

5-Volt Only Dynamic RAM Interface for 8086 Systems . . . . .	2-110
An Intelligent Data Base System Using the 8272 . . . . .	2-132
Software Design and Implementation of Floppy Disk Subsystems . . . . .	2-172

#### Data Communications

Using the 8251 Universal Synchronous/Asynchronous Receiver/Transmitter . . . . .	2-241
Using the 8273 SDLC/HDLC Protocol Controller . . . . .	2-271
Asynchronous Communication with the 8274 Multiple Protocol Serial Controller . . . . .	2-317
Using the 8292 GPIB Controller . . . . .	2-356

#### Controllers

8255A Programmable Peripheral Interface Applications . . . . .	2-409
A Low Cost CRT Terminal Using the 8275 . . . . .	2-439

## APPENDIX

Intel Peripheral Components . . . . .	A-1
---------------------------------------	-----





# ***Slave Processors***

---

# 8041AH/8041AH-2/8641A/8741A UNIVERSAL PERIPHERAL INTERFACE 8-BIT MICROCOMPUTER

- 8041AH-2: 12 MHz  
8041AH: 8 MHz
- 8-Bit CPU plus ROM, RAM, I/O, Timer and Clock in a Single Package
- One 8-Bit Status and Two Data Registers for Asynchronous Slave-to-Master Interface
- DMA, Interrupt, or Polled Operation Supported
- 1024 x 8 ROM/EPROM, 64 x 8 RAM, 8-Bit Timer/Counter, 18 Programmable I/O Pins
- Fully Compatible with MCS-48™, MCS-80™, MCS-85™, and iAPX-86,88 Microprocessor Families
- Interchangeable ROM and EPROM Versions
- Expandable I/O
- RAM Power-Down Capability
- Over 90 Instructions: 70% Single Byte
- Single 5V Supply

The Intel® 8041AH/8741A is a general-purpose, programmable interface device designed for use with a variety of 8-bit microprocessor systems. It contains a low cost microcomputer with program memory, data memory, 8-bit CPU, I/O ports, timer/counter, and clock in a single 40-pin package. Interface registers are included to enable the UPI device to function as a peripheral controller in MCS-48™, MCS-80™, iAPX-85™, iAPX-86, iAPX-88, and other 8- or 16-bit systems.

The UPI-41A™ has 1K words of program memory and 64 words of data memory on-chip. To allow full user flexibility the program memory is available as ROM in the 8041AH version or as UV-erasable EPROM in the 8741A version. The 8741A and the 8041AH are fully pin compatible for easy transition from prototype to production level designs. The 8741A is a one-time programmable (at the factory) 8741A which can be ordered as the first 25 pieces of a new 8041AH order. The substitution of 8641As for 8041AHs allows for very fast turnaround for initial code verification and evaluation results.

The device has two 8-bit, TTL-compatible I/O ports and two test inputs. Individual port lines can function as either inputs or outputs under software control. I/O can be expanded with the 8243 device which is directly compatible and has 16 I/O lines. An 8-bit programmable timer/counter is included in the UPI device for generating timing sequences or counting external inputs. Additional UPI features include: single 5V supply, low power standby mode (in the 8041AH), single-step mode for debug and dual working register banks.

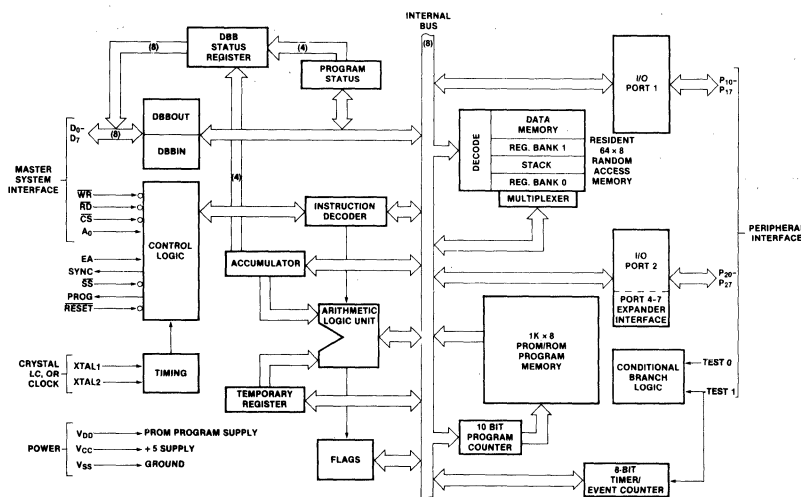


Figure 1. Block Diagram

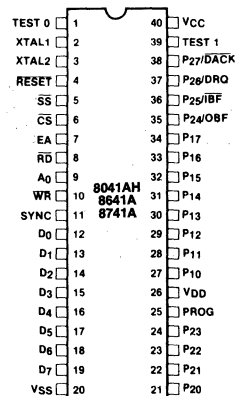


Figure 2. Pin Configuration

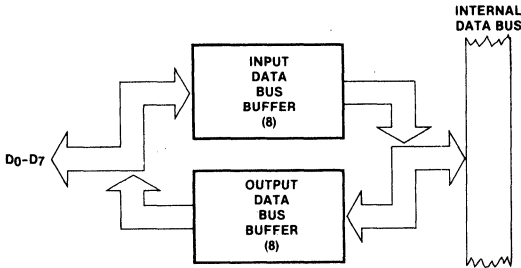


Table 1. Pin Description

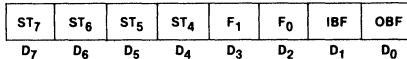
Symbol	Pin No.	Type	Name and Function
TEST 0, TEST 1	1 39	I	<b>Test Inputs:</b> Input pins which can be directly tested using conditional branch instructions.  <b>Frequency Reference:</b> TEST 1 ( $T_1$ ) also functions as the event timer input (under software control). TEST 0 ( $T_0$ ) is used during PROM programming and verification in the 8741A.
XTAL 1, XTAL 2	2 3	I	<b>Inputs:</b> Inputs for a crystal, LC or an external timing signal to determine the internal oscillator frequency.
$\overline{\text{RESET}}$	4	I	<b>Reset:</b> Input used to reset status flip-flops and to set the program counter to zero.  $\overline{\text{RESET}}$ is also used during PROM programming and verification.
$\overline{\text{SS}}$	5	I	<b>Single Step:</b> Single step input used in the 8741A in conjunction with the SYNC output to step the program through each instruction.
$\overline{\text{CS}}$	6	I	<b>Chip Select:</b> Chip select input used to select one UPI-41A microcomputer out of several connected to a common data bus.
EA	7	I	<b>External Access:</b> External access input which allows emulation, testing and PROM/ROM verification. This pin should be tied low if unused.
$\overline{\text{RD}}$	8	I	<b>Read:</b> I/O read input which enables the master CPU to read data and status words from the OUTPUT DATA BUS BUFFER or status register.
$A_0$	9	I	<b>Command/Data Select:</b> Address input used by the master processor to indicate whether byte transfer is data ( $A_0 = 0$ , $F_1$ is reset) or command ( $A_0 = 1$ , $F_1$ is set).
$\overline{\text{WR}}$	10	I	<b>Write:</b> I/O write input which enables the master CPU to write data and command words to the UPI-41A INPUT DATA BUS BUFFER.
SYNC	11	O	<b>Output Clock:</b> Output signal which occurs once per UPI-41A instruction cycle. SYNC can be used as a strobe for external circuitry; it is also used to synchronize single step operation.
$D_0$ - $D_7$ (BUS)	12-19	I/O	<b>Data Bus:</b> Three-state, bidirectional DATA BUS BUFFER lines used to interface the UPI-41A microcomputer to an 8-bit master system data bus.
$P_{10}$ - $P_{17}$	27-34	I/O	<b>Port 1:</b> 8-bit, PORT 1 quasi-bidirectional I/O lines.
$P_{20}$ - $P_{27}$	21-24 35-38	I/O	<b>Port 2:</b> 8-bit, PORT 2 quasi-bidirectional I/O lines. The lower 4 bits ( $P_{20}$ - $P_{23}$ ) interface directly to the 8243 I/O expander device and contain address and data information during PORT 4-7 access. The upper 4 bits ( $P_{24}$ - $P_{27}$ ) can be programmed to provide interrupt Request and DMA Handshake capability. Software control can configure $P_{24}$ as Output Buffer Full (OBF) interrupt, $P_{25}$ as Input Buffer Full (IBF) interrupt, $P_{26}$ as DMA Request (DRQ), and $P_{27}$ as DMA ACKnowledge (DACK).
PROG	25	I/O	<b>Program:</b> Multifunction pin used as the program pulse input during PROM programming.  During I/O expander access the PROG pin acts as an address/data strobe to the 8243. This pin should be tied high if unused.
$V_{CC}$	40		<b>Power:</b> +5V main power supply pin.
$V_{DD}$	26		<b>Power:</b> +5V during normal operation. +25V during programming operation. Low power standby pin in ROM version.
$V_{SS}$	20		<b>Ground:</b> Circuit ground potential.

## UPI-41A™ FEATURES AND ENHANCEMENTS

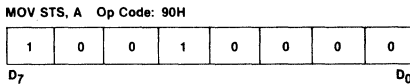
- Two Data Bus Buffers, one for input and one for output. This allows a much cleaner Master/Slave protocol.



- 8 Bits of Status



ST<sub>4</sub>-ST<sub>7</sub> are user definable status bits. These bits are defined by the "MOV STS, A" single byte, single cycle instruction. Bits 4-7 of the accumulator are moved to bits 4-7 of the status register. Bits 0-3 of the status register are not affected.



- $\overline{RD}$  and  $\overline{WR}$  are edge triggered. IBF, OBF, F<sub>1</sub> and INT change internally after the trailing edge of  $\overline{RD}$  or  $\overline{WR}$ .



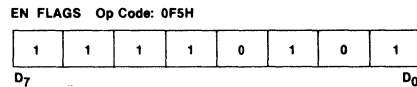
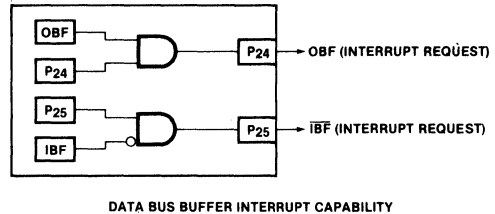
During the time that the host CPU is reading the status register, the 8041AH is prevented from updating this register or is 'locked out.'

- P<sub>24</sub> and P<sub>25</sub> are port pins or Buffer Flag pins which can be used to interrupt a master processor. These pins default to port pins on Reset.

If the "EN FLAGS" instruction has been executed, P<sub>24</sub> becomes the OBF (Output Buffer Full) pin. A "1" written to P<sub>24</sub> enables the OBF pin (the pin outputs the OBF Status Bit). A "0" written to P<sub>24</sub> disables the OBF pin (the pin remains low). This pin can be used to indicate that valid data is available from the UPI-41A (in Output Data Bus Buffer).

If "EN FLAGS" has been executed, P<sub>25</sub> becomes the IBF (Input Buffer Full) pin. A "1" written to P<sub>25</sub> enables the IBF pin (the pin outputs the inverse of the IBF Status Bit). A "0" written to P<sub>25</sub> disables the IBF

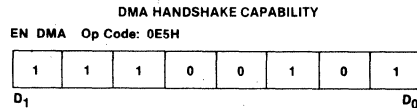
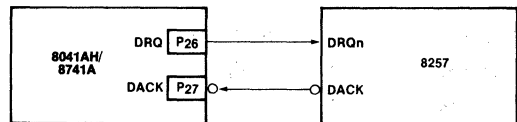
pin (the pin remains low). This pin can be used to indicate that the UPI-41A is ready for data.



- P<sub>26</sub> and P<sub>27</sub> are port pins or DMA handshake pins for use with a DMA controller. These pins default to port pins on Reset.

If the "EN DMA" instruction has been executed, P<sub>26</sub> becomes the DRQ (DMA ReQuest) pin. A "1" written to P<sub>26</sub> causes a DMA request (DRQ is activated). DRQ is deactivated by DACK · RD, DACK · WR, or execution of the "EN DMA" instruction.

If "EN DMA" has been executed, P<sub>27</sub> becomes the DACK (DMA ACKnowledge) pin. This pin acts as a chip select input for the Data Bus Buffer registers during DMA transfers.



### 8041AH ENHANCEMENTS OVER 8041A

- The RESET input on the 8041AH was changed to include a 2 stage synchronizer to support reliable reset operation for 12 MHz operation.
- As noted in the status register description, during the time that the host CPU is reading the status register, the 8041AH is prevented from updating or is 'locked out.'
- When EA is enabled on the 8041A, the program counter is placed on Port 1 and the lower two bits of Port 2. On the 8041AH, this information is multiplexed with PORT DATA (see port timing diagrams at end of this data sheet).
- The 8041AH additionally supports single step mode as described in the pin description section.

APPLICATIONS

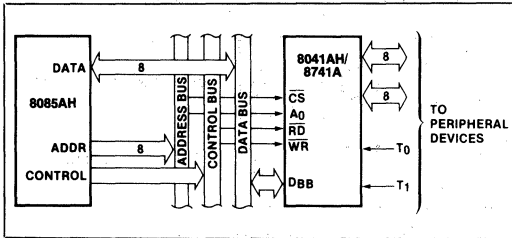


Figure 3. 8085AH-8041AH Interface

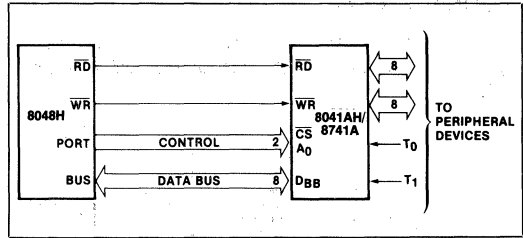


Figure 4. 8048AH-8041AH Interface

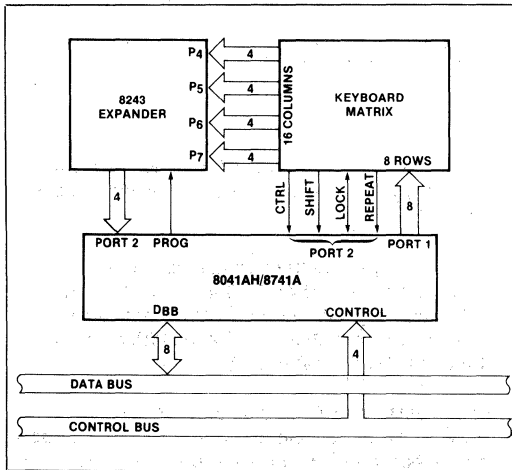


Figure 5. 8041AH-8243 Keyboard Scanner

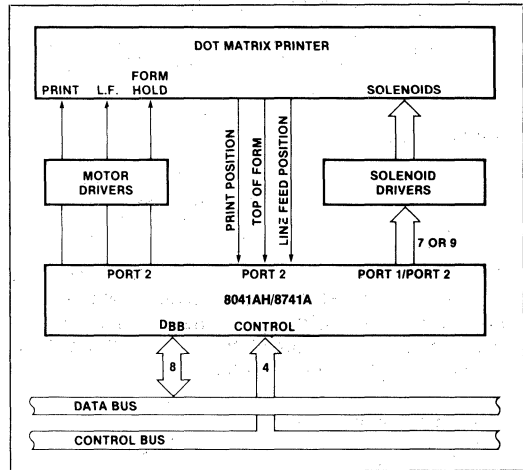


Figure 6. 8041AH Matrix Printer Interface

PROGRAMMING, VERIFYING, AND ERASING THE 8741A EPROM

Programming Verification

In brief, the programming process consists of: activating the program mode, applying an address, latching the address, applying data, and applying a programming pulse. Each word is programmed completely before moving on to the next and is followed by a verification step. The following is a list of the pins used for programming and a description of their functions:

Pin	Function
XTAL 1	Clock Input (1 to 6MHz)
Reset	Initialization and Address Latching
Test 0	Selection of Program or Verify Mode
EA	Activation of Program/Verify Modes
BUS	Address and Data Input Data Output During Verify
P20-1	Address Input
V <sub>DD</sub>	Programming Power Supply
PROG	Program Pulse Input

WARNING:

An attempt to program a missocketed 8741A will result in severe damage to the part. An indication of a properly socketed part is the appearance of the SYNC clock output. The lack of this clock may be used to disable the programmer.

The Program/Verify sequence is:

1. A<sub>0</sub> = 0V, CS = 5V, EA = 5V, RESET = 0V, TEST0 = 5V, V<sub>DD</sub> = 5V, clock applied or internal oscillator operating, BUS and PROG floating.
2. Insert 8741A in programming socket
3. TEST 0 = 0v (select program mode)
4. EA = 23V (activate program model)<sup>1</sup>
5. Address applied to BUS and P20-1
6. RESET = 5v (latch address)
7. Data applied to BUS<sup>2</sup>
8. V<sub>DD</sub> = 25v (programming power)<sup>2</sup>
9. PROG = 0v followed by one 50ms pulse to 23V<sup>2</sup>
10. V<sub>DD</sub> = 5v
11. TEST 0 = 5v (verify mode)

12. Read and verify data on BUS
13. TEST 0 = 0v
14. RESET = 0v and repeat from step 5
15. Programmer should be at conditions of step 1 when 8741A is removed from socket.

**NOTE:**

1. When verifying ROM, EA = 12V.
2. Not used in verify ROM procedure.

**8741A Erasure Characteristics**

The erasure characteristics of the 8741A are such that erasure begins to occur when exposed to light with wavelengths shorter than approximately 4000 Angstroms (Å). It should be noted that sunlight and certain types of fluorescent lamps have wavelengths in the 3000-4000Å range. Data show that constant exposure to room level fluorescent lighting could erase the typical

8741A in approximately 3 years while it would take approximately one week to cause erasure when exposed to direct sunlight. If the 8741A is to be exposed to these types of lighting conditions for extended periods of time, opaque labels are available from Intel which should be placed over the 8741A window to prevent unintentional erasure.

The recommended erasure procedure for the 8741A is exposure to shortwave ultraviolet light which has a wavelength of 2537Å. The integrated dose (i.e., UV intensity x exposure time) for erasure should be a minimum of 15 w-sec/cm<sup>2</sup>. The erasure time with this dosage is approximately 15 to 20 minutes using an ultraviolet lamp with a 12,000 μW/cm<sup>2</sup> power rating. The 8741A should be placed within one inch of the lamp tubes during erasure. Some lamps have a filter on their tubes which should be removed before erasure.

**ABSOLUTE MAXIMUM RATINGS\***

Ambient Temperature Under Bias ..... 0°C to 70°C  
 Storage Temperature ..... - 65°C to + 150°C  
 Voltage on Any Pin With Respect  
     to Ground ..... -0.5V to +7V  
 Power Dissipation ..... 1.5 Watt

*\*NOTICE: Stresses above those listed under "Absolute Maximum Ratings" may cause permanent damage to the device. This is a stress rating only and functional operation of the device at these or any other conditions above those indicated in the operational sections of this specification is not implied. Exposure to absolute maximum rating conditions for extended periods may affect device reliability.*

**D.C. CHARACTERISTICS** ( $T_A = 0^\circ$  to  $+70^\circ\text{C}$ ,  $V_{CC} = V_{DD} = +5\text{V} \pm 10\%$ )

Symbol	Parameter	8041AH/ 8041AH-2		8641A/8741A		Units	Test Conditions
		Min.	Max.	Min.	Max.		
$V_{IL}$	Input Low Voltage (Except XTAL1, XTAL2, $\overline{\text{RESET}}$ )	-0.5	0.8	-0.5	0.8	V	
$V_{IL1}$	Input Low Voltage (8XTAL1, XTAL2, $\overline{\text{RESET}}$ )	-0.5	0.6	-0.5	0.6	V	
$V_{IH}$	Input High Voltage (Except XTAL1, XTAL2, $\overline{\text{RESET}}$ )	2.2	$V_{CC}$	2.2	$V_{CC}$		
$V_{IH1}$	Input High Voltage (XTAL1, XTAL2, $\overline{\text{RESET}}$ )	3.8	$V_{CC}$	3.8	$V_{CC}$	V	
$V_{OL}$	Output Low Voltage ( $D_0$ - $D_7$ )		0.45		0.45	V	$I_{OL} = 2.0 \text{ mA}$
$V_{OL1}$	Output Low Voltage ( $P_{10}$ $P_{17}$ , $P_{20}$ $P_{27}$ , Sync)		0.45		0.45	V	$I_{OL} = 1.6 \text{ mA}$
$V_{OL2}$	Output Low Voltage (Prog)		0.45		0.45	V	$I_{OL} = 1.0 \text{ mA}$
$V_{OH}$	Output High Voltage ( $D_0$ - $D_7$ )	2.4		2.4		V	$I_{OH} = -400 \mu\text{A}$
$V_{OH1}$	Output High Voltage (All Other Outputs)	2.4		2.4		V	$I_{OH} = -50 \mu\text{A}$
$I_{IL}$	Input Leakage Current ( $T_0$ , $T_1$ , $\overline{\text{RD}}$ , $\overline{\text{WR}}$ , $\overline{\text{CS}}$ , $A_0$ , EA)		$\pm 10$		$\pm 10$	$\mu\text{A}$	$V_{SS} \leq V_{IN} \leq V_{CC}$
$I_{OZ}$	Output Leakage Current ( $D_0$ - $D_7$ , High Z State)		$\pm 10$		$\pm 10$	$\mu\text{A}$	$V_{SS} + 0.45 \leq V_{OUT} \leq V_{CC}$
$I_{LI}$	Low Input Load Current ( $P_{10}$ $P_{17}$ , $P_{20}$ $P_{27}$ )		0.3		0.3	mA	$V_{IL} = 0.8 \text{ V}$
$I_{LI1}$	Low Input Load Current ( $\overline{\text{RESET}}$ , $\overline{\text{SS}}$ )		0.2		0.2	mA	$V_{IL} = 0.8 \text{ V}$
$I_{DD}$	$V_{DD}$ Supply Current		15		15	mA	Typical = 5 mA
$I_{CC} + I_{DD}$	Total Supply Current		125		125	mA	Typical = 60 mA
$I_{IH}$	Input Leakage Current		100		100	NA	$V_{IN} = V_{CC}$
$C_{IN}$	Input Capacitance		10		10	pF	
$C_{I/O}$	I/O Capacitance		20		20	pF	

**D.C. CHARACTERISTICS—PROGRAMMING** ( $T_A = 25^\circ\text{C} \pm 5^\circ\text{C}$ ,  $V_{CC} = 5\text{V} \pm 5\%$ ,  $V_{DD} = 25\text{V} \pm 1\text{V}$ )

Symbol	Parameter	Min.	Max.	Unit	Test Conditions
$V_{DOH}$	$V_{DD}$ Program Voltage High Level	24.0	26.0	V	
$V_{DDL}$	$V_{DD}$ Voltage Low Level	4.75	5.25	V	
$V_{PH}$	PROG Program Voltage High Level	21.5	24.5	V	
$V_{PL}$	PROG Voltage Low Level		0.2	V	
$V_{EAH}$	EA Program or Verify Voltage High Level	21.5	24.5	V	
$V_{EAL}$	EA Voltage Low Level		5.25	V	
$I_{DD}$	$V_{DD}$ High Voltage Supply Current		30.0	mA	
$I_{PROG}$	PROG High Voltage Supply Current		16.0	mA	
$I_{EA}$	EA High Voltage Supply Current		1.0	mA	

**A.C. CHARACTERISTICS** ( $T_{CC} = 0^\circ\text{C}$  to  $+70^\circ\text{C}$ ,  $V_{SS} = 0\text{V}$ ,  $V_{CC} = V_{DD} = +5\text{V} \pm 10\%$ )

**DBB READ**

Symbol	Parameter	8041AH		8041AH-2		8641A/8741A		Units
		Min.	Max.	Min.	Max.	Min.	Max.	
$t_{AR}$	$\overline{CS}$ , $A_0$ Setup to $\overline{RD}\downarrow$	0		0		0		ns
$t_{RA}$	$\overline{CS}$ , $A_0$ Hold After $\overline{RD}\uparrow$	0		0		0		ns
$t_{RR}$	$\overline{RD}$ Pulse Width					250		ns
$t_{AD}$	$\overline{CS}$ , $A_0$ to Data Out Delay		130		130		225	ns <sup>[1]</sup>
$t_{RD}$	$\overline{RD}\downarrow$ to Data Out Delay		130		130		225	ns <sup>[1]</sup>
$t_{DF}$	$\overline{RD}\uparrow$ to Data Float Delay		85		85		100	ns
$t_{CY}$	Cycle Time (Except 8741A-8)	2	15	1.25	15	2.5	15	$\mu\text{s}$ <sup>[2]</sup>
$t_{CY}$	Cycle Time (8741A-8)					4.17	15	$\mu\text{s}$ <sup>[3]</sup>

**DBB WRITE**

Symbol	Parameter	Min.	Max.	Min.	Max.	Min.	Max.	Units
$t_{AW}$	$\overline{CS}$ , $A_0$ Setup to $\overline{WR}\downarrow$	0		0		0		ns
$t_{WA}$	$\overline{CS}$ , $A_0$ Hold After $\overline{WR}\uparrow$	0		0		0		ns
$t_{WW}$	$\overline{WR}$ Pulse Width	160		160		250		ns
$t_{DW}$	Data Setup to $\overline{WR}\uparrow$	130		130		150		ns
$t_{WD}$	Data Hold After $\overline{WR}\uparrow$	0		0		0		ns

**NOTES:**

- $C_L = 150$  pF.
- 8, 12, 6 MHz XTAL respectively.
- 3.6 MHz XTAL.

**A.C. CHARACTERISTICS—PROGRAMMING** ( $T_A = 25^\circ\text{C} \pm 5^\circ\text{C}$ ,  $V_{CC} = 5\text{V} \pm 5\%$ ,  $V_{DD} = 25\text{V} \pm 1\text{V}$ )

Symbol	Parameter	Min.	Max.	Unit	Test Conditions
$t_{AW}$	Address Setup Time to $\overline{\text{RESET}}$ 1	4tcy			
$t_{WA}$	Address Hold Time After $\overline{\text{RESET}}$ 1	4tcy			
$t_{DW}$	Data in Setup Time to PROG 1	4tcy			
$t_{WD}$	Data in Hold Time After PROG 1	4tcy			
$t_{PH}$	$\overline{\text{RESET}}$ Hold Time to Verify	4tcy			
$t_{VDDW}$	$V_{DD}$ Setup Time to PROG 1	4tcy			
$t_{VDDH}$	$V_{DD}$ Hold Time After PROG 1	0			
$t_{PW}$	Program Pulse Width	50	60	mS	
$t_{TW}$	Test 0 Setup Time for Program Mode	4tcy			
$t_{WT}$	Test 0 Hold Time After Program Mode	4tcy			
$t_{DO}$	Test 0 to Data Out Delay		4tcy		
$t_{WW}$	$\overline{\text{RESET}}$ Pulse Width to Latch Address	4tcy			
$t_r, t_f$	$V_{DD}$ and PROG Rise and Fall Times	0.5	2.0	$\mu\text{S}$	
$t_{CY}$	CPU Operation Cycle Time	5.0		$\mu\text{S}$	
$t_{RE}$	$\overline{\text{RESET}}$ Setup Time Before EA 1.	4tcy			

Note: If TEST 0 is high,  $t_{DO}$  can be triggered by  $\overline{\text{RESET}}$  1.

**A.C. CHARACTERISTICS**
**DMA**

Symbol	Parameter	8041AH		8041AH-2		8641A/8741A		Units
		Min.	Max.	Min.	Max.	Min.	Max.	
$t_{ACC}$	$\overline{\text{DACK}}$ to $\overline{\text{WR}}$ or $\overline{\text{RD}}$	0		0		0		ns
$t_{CAC}$	$\overline{\text{RD}}$ or $\overline{\text{WR}}$ to $\overline{\text{DACK}}$	0		0		0		ns
$t_{ACD}$	$\overline{\text{DACK}}$ to Data Valid		130		130		225	ns <sup>[1]</sup>
$t_{CRQ}$	$\overline{\text{RD}}$ or $\overline{\text{WR}}$ to DRQ Cleared		90		90		200	ns

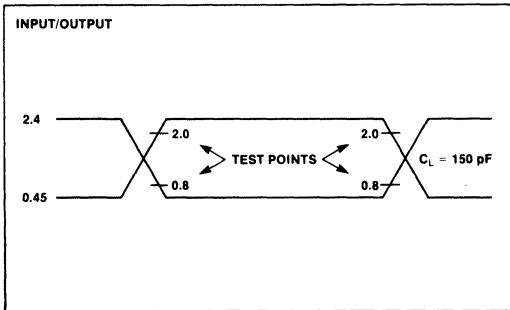
**A.C. CHARACTERISTICS**
**PORT 2**

( $T_A = 0^\circ\text{C}$  to  $+70^\circ\text{C}$ ,  $V_{CC} = +5\text{V} \pm 10\%$ )

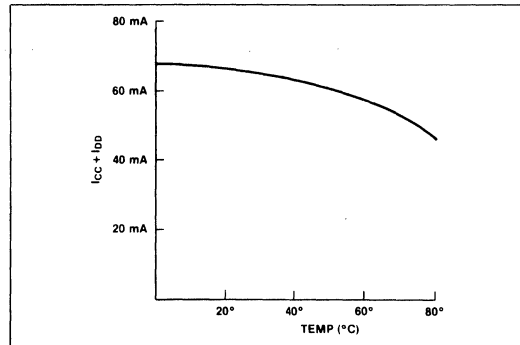
Symbol	Parameter	8041AH		8041AH-2		8641A/8741A		Units
		Min.	Max.	Min.	Max.	Min.	Max.	
$t_{CP}$	Port Control Setup Before Falling Edge of PROG	100		100		110		ns <sup>[1]</sup>
$t_{PC}$	Port Control Hold After Falling Edge of PROG			60		100		ns <sup>[2]</sup>
$t_{PR}$	PROG to Time P2 Input Must Be Valid						810	ns <sup>[1]</sup>
$t_{PF}$	Input Data Hold Time	0	150	0	150	0	150	ns <sup>[2]</sup>
$t_{DP}$	Output Data Setup time			200		250		ns <sup>[1]</sup>
$t_{PD}$	Output Data Hold Time					65		ns <sup>[2]</sup>
$t_{PP}$	PROG Pulse Width			700		1200		ns

NOTES: 1.  $C_L = 80\text{ pF}$ . 2.  $C_L = 20\text{ pF}$ .

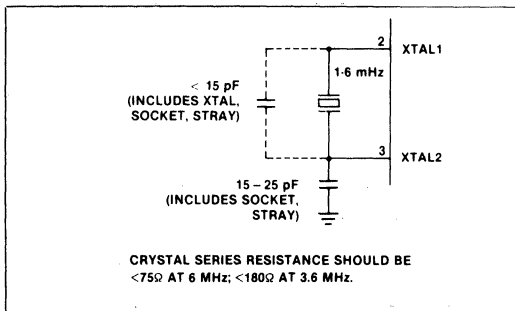
**A.C. TESTING INPUT, OUTPUT WAVEFORM**



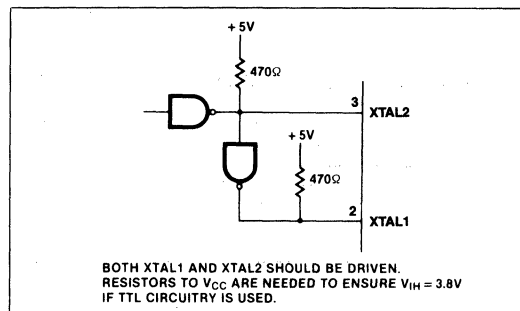
**TYPICAL 8041AH/8741A CURRENT**



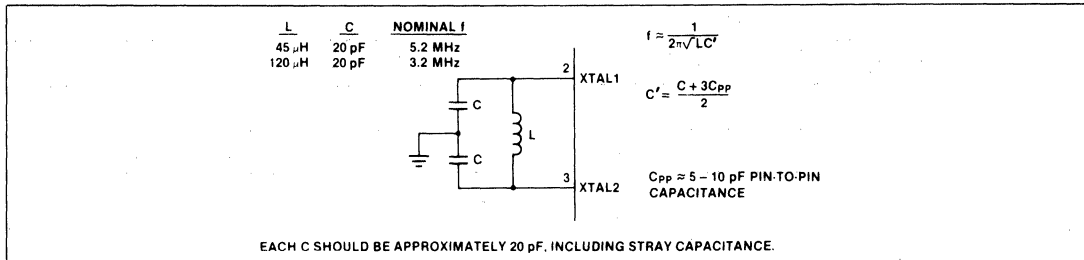
**CRYSTAL OSCILLATOR MODE**



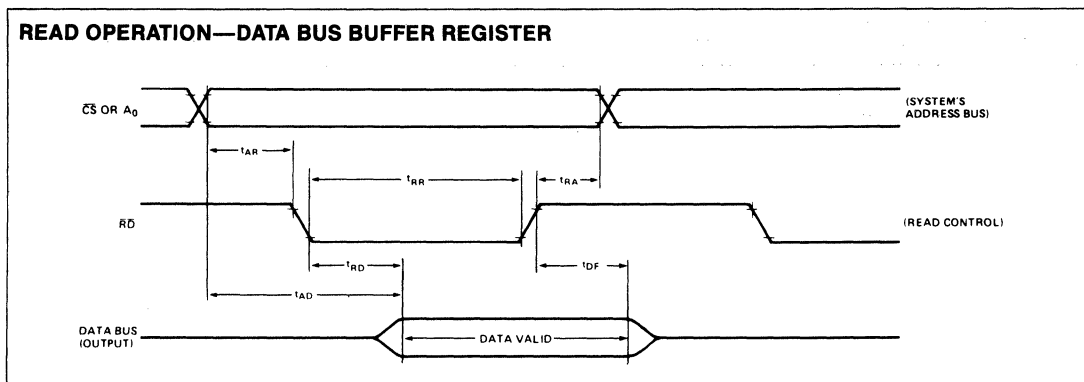
**DRIVING FROM EXTERNAL SOURCE**



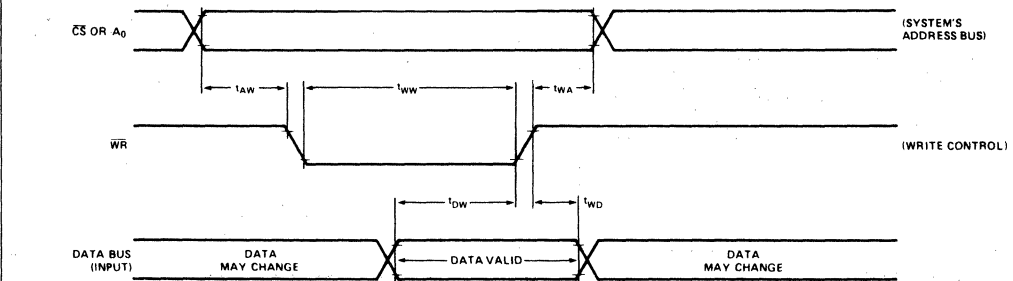
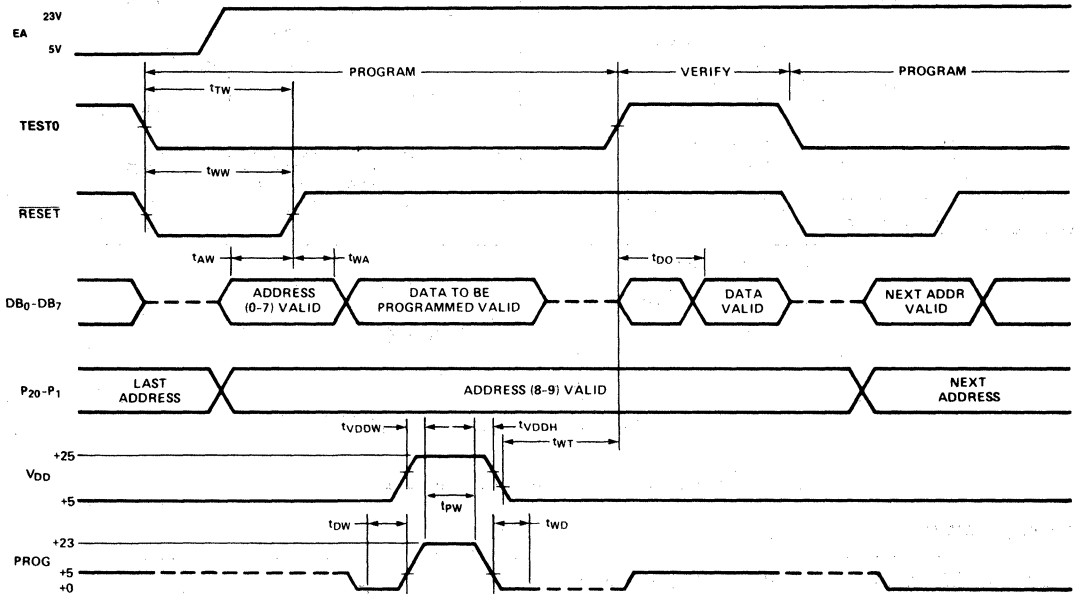
**LC OSCILLATOR MODE**



**WAVEFORMS**

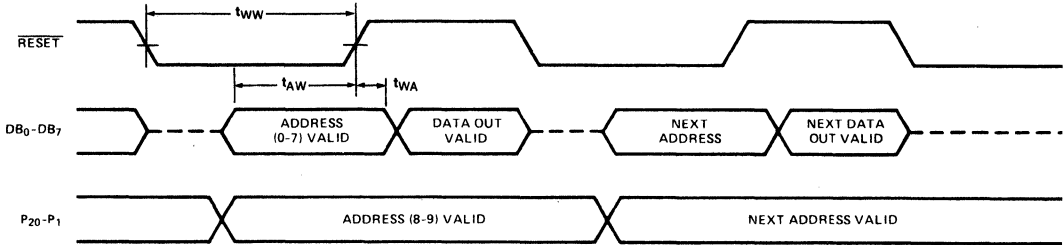




**WAVEFORMS (Continued)**
**WRITE OPERATION—DATA BUS BUFFER REGISTER**

**COMBINATION PROGRAM/VERIFY MODE (EPROM'S ONLY)**


WAVEFORMS (Continued)

VERIFY MODE (ROM/EPROM)



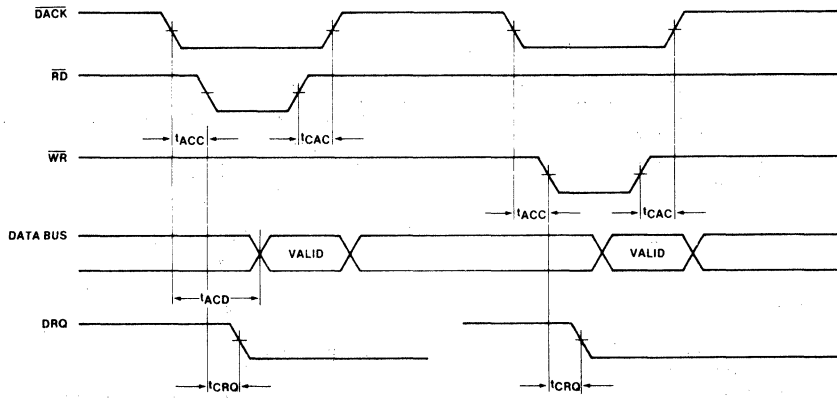
NOTES:

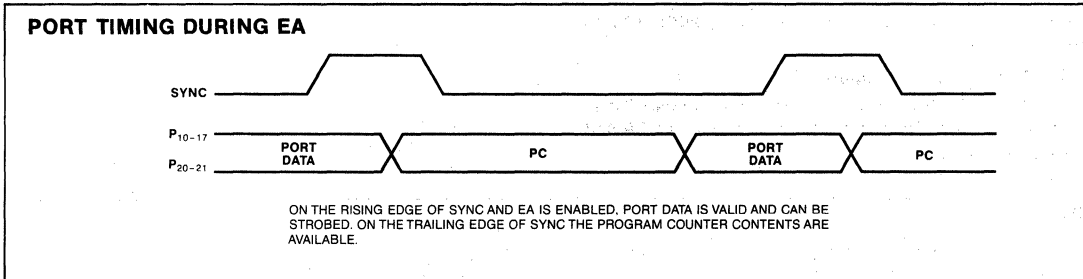
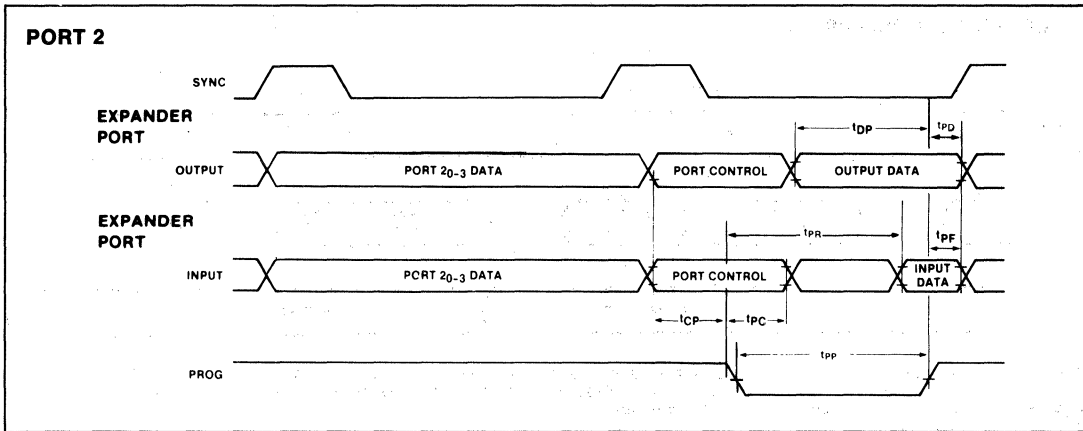
1. PROG MUST FLOAT IF EA IS LOW (i.e., = 23V), OR IF TO = 5V FOR THE 8741A. FOR THE 8041AH PROG MUST ALWAYS FLOAT.
2. XTAL1 and XTAL2 DRIVEN BY 3.6 MHz CLOCK WILL GIVE 4.17  $\mu$ sec  $t_{CY}$ . THIS IS ACCEPTABLE FOR 8741A-8 PARTS AS WELL AS STANDARD PARTS.
3. AO MUST BE HELD LOW (i.e., = 0V) DURING PROGRAM/VERIFY MODES.
4. TEST 0 MUST BE HELD HIGH.

The 8741A EPROM can be programmed by either of two Intel products:

1. PROMPT-48 Microcomputer Design Aid, or
2. Universal PROM Programmer (UPP series) peripheral of the Intellec® Development System with a UPP-848 Personality Card.

DMA



**WAVEFORMS (Continued)**

**Table 2. UPI™ Instruction Set**

Mnemonic	Description	Bytes	Cycles
<b>ACCUMULATOR</b>			
ADD A, Rr	Add register to A	1	1
ADD A, @Rr	Add data memory to A	1	1
ADD A, #data	Add immediate to A	2	2
ADDC A, Rr	Add register to A with carry	1	1
ADDC A, @Rr	Add data memory to A with carry	1	1
ADDC A, #data	Add immediate to A with carry	2	2
ANL A, Rr	AND register to A	1	1
ANL A, @Rr	AND data memory to A	1	1
ANL A, #data	AND immediate to A	2	2
ORL A, Rr	OR register to A	1	1
ORL A, @Rr	OR data memory to A	1	1
ORL A, #data	OR immediate to A	2	2
XRL A, Rr	Exclusive OR register to A	1	1
XRL A, @Rr	Exclusive OR data memory to A	1	1
XRL A, #data	Exclusive OR immediate to A	2	2

Mnemonic	Description	Bytes	Cycles
<b>DATA MOVES</b>			
MOV A, Rr	Move register to A	1	1
MOV A, @Rr	Move data memory to A	1	1
MOV A, #data	Move immediate to A	2	2
MOV Rr, A	Move A to register	1	1
MOV @Rr, A	Move A to data memory	1	1
MOV Rr, #data	Move immediate to register	2	2
MOV @Rr, #data	Move immediate to data memory	2	2
MOV A, PSW	Move PSW to A	1	1
MOV PSW, A	Move A to PSW	1	1
XCH A, Rr	Exchange A and register	1	1
XCH A, @Rr	Exchange A and data memory	1	1
XCHD A, @Rr	Exchange digit of A and register	1	1
MOVP A, @A	Move to A from current page	1	2
MOVP3, A, @A	Move to A from page 3	1	2

Table 2. UPI™ Instruction Set (Continued)

Mnemonic	Description	Bytes	Cycles
<b>ACCUMULATOR</b>			
INC A	Increment A	1	1
DEC A	Decrement A	1	1
CLR A	Clear A	1	1
CPL A	Complement A	1	1
DA A	Decimal Adjust A	1	1
SWAP A	Swap nibbles of A	1	1
RL A	Rotate A left	1	1
RLC A	Rotate A left through carry	1	1
RR A	Rotate A right	1	1
RRC A	Rotate A right through carry	1	1
<b>INPUT/OUTPUT</b>			
IN A, Pp	Input port to A	1	2
OUTL Pp, A	Output A to port	1	2
ANL Pp, #data	AND immediate to port	2	2
ORL Pp, #data	OR immediate to port	2	2
IN A, DBB	Input DBB to A, clear IBF	1	1
OUT DBB, A	Output A to DBB, set OBF	1	1
MOV STS, A	A <sub>4</sub> -A <sub>7</sub> to Bits 4-7 of Status	1	1
MOVD A, Pp	Input Expander port to A	1	2
MOVD Pp, A	Output A to Expander port	1	2
ANLD Pp, A	AND A to Expander port	1	2
ORLD Pp, A	OR A to Expander port	1	2
<b>TIMER/COUNTER</b>			
MOV A, T	Read Timer/Counter	1	1
MOV T, A	Load Timer/Counter	1	1
STRT T	Start Timer	1	1
STRT CNT	start Counter	1	1
STOP TCNT	Stop Timer/Counter	1	1
EN TCNTI	Enable Timer/Counter Interrupt	1	1
DIS TCNTI	Disable Timer/Counter Interrupt	1	1
<b>CONTROL</b>			
EN DMA	Enable DMA Handshake Lines	1	1
EN I	Enable IBF Interrupt	1	1
DIS I	Disable IBF Interrupt	1	1
EN FLAGS	Enable Master Interrupts	1	1
SEL RB0	Select register bank 0	1	1
SEL RB1	Select register bank 1	1	1
NOP	No Operation	1	1

Mnemonic	Description	Bytes	Cycles
<b>REGISTERS</b>			
INC Rr	Increment register	1	1
INC @Rr	Increment data memory	1	1
DEC Rr	Decrement register	1	1
<b>SUBROUTINE</b>			
CALL addr	Jump to subroutine	2	2
RET	Return	1	2
RETR	Return and restore status	1	2
<b>FLAGS</b>			
CLR C	Clear Carry	1	1
CPL C	Complement Carry	1	1
CLR F0	Clear Flag 0	1	1
CPL F0	Complement Flag 0	1	1
CLR F1	Clear F1 Flag	1	1
CPL F1	Complement F1 Flag	1	1
<b>BRANCH</b>			
JMP addr	Jump unconditional	2	2
JMPP @A	Jump indirect	1	2
DJNZ Rr, addr	Decrement register and jump	2	2
JC addr	Jump on Carry=1	2	2
JNC addr	Jump on Carry=0	2	2
JZ addr	Jump on A Zero	2	2
JNZ addr	Jump on A not Zero	2	2
JT0 addr	Jump on T0=1	2	2
JNT0 addr	Jump on T0=0	2	2
JT1 addr	Jump on T1=1	2	2
JNT1 addr	Jump on T1=0	2	2
JF0 addr	Jump on F0 Flag=1	2	2
JF1 addr	Jump on F1 Flag=1	2	2
JTF addr	Jump on Timer Flag =1, Clear Flag	2	2
JNIBF addr	Jump on IBF Flag =0	2	2
JOBF addr	Jump on OBF Flag =1	2	2
JBb addr	Jump on Accumulator Bit	2	2

# 8042/8742 UNIVERSAL PERIPHERAL INTERFACE 8-BIT MICROCOMPUTER

- **8042/8742: 12 MHz**
- **Pin, Software and Architecturally Compatible with 8041A/8741A/8041AH**
- **8-Bit CPU plus ROM, RAM, I/O, Timer and Clock in a Single Package**
- **2048 x 8 ROM/EPROM, 128 x 8 RAM, 8-Bit Timer/Counter, 18 Programmable I/O Pins**
- **One 8-Bit Status and Two Data Registers for Asynchronous Slave-to-Master Interface**
- **DMA, Interrupt, or Polled Operation Supported**
- **Fully Compatible with MCS-48™, MCS-51™, MCS-80™, MCS-85™, and iAPX-86, 88 Microprocessor Families**
- **Interchangeable ROM and EPROM Versions**
- **Expandable I/O**
- **RAM Power-Down Capability**
- **Over 90 Instructions: 70% Single Byte**
- **Single 5V Supply**

The Intel 8042/8742 is a general-purpose Universal Peripheral Interface that allows the designer to grow his own customized solution for peripheral device control. It contains a low-cost microcomputer with 2K of program memory, 128 bytes of data memory, 8-bit CPU, I/O ports, 8-bit timer/counter, and clock generator in a single 40-pin package. Interface registers are included to enable the UPI device to function as a peripheral controller in the MCS-48™, MCS-51™, MCS-80™, MCS-85™, iAPX-88, iAPX-86 and other 8-, 16-bit systems.

The 8042/8742 is software, pin, and architecturally compatible with the 8041AH, 8741A. The 8042/8742 doubles the on-chip memory space to allow for additional features and performance to be incorporated in upgraded 8041AH/8741A designs. For new designs, the additional memory and performance of the 8042/8742 extends the UPI concept to more complex motor control tasks, 80-column printers and process control applications as examples.

To allow full user flexibility, the program memory is available as ROM in the 8042 version or as UV-erasable EPROM in the 8742 version. The 8742 and the 8042 are fully pin compatible for easy transition from prototype to production level designs. The 8642 is a one-time programmable (at the factory) 8742 which can be ordered as the first 25 pieces of a new 8042 order. The substitution of 8642's for 8042's allows for very fast turnaround for initial code verification and evaluation results.

The device has two 8-bit, TTL compatible I/O ports and two test inputs. Individual port lines can function as either inputs or outputs under software control. I/O can be expanded with the 8243 device which is directly compatible and has 16 I/O lines. An 8-bit programmable timer/counter is included in the UPI device for generating timing sequences or counting external inputs. Additional UPI features include: single 5V supply, low power standby mode (in the 8042), single-step mode for debug, and dual working register banks.

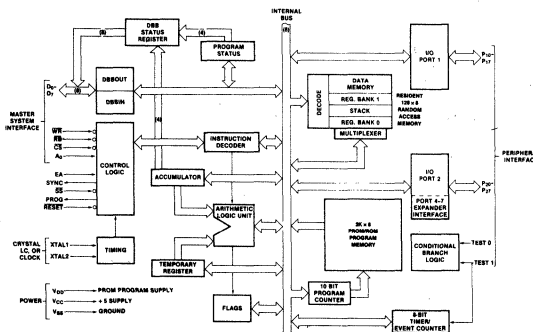


Figure 1. Block Diagram

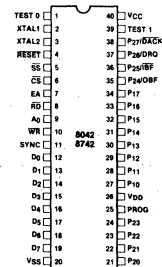


Figure 2. Pin Configuration

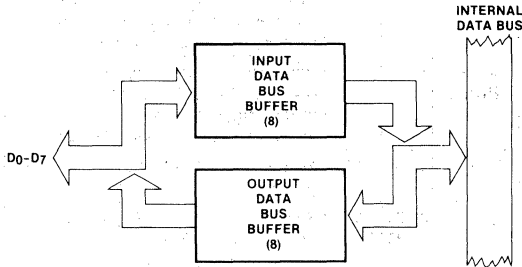
**Table 1. Pin Description**

Symbol	Pin No.	Type	Name and Function
TEST 0, TEST 1	1 39	I	<p><b>Test Inputs:</b> Input pins which can be directly tested using conditional branch instructions.</p> <p><b>Frequency Reference:</b> TEST 1 (T<sub>1</sub>) also functions as the event timer input (under software control). TEST 0 (T<sub>0</sub>) is used during PROM programming and verification in the 8742.</p>
XTAL 1, XTAL 2	2 3	I	<p><b>Inputs:</b> Inputs for a crystal, LC or an external timing signal to determine the internal oscillator frequency.</p>
RESET	4	I	<p><b>Reset:</b> Input used to reset status flip-flops and to set the program counter to zero.</p> <p>RESET is also used during PROM programming and verification.</p>
SS	5	I	<p><b>Single Step:</b> Single step input used in conjunction with the SYNC output to step the program through each instruction.</p>
CS	6	I	<p><b>Chip Select:</b> Chip select input used to select one UPI microcomputer out of several connected to a common data bus.</p>
EA	7	I	<p><b>External Access:</b> External access input which allows emulation, testing and PROM/ROM verification. This pin should be tied low if unused.</p>
RD	8	I	<p><b>Read:</b> I/O read input which enables the master CPU to read data and status words from the OUTPUT DATA BUS BUFFER or status register.</p>
A <sub>0</sub>	9	I	<p><b>Command/Data Select:</b> Address input used by the master processor to indicate whether byte transfer is data (A<sub>0</sub>=0, F1 is reset) or command (A<sub>0</sub>=1, F1 is set).</p>
WR	10	I	<p><b>Write:</b> I/O write input which enables the master CPU to write data and command words to the UPI INPUT DATA BUS BUFFER.</p>

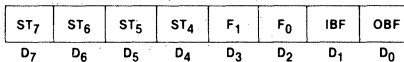
Symbol	Pin No.	Type	Name and Function
SYNC	11	O	<p><b>Output Clock:</b> Output signal which occurs once per UPI-42 instruction cycle. SYNC can be used as a strobe for external circuitry; it is also used to synchronize single step operation.</p>
D <sub>0</sub> -D <sub>7</sub> (BUS)	12-19	I/O	<p><b>Data Bus:</b> Three-state, bidirectional DATA BUS BUFFER lines used to interface the UPI-42 microcomputer to an 8-bit master system data bus.</p>
P <sub>10</sub> -P <sub>17</sub>	27-34	I/O	<p><b>Port 1:</b> 8-bit, PORT 1 quasi-bidirectional I/O lines.</p>
P <sub>20</sub> -P <sub>27</sub>	21-24 35-38	I/O	<p><b>Port 2:</b> 8-bit, PORT 2 quasi-bidirectional I/O lines. The lower 4 bits (P<sub>20</sub>-P<sub>23</sub>) interface directly to the 8243 I/O expander device and contain address and data information during PORT 4-7 access. The upper 4 bits (P<sub>24</sub>-P<sub>27</sub>) can be programmed to provide interrupt Request and DMA Handshake capability. Software control can configure P<sub>24</sub> as Output Buffer Full (OBF) interrupt, P<sub>25</sub> as Input Buffer Full (IBF) interrupt, P<sub>26</sub> as DMA Request (DRQ), and P<sub>27</sub> as DMA ACKnowledge (DACK).</p>
PROG	25	I/O	<p><b>Program:</b> Multifunction pin used as the program pulse input during PROM programming.</p> <p>During I/O expander access the PROG pin acts as an address/data strobe to the 8243. This pin should be tied high if unused.</p>
V <sub>CC</sub>	40		<p><b>Power:</b> +5V main power supply pin.</p>
V <sub>DD</sub>	26		<p><b>Power:</b> +5V during normal operation. +21V during programming operation. Low power standby pin in ROM version.</p>
V <sub>SS</sub>	20		<p><b>Ground:</b> Circuit ground potential.</p>

## UPI-42 FEATURES

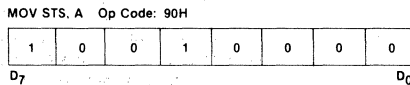
- Two Data Bus Buffers, one for input and one for output. This allows a much cleaner Master/Slave protocol.



- 8 Bits of Status



ST<sub>4</sub>-ST<sub>7</sub> are user definable status bits. These bits are defined by the "MOV STS, A" single byte, single cycle instruction. Bits 4-7 of the accumulator are moved to bits 4-7 of the status register. Bits 0-3 of the status register are not affected.



- $\overline{RD}$  and  $\overline{WR}$  are edge triggered. IBF, OBF, F<sub>1</sub> and INT change internally after the trailing edge of  $\overline{RD}$  or  $\overline{WR}$ .



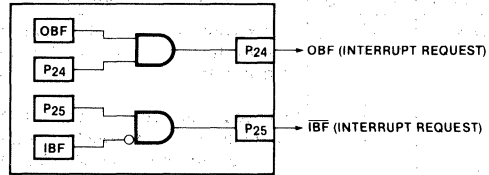
During the time that the host CPU is reading the status register, the 8042/8742 is prevented from updating this register or is 'locked out.'

- P<sub>24</sub> and P<sub>25</sub> are port pins or Buffer Flag pins which can be used to interrupt a master processor. These pins default to port pins on Reset.

If the "EN FLAGS" instruction has been executed, P<sub>24</sub> becomes the OBF (Output Buffer Full) pin. A "1" written to P<sub>24</sub> enables the OBF pin (the pin outputs the OBF Status Bit). A "0" written to P<sub>24</sub> disables the OBF pin (the pin remains low). This pin can be used to indicate that valid data is available from the UPI-41A (in Output Data Bus Buffer).

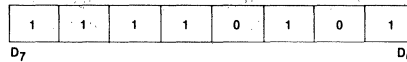
If "EN FLAGS" has been executed, P<sub>25</sub> becomes the  $\overline{IBF}$  (Input Buffer Full) pin. A "1" written to P<sub>25</sub> enables the  $\overline{IBF}$  pin (the pin outputs the inverse of the IBF Status Bit). A "0" written to P<sub>25</sub> disables the  $\overline{IBF}$

pin (the pin remains low). This pin can be used to indicate that the UPI-42 is ready for data.



DATA BUS BUFFER INTERRUPT CAPABILITY

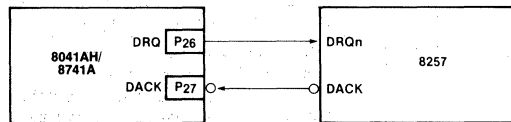
EN FLAGS Op Code: 0F5H



- P<sub>26</sub> and P<sub>27</sub> are port pins or DMA handshake pins for use with a DMA controller. These pins default to port pins on Reset.

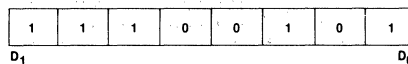
If the "EN DMA" instruction has been executed, P<sub>26</sub> becomes the DRQ (DMA ReQuest) pin. A "1" written to P<sub>26</sub> causes a DMA request (DRQ is activated). DRQ is deactivated by DACK · RD, DACK · WR, or execution of the "EN DMA" instruction.

If "EN DMA" has been executed, P<sub>27</sub> becomes the  $\overline{DACK}$  (DMA ACKnowledge) pin. This pin acts as a chip select input for the Data Bus Buffer registers during DMA transfers.



DMA HANDSHAKE CAPABILITY

EN DMA Op Code: 0E5H



- The RESET input on the 8042/8742 includes a 2-stage synchronizer to support reliable reset operation for 12 MHz operation.
- When EA is enabled on the 8042/8742, the program counter is placed on Port 1 and the lower three bits of Port 2 (MSB = P<sub>22</sub>, LSB = P<sub>10</sub>). On the 8042/8742 this information is multiplexed with PORT DATA (see port timing diagrams at end of this data sheet).
- The 8042/8742 supports single step mode as described in the pin description section.

APPLICATIONS

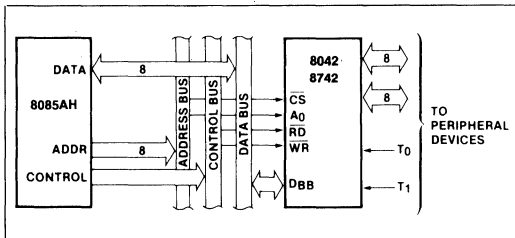


Figure 3. 8085AH-8042/8742 Interface

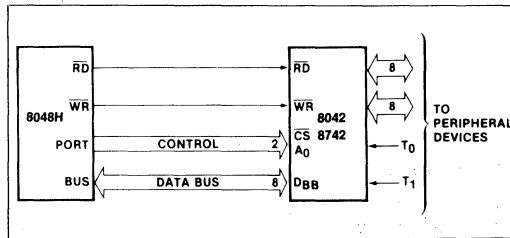


Figure 4. 8048H-8042/8742 Interface

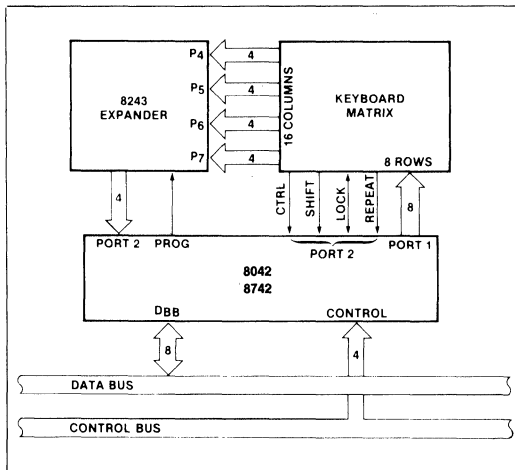


Figure 5. 8042/8742-8243 Keyboard Scanner

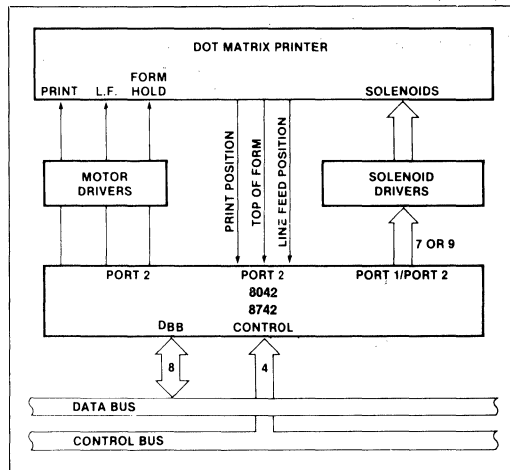


Figure 6. 8042/8742 80-Column Matrix Printer Interface

PROGRAMMING, VERIFYING, AND ERASING THE 8742 EPROM

Programming Verification

In brief, the programming process consists of: activating the program mode, applying an address, latching the address, applying data, and applying a programming pulse. Each word is programmed completely before moving on to the next and is followed by a verification step. The following is a list of the pins used for programming and a description of their functions:

Pin	Function
XTAL 1	Clock Input (1 to 12MHz)
Reset	Initialization and Address Latching
Test 0	Selection of Program or Verify Mode
EA	Activation of Program/Verify Modes
BUS	Address and Data Input Data Output During Verify
P20-1	Address Input
V <sub>DD</sub>	Programming Power Supply
PROG	Program Pulse Input

WARNING

An attempt to program a missocketed 8742 will result in severe damage to the part. An indication of a properly socketed part is the appearance of the SYNC clock output. The lack of this clock may be used to disable the programmer.

The Program/Verify sequence is:

1. A<sub>0</sub> = 0V, CS = 5V, EA = 5V, RESET = 0V, TEST0 = 5V, V<sub>DD</sub> = 5V, clock applied or internal oscillator operating, BUS and PROG floating.
2. Insert 8742 in programming socket
3. TEST 0 = 0v (select program mode)
4. EA = 21V (active program mode)\*
5. Address applied to BUS and P20-22
6. RESET = 5v (latch address)
7. Data applied to BUS\*\*
8. V<sub>DD</sub> = 21V (programming power)\*\*
9. PROG = 0v followed by one 50 ms pulse to 21V\*\*
10. V<sub>DD</sub> = 5v
11. TEST 0 = 5v (verify mode)



12. Read and verify data on BUS
13. TEST 0 = 0v
14. RESET = 0v and repeat from step 5
15. Programmer should be at conditions of step 1 when 8742 is removed from socket

\*When verifying ROM, EA = 12V.

\*\*Not used in verifying ROM procedure.

#### 8742 Erasure Characteristics

The erasure characteristics of the 8742 are such that erasure begins to occur when exposed to light with wavelengths shorter than approximately 4000 Angstroms (Å). It should be noted that sunlight and certain types of fluorescent lamps have wavelengths in the 3000-4000Å range. Data show that constant exposure to room level fluorescent lighting could erase the typical 8742 in approximately 3 years while it would take ap-

proximately one week to cause erasure when exposed to direct sunlight. If the 8742 is to be exposed to these types of lighting conditions for extended periods of time, opaque labels are available from Intel which should be placed over the 8742 window to prevent unintentional erasure.

The recommended erasure procedure for the 8742 is exposure to shortwave ultraviolet light which has a wavelength of 2537Å. The integrated dose (i.e., UV intensity x exposure time) for erasure should be a minimum of 15 w-sec/cm<sup>2</sup>. The erasure time with this dosage is approximately 15 to 20 minutes using an ultraviolet lamp with a 12,000 μW/cm<sup>2</sup> power rating. The 8742 should be placed within one inch of the lamp tubes during erasure. Some lamps have a filter on their tubes which should be removed before erasure.

**ABSOLUTE MAXIMUM RATINGS\***

Ambient Temperature Under Bias ..... 0°C to 70°C  
 Storage Temperature ..... - 65°C to + 150°C  
 Voltage on Any Pin With Respect  
     to Ground ..... -0.5V to +7V  
 Power Dissipation ..... 1.5 Watt

*\*NOTICE: Stresses above those listed under "Absolute Maximum Ratings" may cause permanent damage to the device. This is a stress rating only and functional operation of the device at these or any other conditions above those indicated in the operational sections of this specification is not implied. Exposure to absolute maximum rating conditions for extended periods may affect device reliability.*

**D.C. CHARACTERISTICS** ( $T_A = 0^\circ$  to  $+70^\circ\text{C}$ ,  $V_{CC} = V_{DD} = +5V \pm 10\%$ )

Symbol	Parameter	8042		8742/8642		Units	Notes
		Min.	Max.	Min.	Max.		
$V_{IL}$	Input Low Voltage (Except XTAL1, XTAL2, RESET)	-0.5	0.8	-0.5	0.8	V	
$V_{IL1}$	Input Low Voltage (XTAL1, XTAL2, RESET)	-0.5	0.6	-0.5	0.6	V	
$V_{IH}$	Input High Voltage (Except XTAL1, XTAL2, RESET)	2.2	$V_{CC}$	2.2	$V_{CC}$	V	
$V_{IH1}$	Input High Voltage (XTAL1, XTAL2, RESET)	3.8	$V_{CC}$	3.8	$V_{CC}$	V	
$V_{OL}$	Output Low Voltage (D <sub>0</sub> -D <sub>7</sub> )		0.45		0.45	V	$I_{OL} = 2.0\text{ mA}$
$V_{OL1}$	Output Low Voltage (P <sub>10</sub> P <sub>17</sub> , P <sub>20</sub> P <sub>27</sub> , Sync)		0.45		0.45	V	$I_{OL} = 1.6\text{ mA}$
$V_{OL2}$	Output Low Voltage (PROG)		0.45		0.45	V	$I_{OL} = 1.0\text{ mA}$
$V_{OH}$	Output High Voltage (D <sub>0</sub> -D <sub>7</sub> )	2.4		2.4		V	$I_{OH} = -400\ \mu\text{A}$
$V_{OH1}$	Output High Voltage (All Other Outputs)	2.4		2.4		V	$I_{OH} = -50\ \mu\text{A}$
$I_{IL}$	Input Leakage Current (T <sub>0</sub> , T <sub>1</sub> , RD, WR, CS, A <sub>0</sub> , EA)		$\pm 10$		$\pm 10$	$\mu\text{A}$	$V_{SS} \leq V_{IN} \leq V_{CC}$
$I_{OZ}$	Output Leakage Current (D <sub>0</sub> -D <sub>7</sub> , High Z State)		$\pm 10$		$\pm 10$	$\mu\text{A}$	$V_{SS} + 0.45 \leq V_{OUT} \leq V_{CC}$
$I_{LI}$	Low Input Load Current (P <sub>10</sub> P <sub>17</sub> , P <sub>20</sub> P <sub>27</sub> )		0.3		0.3	mA	$V_{IL} = 0.8\text{V}$
$I_{LI1}$	Low Input Load Current (RESET, SS)		0.2		0.2	mA	$V_{IL} = 0.8\text{V}$
$I_{DD}$	$V_{DD}$ Supply Current		15		15	mA	Typical = 5 mA
$I_{CC} + I_{DD}$	Total Supply Current		125		125	mA	Typical = 60 mA
$I_{IH}$	Input Leakage Current		100		100	$\mu\text{A}$	$V_{IN} = V_{CC}$
$C_{IN}$	Input Capacitance		10		10	pF	
$C_{I/O}$	I/O Capacitance		20		20	pF	

**D.C. CHARACTERISTICS—PROGRAMMING** ( $T_A = 25^\circ\text{C} \pm 5^\circ\text{C}$ ,  $V_{CC} = 5V \pm 5\%$ ,  $V_{DD} = 21V \pm 1V$ )

Symbol	Parameter	Min.	Max.	Unit	Test Conditions
$V_{DOH}$	$V_{DD}$ Program Voltage High Level	20.0	22.0	V	
$V_{DDL}$	$V_{DD}$ Voltage Low Level	4.75	5.25	V	
$V_{PH}$	PROG Program Voltage High Level	21.5	24.5	V	
$V_{PL}$	PROG Voltage Low Level		0.2	V	
$V_{EAH}$	EA Program or Verify Voltage High Level	21.5	24.5	V	
$V_{EAL}$	EA Voltage Low Level		5.25	V	
$I_{DD}$	$V_{DD}$ High Voltage Supply Current		30.0	mA	
$I_{PROG}$	PROG High Voltage Supply Current		16.0	mA	
$I_{EA}$	EA High Voltage Supply Current		1.0	mA	

**A.C. CHARACTERISTICS** ( $T_A = 0^\circ\text{C}$  to  $+70^\circ\text{C}$ ,  $V_{SS} = 0\text{V}$ ,  $V_{CC} = V_{DD} = +5\text{V} \pm 10\%$ )**DBB READ**

Symbol	Parameter	8042		8642/8742		Units
		Min.	Max.	Min.	Max.	
$t_{AR}$	CS, $A_0$ Setup to RD $\downarrow$	0		0		ns
$t_{RA}$	CS, $A_0$ Hold After RD $\downarrow$	0		0		ns
$t_{RR}$	RD Pulse Width	160		160		ns
$t_{AD}$	CS, $A_0$ to Data Out Delay		130		130	ns <sup>[1]</sup>
$t_{RD}$	RD $\downarrow$ to Data Out Delay		130		130	ns <sup>[1]</sup>
$t_{DF}$	RD $\downarrow$ to Data Float Delay		85		85	ns
$t_{CY}$	Cycle Time	1.25	15	1.25	15	$\mu\text{s}$ <sup>[2]</sup>

**DBB WRITE**

Symbol	Parameter	Min.	Max.	Min.	Max.	Units
$t_{AW}$	CS, $A_0$ Setup to WR $\downarrow$	0		0		ns
$t_{WA}$	CS, $A_0$ Hold After WR $\downarrow$	0		0		ns
$t_{WW}$	WR Pulse Width	160		260		ns
$t_{DW}$	Data Setup to WR $\downarrow$	130		150		ns
$t_{WD}$	Data Hold After WR $\downarrow$	0		0		ns

**NOTES:**

- $C_L = 100$  pF.
- 12 MHz XTAL.

**A.C. CHARACTERISTICS—PROGRAMMING** ( $T_A = 25^\circ\text{C} \pm 5^\circ\text{C}$ ,  $V_{CC} = 5\text{V} \pm 5\%$ ,  $V_{DD} = 21\text{V} \pm 1\text{V}$ )

Symbol	Parameter	Min.	Max.	Unit	Test Conditions
$t_{AW}$	Address Setup Time to $\overline{\text{RESET}} \downarrow$	4tcy			
$t_{WA}$	Address Hold Time After $\overline{\text{RESET}} \downarrow$	4tcy			
$t_{DW}$	Data in Setup Time to PROG $\downarrow$	4tcy			
$t_{WD}$	Data in Hold Time After PROG $\downarrow$	4tcy			
$t_{PH}$	$\overline{\text{RESET}}$ Hold Time to Verify	4tcy			
$t_{VDDW}$	$V_{DD}$ Setup Time to PROG $\downarrow$	4tcy			
$t_{VDDH}$	$V_{DD}$ Hold Time After PROG $\downarrow$	0			
$t_{PW}$	Program Pulse Width	50	60	mS	
$t_{TW}$	Test 0 Setup Time for Program Mode	4tcy			
$t_{WT}$	Test 0 Hold Time After Program Mode	4tcy			
$t_{DO}$	Test 0 to Data Out Delay		4tcy		
$t_{WW}$	$\overline{\text{RESET}}$ Pulse Width to Latch Address	4tcy			
$t_r, t_f$	$V_{DD}$ and PROG Rise and Fall Times	0.5	2.0	$\mu\text{s}$	
$t_{CY}$	CPU Operation Cycle Time	5.0		$\mu\text{s}$	
$t_{RE}$	$\overline{\text{RESET}}$ Setup Time Before EA $\downarrow$	4tcy			

Note: If TEST 0 is high,  $t_{DO}$  can be triggered by  $\overline{\text{RESET}} \downarrow$ .

**A.C. CHARACTERISTICS DMA**

Symbol	Parameter	8042		8642/8742		Units
		Min.	Max.	Min.	Max.	
$t_{ACC}$	DACK to WR or RD	0		0		ns
$t_{CAC}$	RD or WR to DACK	0		0		ns
$t_{ACD}$	DACK to Data Valid		130		130	ns <sup>[1]</sup>
$t_{CRQ}$	RD or WR to DRQ Cleared		90		90	ns

**NOTE:**

 1.  $C_L = 150$  pF.

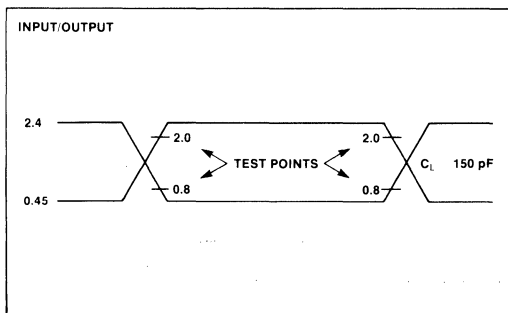
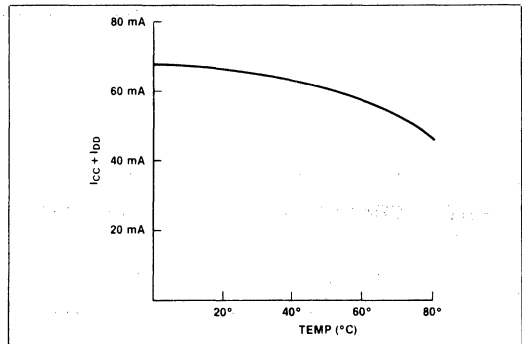
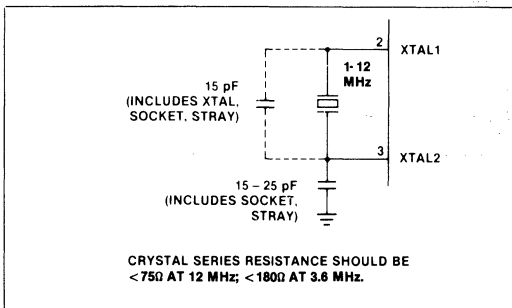
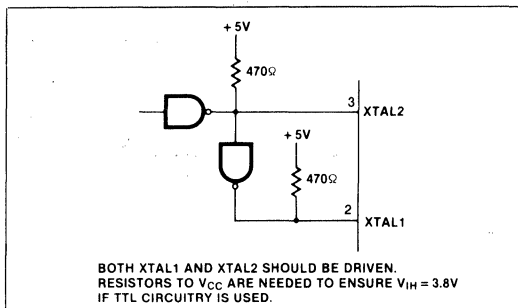
**A.C. CHARACTERISTICS PORT 2 ( $T_A = 0^\circ\text{C}$  to  $+70^\circ\text{C}$ ,  $V_{CC} = +5V \pm 10\%$ )**

Symbol	Parameter	8042		8642/8742		Units
		Min.	Max.	Min.	Max.	
$t_{CP}$	Port Control Setup Before Falling Edge of PROG	100		100		ns <sup>[1]</sup>
$t_{PC}$	Port Control Hold After Falling Edge of PROG	60		60		ns <sup>[2]</sup>
$t_{PR}$	PROG to Time P2 Input Must Be Valid		650		650	ns <sup>[1]</sup>
$t_{PF}$	Input Data Hold Time	0	150	0	150	ns <sup>[2]</sup>
$t_{DP}$	Output Data Setup Time	200		200		ns <sup>[1]</sup>
$t_{PD}$	Output Data Hold Time	60		60		ns <sup>[2]</sup>
$t_{PP}$	PROG Pulse Width	700		700		ns

**NOTES:**

 1.  $C_L = 80$  pF.

 2.  $C_L = 20$  pF.

**A.C. TESTING INPUT, OUTPUT WAVEFORM**

**TYPICAL 8042/8742 CURRENT**

**CRYSTAL OSCILLATOR MODE**

**DRIVING FROM EXTERNAL SOURCE**


**LC OSCILLATOR MODE**

L	C	NOMINAL f
45 $\mu$ H	20 pF	5.2 MHz
120 $\mu$ H	20 pF	3.2 MHz

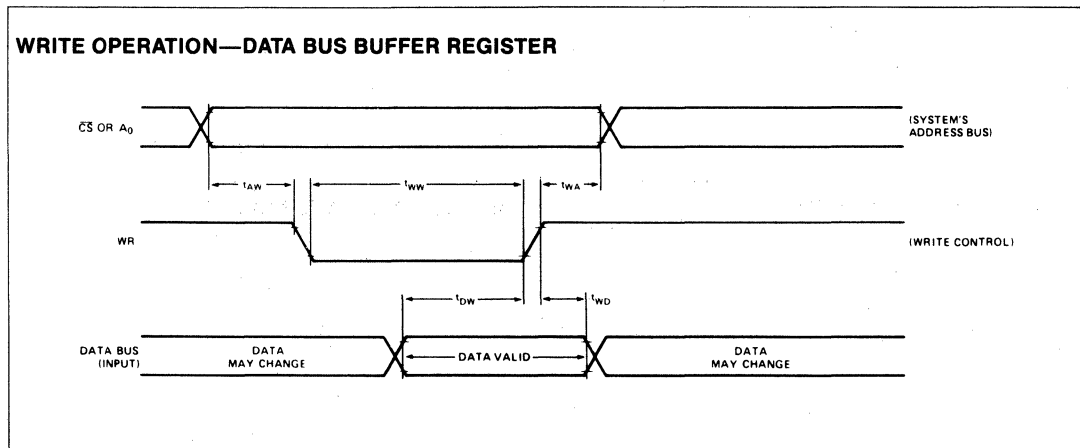
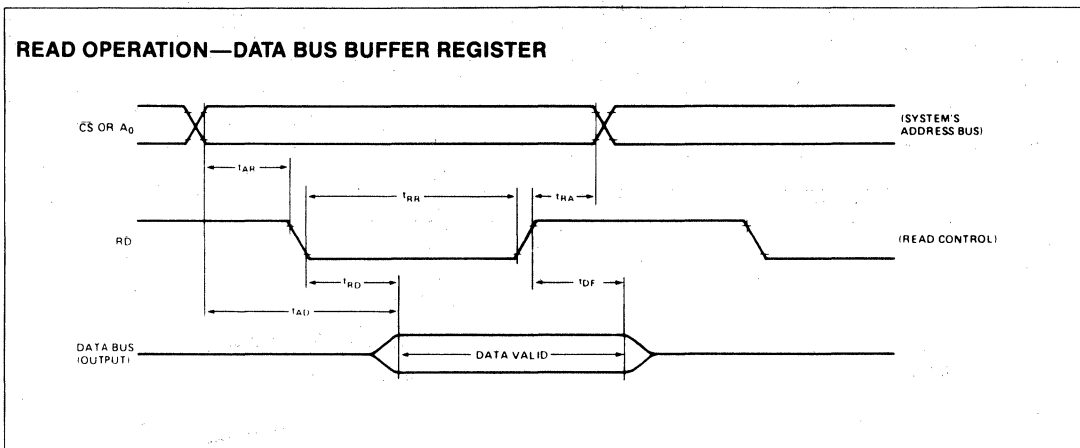
$$f = \frac{1}{2\pi\sqrt{LC}}$$

$$C' = \frac{C + 3C_{pp}}{2}$$

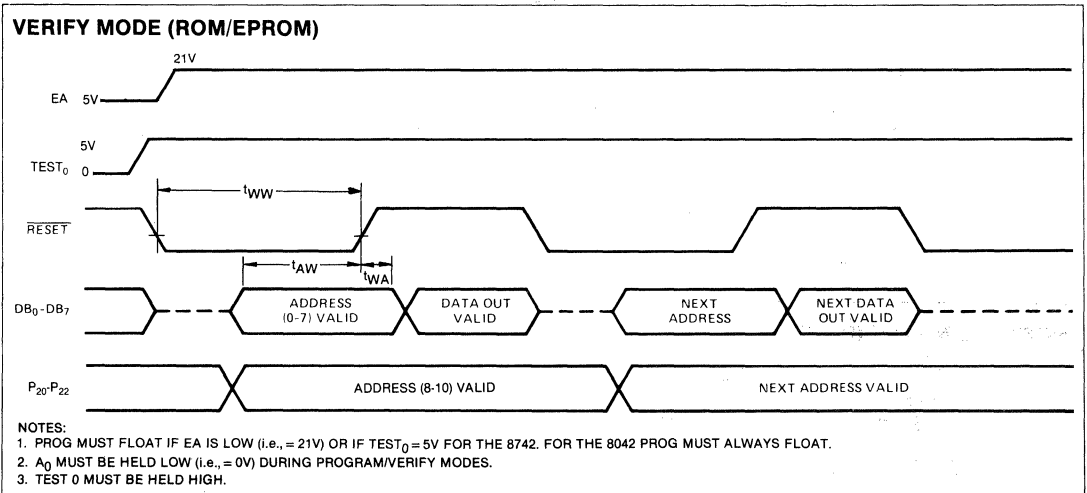
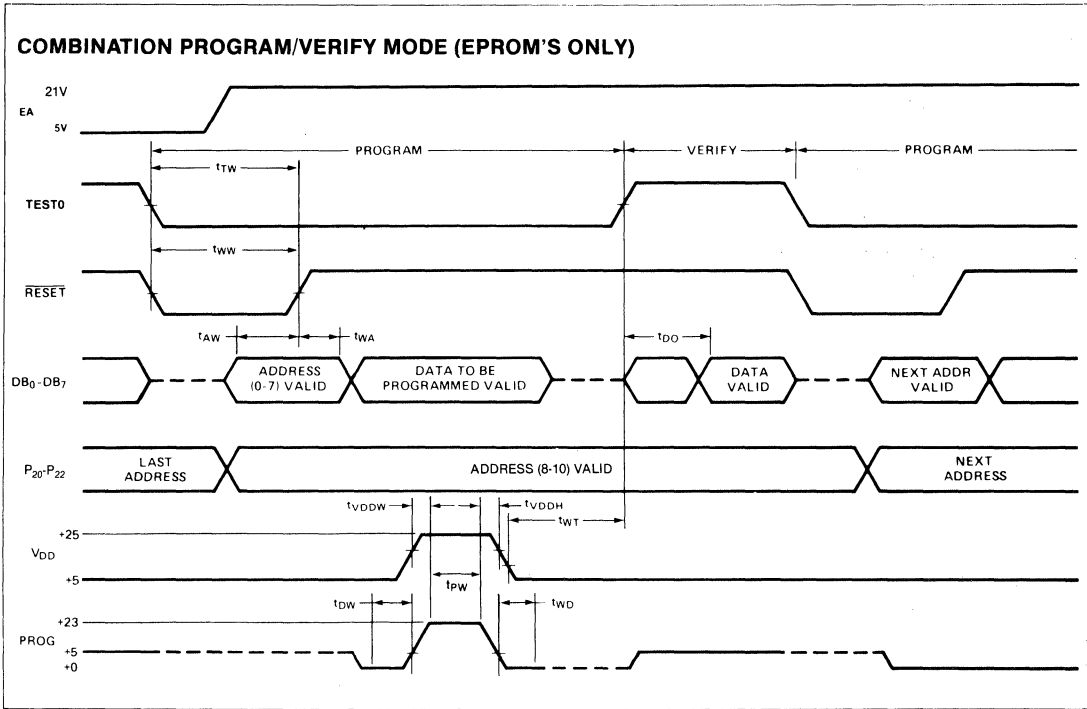
$C_{pp} \approx 5 - 10$  pF PIN-TO-PIN CAPACITANCE

EACH C SHOULD BE APPROXIMATELY 20 pF, INCLUDING STRAY CAPACITANCE.

**WAVEFORMS**



WAVEFORMS (Continued)



The 8742 EPROM can be programmed by the following Intel product:

1. Universal PROM Programmer (UPP series) peripheral of the Intellec® Development System with a UPP-549 Personality Card.

WAVEFORMS (Continued)

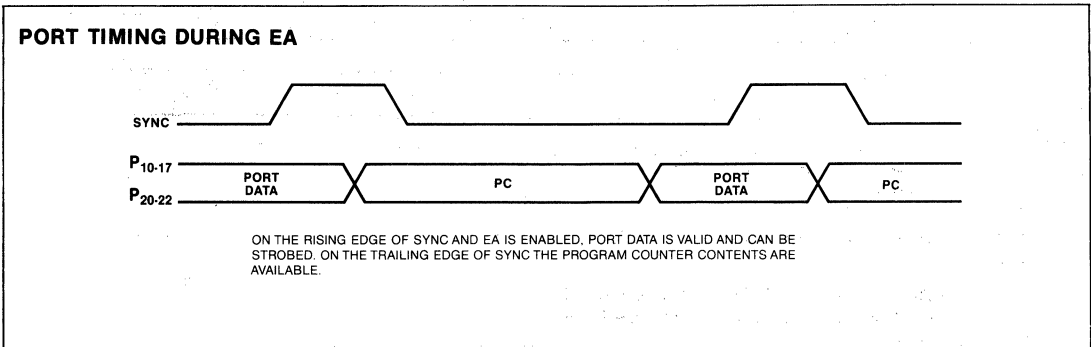
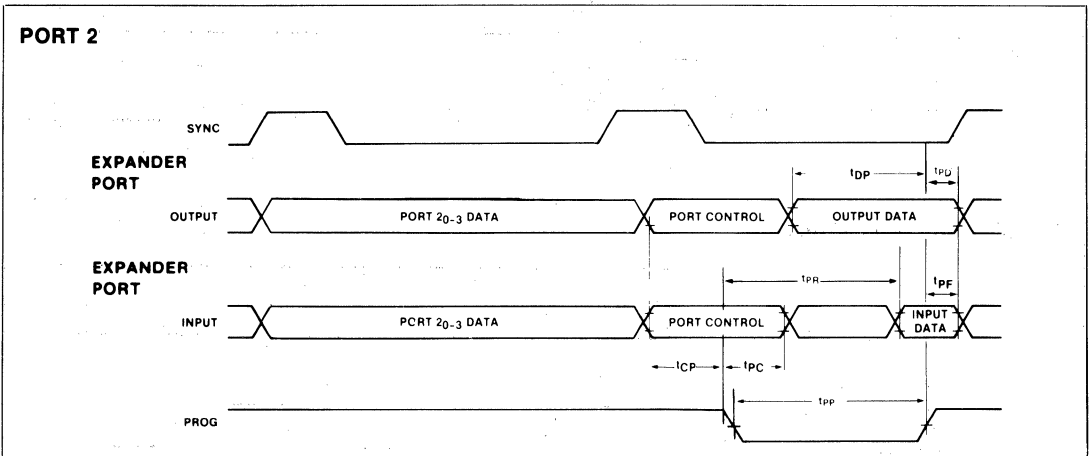
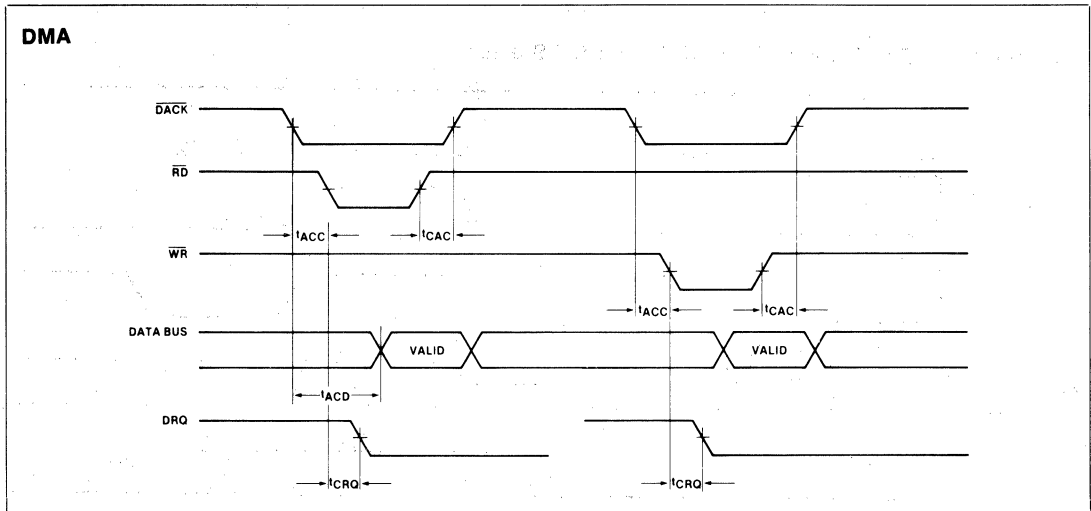


Table 2. UPI™ Instruction Set

Mnemonic	Description	Bytes	Cycles
<b>ACCUMULATOR</b>			
ADD A, Rr	Add register to A	1	1
ADD A, @Rr	Add data memory to A	1	1
ADD A, #data	Add immediate to A	2	2
ADDC A, Rr	Add register to A with carry	1	1
ADDC A, @Rr	Add data memory to A with carry	1	1
ADDC A, #data	Add immediate to A with carry	2	2
ANL A, Rr	AND register to A	1	1
ANL A, @Rr	AND data memory to A	1	1
ANL A, #data	AND immediate to A	2	2
ORL A, Rr	OR register to A	1	1
ORL A, @Rr	OR data memory to A	1	1
ORL A, #data	OR immediate to A	2	2
XRL A, Rr	Exclusive OR register to A	1	1
XRL A, @Rr	Exclusive OR data memory to A	1	1
XRL A, #data	Exclusive OR immediate to A	2	2
INC A	Increment A	1	1
DEC A	Decrement A	1	1
CLR A	Clear A	1	1
CPL A	Complement A	1	1
DA A	Decimal Adjust A	1	1
SWAP A	Swap nibbles of A	1	1
RL A	Rotate A left	1	1
RLC A	Rotate A left through carry	1	1
RR A	Rotate A right	1	1
RRC A	Rotate A right through carry	1	1
<b>INPUT/OUTPUT</b>			
IN A, Pp	Input port to A	1	2
OUTL Pp, A	Output A to port	1	2
ANL Pp, #data	AND immediate to port	2	2
ORL Pp, #data	OR immediate to port	2	2
IN A, DBB	Input DBB to A, clear IBF	1	1
OUT DBB, A	Output A to DBB, set OBF	1	1
MOV STS, A	A <sub>4</sub> -A <sub>7</sub> to Bits 4-7 of Status	1	1
MOVD A, Pp	Input Expander port to A	1	2
MOVD Pp, A	Output A to Expander port	1	2
ANLD Pp, A	AND A to Expander port	1	2
ORLD Pp, A	OR A to Expander port	1	2
<b>DATA MOVES</b>			
MOV A, Rr	Move register to A	1	1
MOV A, @Rr	Move data memory to A	1	1
MOV A, #data	Move immediate to A	2	2
MOV Rr, A	Move A to register	1	1
MOV @Rr, A	Move A to data memory	1	1
MOV Rr, #data	Move immediate to register	2	2
MOV @Rr, #data	Move immediate to data memory	2	2
MOV A, PSW	Move PSW to A	1	1
MOV PSW, A	Move A to PSW	1	1
XCH A, Rr	Exchange A and register	1	1
XCH A, @Rr	Exchange A and data memory	1	1
XCHD A, @Rr	Exchange digit of A and register	1	1
MOVP A, @A	Move to A from current page	1	2
MOVP3, A, @A	Move to A from page 3	1	2
<b>TIMER/COUNTER</b>			
MOV A, T	Read Timer/Counter	1	1
MOV T, A	Load Timer/Counter	1	1
STRT T	Start Timer	1	1
STRT CNT	start Counter	1	1
STOP TCNT	Stop Timer/Counter	1	1
EN TCNTI	Enable Timer/Counter Interrupt	1	1
DIS TCNTI	Disable Timer/Counter Interrupt	1	1
<b>CONTROL</b>			
EN DMA	Enable DMA Handshake Lines	1	1
EN I	Enable IBF Interrupt	1	1
DIS I	Disable IBF Interrupt	1	1
EN FLAGS	Enable Master Interrupts	1	1
SEL RB0	Select register bank 0	1	1
SEL RB1	Select register bank 1	1	1
NOP	No Operation	1	1
<b>REGISTERS</b>			
INC Rr	Increment register	1	1
INC @Rr	Increment data memory	1	1
DEC Rr	Decrement register	1	1
<b>SUBROUTINE</b>			
CALL addr	Jump to subroutine	2	2
RET	Return	1	2
RETR	Return and restore status	1	2



Table 2. UPI™ Instruction Set (Continued)

Mnemonic	Description	Bytes	Cycles
<b>FLAGS</b>			
CLR C	Clear Carry	1	1
CPL C	Complement Carry	1	1
CLR F0	Clear Flag 0	1	1
CPL F0	Complement Flag 0	1	1
CLR F1	Clear F1 Flag	1	1
CPL F1	Complement F1 Flag	1	1
<b>BRANCH</b>			
JMP addr	Jump unconditional	2	2
JMPP @A	Jump indirect	1	2
DJNZ Rr, addr	Decrement register and jump	2	2
JC addr	Jump on Carry=1	2	2
JNC addr	Jump on Carry=0	2	2
JZ addr	Jump on A Zero	2	2
JNZ addr	Jump on A not Zero	2	2
JT0 addr	Jump on T0=1	2	2
JNT0 addr	Jump on T0=0	2	2
JT1 addr	Jump on T1=1	2	2
JNT1 addr	Jump on T1=0	2	2
JF0 addr	Jump on F0 Flag=1	2	2
JF1 addr	Jump on F1 Flag=1	2	2
JTF addr	Jump on Timer Flag = 1, Clear Flag	2	2
JNIBF addr	Jump on IBF Flag =0	2	2
JOBF addr	Jump on OBF Flag =1	2	2
JBb addr	Jump on Accumula- tor Bit	2	2

# 8231A

## ARITHMETIC PROCESSING UNIT

- Fixed Point Single and Double Precision (16/32 Bit)
- Floating Point Single Precision (32 Bit)
- Binary Data Formats
- Add, Subtract, Multiply and Divide
- Trigonometric and Inverse Trigonometric Functions
- Square Roots, Logarithms, Exponentiation
- Float to Fixed and Fixed to Float Conversions
- Stack Oriented Operand Storage
- Compatible with MCS-80™ and MCS-85™ Microprocessor Families
- Direct Memory Access or Programmed I/O Data Transfers
- End of Execution Signal
- General Purpose 8-Bit Data Bus Interface
- Standard 24 Pin Package
- + 12 Volt and + 5 Volt Power Supplies
- Advanced N-Channel Silicon Gate HMOS Technology

The Intel® 8231A Arithmetic Processing Unit (APU) is a monolithic HMOS LSI device that provides high performance fixed and floating point arithmetic and floating point trigonometric operations. It may be used to enhance the mathematical capability of a wide variety of processor-oriented systems. Chebyshev polynomials are used in the implementation of the APU algorithms.

All transfers, including operand, result, status and command information, take place over an 8-bit bidirectional data bus. Operands are pushed onto an internal stack and commands are issued to perform operations on the data in the stack. Results are then available to be retrieved from the stack.

Transfers to and from the APU may be handled by the associated processor using conventional programmed I/O, or may be handled by a direct memory access controller for improved performance. Upon completion of each command, the APU issues an end of execution signal that may be used as an interrupt by the CPU to help coordinate program execution.

In January 1981 Intel will be converting from 8231 to 8231A. The 8231A provides enhancements over the 8231 to allow use in both asynchronous and synchronous systems.

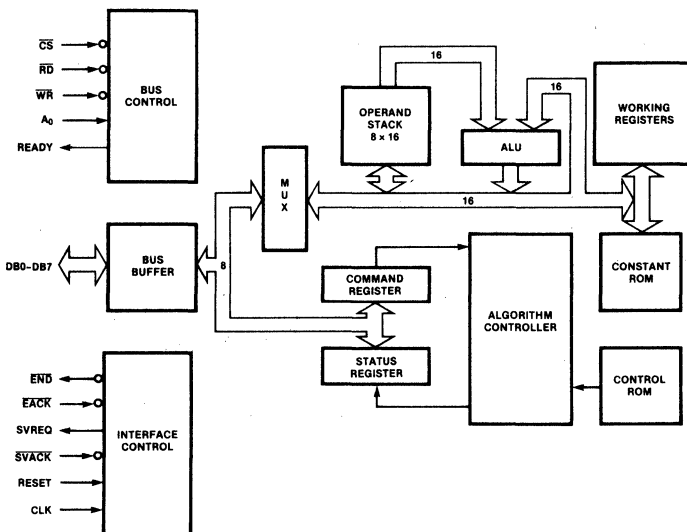


Figure 1. Block Diagram

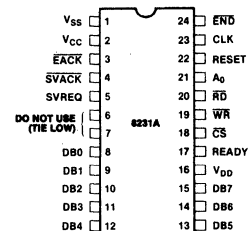


Figure 2. Pin Configuration

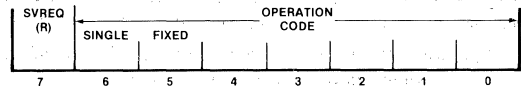
Table 1. Pin Description

Symbol	Pin No.	Type	Name and Function																				
V <sub>CC</sub>	2		<b>Power:</b> +5 Volt power supply.																				
V <sub>DD</sub>	16		<b>Power:</b> +12 Volt power supply.																				
V <sub>SS</sub>	1		<b>Ground.</b>																				
CLK	23	I	<b>Clock:</b> An external, TTL compatible, timing source is applied to the CLK pin.																				
RESET	22	I	<b>Reset:</b> The active high reset signal provides initialization for the chip. RESET also terminates any operation in progress. RESET clears the status register and places the 8231A into the idle state. Stack contents and command registers are not affected (5 clock cycles).																				
$\overline{CS}$	18	I	<b>Chip Select:</b> $\overline{CS}$ is an active low input signal which selects the 8231A and enables communication with the data bus.																				
A <sub>0</sub>	21	I	<b>Address:</b> In conjunction with the $\overline{RD}$ and $\overline{WR}$ signals, the A <sub>0</sub> control line establishes the type of communication that is to be performed with the 8231A as shown below:																				
<table border="1"> <thead> <tr> <th>A<sub>0</sub></th> <th><math>\overline{RD}</math></th> <th><math>\overline{WR}</math></th> <th>Function</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>1</td> <td>0</td> <td>Enter data byte into stack</td> </tr> <tr> <td>0</td> <td>0</td> <td>1</td> <td>Read data byte from stack</td> </tr> <tr> <td>1</td> <td>1</td> <td>0</td> <td>Enter command</td> </tr> <tr> <td>1</td> <td>0</td> <td>1</td> <td>Read status</td> </tr> </tbody> </table>				A <sub>0</sub>	$\overline{RD}$	$\overline{WR}$	Function	0	1	0	Enter data byte into stack	0	0	1	Read data byte from stack	1	1	0	Enter command	1	0	1	Read status
A <sub>0</sub>	$\overline{RD}$	$\overline{WR}$	Function																				
0	1	0	Enter data byte into stack																				
0	0	1	Read data byte from stack																				
1	1	0	Enter command																				
1	0	1	Read status																				
$\overline{RD}$	20	I	<b>Read:</b> This active low input indicates that data or status is to be read from the 8231A if $\overline{CS}$ is low.																				
$\overline{WR}$	19	I	<b>Write:</b> This active low input indicates that data or a command is to be written into the 8231A if $\overline{CS}$ is low.																				
$\overline{EACK}$	3	I	<b>End of Execution:</b> This active low input clears the end of execution output signal ( $\overline{END}$ ). If $\overline{EACK}$ is tied low, the $\overline{END}$ output will be a pulse that is one clock period wide.																				
$\overline{SVACK}$	4	I	<b>Service Request:</b> This active low input clears the service request output (SVREQ).																				
$\overline{END}$	24	O	<b>End:</b> This active low, open-drain output indicates that execution of the previously entered command is complete. It can be used as an interrupt request and is cleared by $\overline{EACK}$ , RESET or any read or write access to the 8231.																				

Symbol	Pin No.	Type	Name and Function
SVREQ	5	O	<b>Service Request:</b> This active high output signal indicates that command execution is complete and that post execution service was requested in the previous command byte. It is cleared by $\overline{SVACK}$ , the next command output to the device, or by RESET.
READY	17	O	<b>Ready:</b> This active high output indicates that the 8231A is able to accept communication with the data bus. When an attempt is made to read data, write data or to enter a new command while the 8231A is executing a command, READY goes low until execution of the current command is complete (See READY Operation, p. 5).
DB0-DB7	8-15	I/O	<b>Data Bus:</b> These eight bidirectional lines provide for transfer of commands, status and data between the 8231A and the CPU. The 8231A can drive the data bus only when $\overline{CS}$ and $\overline{RD}$ are low.

COMMAND STRUCTURE

Each command entered into the 8231A consists of a single 8-bit byte having the format illustrated below:



Bits 0-4 select the operation to be performed as shown in the table. Bits 5-6 select the data format appropriate to the selected operation. If bit 5 is a 1, a fixed point data format is specified. If bit 5 is a 0, floating point format is specified. Bit 6 selects the precision of the data to be operated upon by fixed point commands only (if bit 5 = 0, bit 6 must be 0). If bit 6 is a 1, single-precision (16-bit) operands are assumed. If bit 6 is a 0, double-precision (32-bit) operands are indicated. Results are undefined for all illegal combinations of bits in the command byte. Bit 7 indicates whether a service request is to be issued after the command is executed. If bit 7 is a 1, the service request output (SVREQ) will go high at the conclusion of the command and will remain high until reset by a low level on the service acknowledge pin ( $\overline{SVACK}$ ) or until completion of execution of the succeeding command where service request (bit 7) is 0. Each command issued to the 8231A requests post execution service based upon the state of bit 7 in the command byte. When bit 7 is a 0, SVREQ remains low.

**Table 2. 32-Bit Floating Point Instructions**

Instruction	Description	Hex <sup>(1)</sup> Code	Stack Contents <sup>(2)</sup> After Execution				Status Flags <sup>(4)</sup> Affected
			A	B	C	D	
ACOS	Inverse Cosine of A	0 6	R	U	U	U	S, Z, E
ASIN	Inverse Sine of A	0 5	R	U	U	U	S, Z, E
ATAN	Inverse Tangent of A	0 7	R	B	U	U	S, Z
CHSF	Sign Change of A	1 5	R	B	C	D	S, Z
COS	Cosine of A (radians)	0 3	R	B	U	U	S, Z
EXP	e <sup>A</sup> Function	0 A	R	B	U	U	S, Z, E
FADD	Add A and B	1 0	R	C	D	U	S, Z, E
FDIV	Divide B by A	1 3	R	C	D	U	S, Z, E
FLTD	32-Bit Integer to Floating Point Conversion	1 C	R	B	C	U	S, Z
FLTS	16-Bit Integer to Floating Point Conversion	1 D	R	B	C	U	S, Z
FMUL	Multiply A and B	1 2	R	C	D	U	S, Z, E
FSUB	Subtract A from B	1 1	R	C	D	U	S, Z, E
LOG	Common Logarithm (base 10) of A	0 8	R	B	U	U	S, Z, E
LN	Natural Logarithm of A	0 9	R	B	U	U	S, Z, E
POPF	Stack Pop	1 8	B	C	D	A	S, Z
PTOF	Stack Push	1 7	A	A	B	C	S, Z
PUPI	Push $\pi$ onto Stack	1 A	R	A	B	C	S, Z
PWR	B <sup>A</sup> Power Function	0 B	R	C	U	U	S, Z, E
SIN	Sine of A (radians)	0 2	R	B	U	U	S, Z
SQRT	Square Root of A	0 1	R	B	C	U	S, Z, E
TAN	Tangent of A (radians)	0 4	R	B	U	U	S, Z, E
XCHF	Exchange A and B	1 9	B	A	C	D	S, Z

**Table 3. 32-Bit Integer Instructions**

Instruction	Description	Hex <sup>(1)</sup> Code	Stack Contents <sup>(2)</sup> After Execution				Status Flags <sup>(4)</sup> Affected
			A	B	C	D	
CHSD	Sign Change of A	3 4	R	B	C	D	S, Z, O
DADD	Add A and B	2 C	R	C	D	A	S, Z, C, E
DDIV	Divide B by A	2 F	R	C	D	U	S, Z, E
DMUL	Multiply A and B (R = lower 32-bits)	2 E	R	C	D	U	S, Z, O
DMUO	Multiply A and B (R = upper 32-bits)	3 6	R	C	D	U	S, Z, O
DSUB	Subtract A from B	2 D	R	C	D	A	S, Z, C, O
FIXD	Floating Point to Integer Conversion	1 E	R	B	C	U	S, Z, O
POPD	Stack Pop	3 8	B	C	D	A	S, Z
PTOD	Stack Push	3 7	A	A	B	C	S, Z
XCHD	Exchange A and B	3 9	B	A	C	D	S, Z

**Table 4. 16-Bit Integer Instructions**

Instruction	Description	Hex <sup>(1)</sup> Code	Stack Contents <sup>(3)</sup> After Execution								Status Flags <sup>(4)</sup> Affected
			A <sub>U</sub>	A <sub>L</sub>	B <sub>U</sub>	B <sub>L</sub>	C <sub>U</sub>	C <sub>L</sub>	D <sub>U</sub>	D <sub>L</sub>	
CHSS	Change Sign of A <sub>U</sub>	7 4	R	A <sub>L</sub>	B <sub>U</sub>	B <sub>L</sub>	C <sub>U</sub>	C <sub>L</sub>	D <sub>U</sub>	D <sub>L</sub>	S, Z, O
FIXS	Floating Point to Integer Conversion	1 F	R	B <sub>U</sub>	B <sub>L</sub>	C <sub>U</sub>	C <sub>L</sub>	U	U	U	S, Z, O
POPS	Stack Pop	7 8	A <sub>L</sub>	B <sub>U</sub>	B <sub>L</sub>	C <sub>U</sub>	C <sub>L</sub>	D <sub>U</sub>	D <sub>L</sub>	A <sub>U</sub>	S, Z
PTOS	Stack Push	7 7	A <sub>U</sub>	A <sub>L</sub>	B <sub>U</sub>	B <sub>L</sub>	C <sub>U</sub>	C <sub>L</sub>	D <sub>U</sub>	D <sub>L</sub>	S, Z
SADD	Add A <sub>U</sub> and A <sub>L</sub>	6 C	R	B <sub>U</sub>	B <sub>L</sub>	C <sub>U</sub>	C <sub>L</sub>	D <sub>U</sub>	D <sub>L</sub>	A <sub>U</sub>	S, Z, C, E
SDIV	Divide A <sub>L</sub> by A <sub>U</sub>	6 F	R	B <sub>U</sub>	B <sub>L</sub>	C <sub>U</sub>	C <sub>L</sub>	D <sub>U</sub>	D <sub>L</sub>	U	S, Z, E
SMUL	Multiply A <sub>L</sub> by A <sub>U</sub> (R = lower 16-bits)	6 E	R	B <sub>U</sub>	B <sub>L</sub>	C <sub>U</sub>	C <sub>L</sub>	D <sub>U</sub>	D <sub>L</sub>	U	S, Z, E
SMUO	Multiply A <sub>L</sub> by A <sub>U</sub> (R = upper 16-bits)	7 6	R	B <sub>U</sub>	B <sub>L</sub>	C <sub>U</sub>	C <sub>L</sub>	D <sub>U</sub>	D <sub>L</sub>	U	S, Z, E
SSUB	Subtract A <sub>U</sub> from A <sub>L</sub>	6 D	R	B <sub>U</sub>	B <sub>L</sub>	C <sub>U</sub>	C <sub>L</sub>	D <sub>U</sub>	D <sub>L</sub>	A <sub>U</sub>	S, Z, C, E
XCHS	Exchange A <sub>U</sub> and A <sub>L</sub>	7 9	A <sub>L</sub>	A <sub>U</sub>	B <sub>U</sub>	B <sub>L</sub>	C <sub>U</sub>	C <sub>L</sub>	D <sub>U</sub>	D <sub>L</sub>	S, Z
NOP	No Operation	0 0	A <sub>U</sub>	A <sub>L</sub>	B <sub>U</sub>	B <sub>L</sub>	C <sub>U</sub>	C <sub>L</sub>	D <sub>U</sub>	D <sub>L</sub>	

**Notes:** 1. In the hex code column, SVREQ is a 0.

2. The stack initially is composed of four 32-bit numbers (A, B, C, D). A is equivalent to Top Of Stack (TOS) and B is Next On Stack (NOS). Upon completion of a command the stack is composed of: the result (R); undefined (U); or the initial contents (A, B, C, or D).

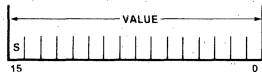
3. The stack initially is composed of eight 16-bit numbers (A<sub>U</sub>, A<sub>L</sub>, B<sub>U</sub>, B<sub>L</sub>, C<sub>U</sub>, C<sub>L</sub>, D<sub>U</sub>, D<sub>L</sub>). A<sub>U</sub> is the TOS and A<sub>L</sub> is NOS. Upon completion of a command the stack is composed of: the result (R); undefined (U); or the initial contents (A<sub>U</sub>, A<sub>L</sub>, B<sub>U</sub>, B<sub>L</sub>, ...).

4. Nomenclature: Sign (S); Zero (Z); Overflow (O); Carry (C); Error Code Field (E).

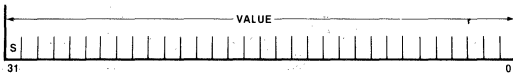
**DATA FORMATS**

The 8231A arithmetic processing unit handles operands in both fixed point and floating point formats. Fixed point operands may be represented in either single (16-bit operands) or double precision (32-bit operands), and are always represented as binary, two's complement values.

**SINGLE PRECISION FIXED POINT FORMAT**



**DOUBLE PRECISION FIXED POINT FORMAT**



The sign (positive or negative) of the operand is located in the most significant bit (MSB). Positive values are represented by a sign bit of zero (S = 0). Negative values are represented by the two's complement of the corresponding positive value with a sign bit equal to 1 (S = 1). The range of values that may be accommodated by each of these formats is -32,768 to +32,767 for single precision and -2,147,483,648 to +2,147,483,647 for double precision.

Floating point binary values are represented in a format that permits arithmetic to be performed in a fashion analogous to operations with decimal values expressed in scientific notation.

$$(5.83 \times 10^2) (8.16 \times 10^1) = (4.75728 \times 10^4)$$

In the decimal system, data may be expressed as values between 0 and 10 times 10 raised to a power that effectively shifts the implied decimal point right or left the number of places necessary to express the result in conventional form (e.g., 47,572.8). The value-portion of the data is called the mantissa. The exponent may be either negative or positive.

The concept of floating point notation has both a gain and a loss associated with it. The gain is the ability to represent the significant digits of data with values spanning a large dynamic range limited only by the capacity of the exponent field. For example, in decimal notation if the exponent field is two digits wide, and the mantissa is five digits, a range of values (positive or negative) from  $1.0000 \times 10^{-99}$  to  $9.9999 \times 10^{+99}$  can be accommodated. The loss is that only the significant digits of the value can be represented. Thus there is no distinction in this representation between the values 123451 and 123452, for example, since each would be expressed as:  $1.2345 \times 10^5$ . The sixth digit has been discarded. In most applications where the dynamic range of values to be represented is large, the loss of significance, and hence accuracy of results, is a minor consideration. For greater precision a fixed point format could be chosen, although with a loss of potential dynamic range.

The 8231A is a binary arithmetic processor and requires that floating point data be represented by a fractional mantissa value between .5 and 1 multiplied by 2 raised to an appropriate power. This is expressed as follows:

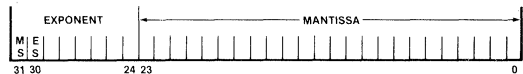
$$\text{value} = \text{mantissa} \times 2^{\text{exponent}}$$

For example, the value 100.5 expressed in this form is  $0.1100\ 1001 \times 2^7$ . The decimal equivalent of this value may be computed by summing the components (powers of two) of the mantissa and then multiplying by the exponent as shown below:

$$\begin{aligned} \text{value} &= (2^{-1} + 2^{-2} + 2^{-5} + 2^{-8}) \times 2^7 \\ &= 0.5 + 0.25 + 0.03125 + 0.00290625 \times 128 \\ &= 0.78515625 \times 128 \\ &= 100.5 \end{aligned}$$

**FLOATING POINT FORMAT**

The format for floating point values in the 8231A is given below. The mantissa is expressed as a 24-bit (fractional) value; the exponent is expressed as a two's complement 7-bit value having a range of -64 to +63. The most significant bit is the sign of the mantissa (0 = positive, 1 = negative), for a total of 32 bits. The binary point is assumed to be to the left of the most significant mantissa bit (bit 23). All floating point data values must be normalized. Bit 23 must be equal to 1, except for the value zero, which is represented by all zeros.

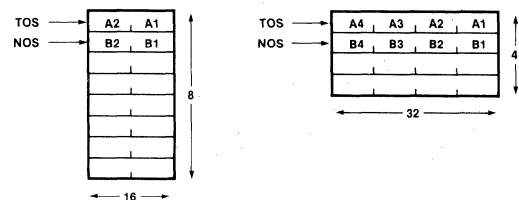


The range of values that can be represented in this format is  $\pm(2.7 \times 10^{-20}$  to  $9.2 \times 10^{18})$  and zero.

**FUNCTIONAL DESCRIPTION**

**STACK CONTROL**

The user interface to the 8231A includes access to an 8 level 16-bit wide data stack. Since single precision fixed point operands are 16-bits in length, eight such values may be maintained in the stack. When using double precision fixed point or floating point formats four values may be stored. The stack in these two configurations can be visualized as shown below:



Data are written onto the stack, eight bits at a time, in the order shown (A1, A2, A3, . . .). Data are removed from the stack in reverse byte order (A4, A3, A2 . . .). Data should be entered onto the stack in multiples of the number of bytes appropriate to the chosen data format.

**DATA ENTRY**

Data entry is accomplished by bringing the chip select ( $\overline{CS}$ ), the command/data line ( $A_0$ ), and  $\overline{WR}$  low, as shown in the timing diagram. The entry of each new data word "pushes down" the previously entered data and places the new byte on the top of stack (TOS). Data on the bottom of the stack prior to a stack entry are lost.

**DATA REMOVAL**

Data are removed from the stack in the 8231A by bringing chip select ( $\overline{CS}$ ), command/data ( $A_0$ ), and  $\overline{RD}$  low as shown in the timing diagram. The removal of each data word redefines TOS so that the next successive byte to be removed becomes TOS. Data removed from the stack rotates to the bottom of the stack.

**COMMAND ENTRY**

After the appropriate number of bytes of data have been entered onto the stack, a command may be issued to perform an operation on that data. Commands which require two operands for execution (e.g., add) operate on the TOS and NOS values. Single operand commands operate only on the TOS.

Commands are issued to the 8231A by bringing the chip select ( $\overline{CS}$ ) line low, command data ( $A_0$ ) line high, and  $\overline{WR}$  line low as indicated by the timing diagram. After a command is issued, the CPU can continue execution of its program concurrently with the 8231A command execution.

**COMMAND COMPLETION**

The 8231A signals the completion of each command execution by lowering the End Execution line ( $\overline{END}$ ). Simultaneously, the busy bit in the status register is cleared and the Service Request bit of the command register is checked. If it is a "1" the service request output level (SVREQ) is raised.  $\overline{END}$  is cleared on receipt of an active low End Acknowledge ( $\overline{EACK}$ ) pulse. Similarly, the service request line is cleared by recognition of an active low Service Acknowledge ( $\overline{SVACK}$ ) pulse.

**READY OPERATION**

An active high ready (READY) is provided. This line is high in its quiescent state and is pulled low by the 8231A under the following conditions:

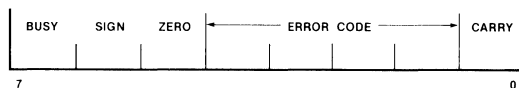
1. A previously initiated operation is in progress (device busy) and Command Entry has been attempted. In this case, the READY line will be pulled low and remain low until completion of the current command execution. It will then go high, permitting entry of the new command.
2. A previously initiated operation is in progress and stack access has been attempted. In this case, the READY line will be pulled low, will remain in that state until execution is complete, and will then be raised to permit completion of the stack access.
3. The 8231A is not busy, and data removal has been requested. READY will be pulled low for the length of time necessary to transfer the byte from the top of stack to the interface latch, and will then go high, indicating availability of the data.

4. The 8231A is not busy, and a data entry has been requested. READY will be pulled low for the length of time required to ascertain if the preceding data byte, if any, has been written to the stack. If so READY will immediately go high. If not, READY will remain low until the interface latch is free and will then go high.
5. When a status read has been requested, READY will be pulled low for the length of time necessary to transfer the status to the interface latch, and will then be raised to permit completion of the status read. Status may be read whether or not the 8231A is busy.

When READY goes low, the APU expects the bus control signals present at the time to remain stable until READY goes high.

**DEVICE STATUS**

Device status is provided by means of an internal status register whose format is shown below:



**BUSY:** Indicates that 8231A is currently executing a command (1=Busy)

**SIGN:** Indicates that the value on the top of stack is negative (1 = Negative)

**ZERO:** Indicates that the value on the top of stack is zero (1 = Value is zero)

**ERROR CODE:** This field contains an indication of the validity of the result of the last operation. The error codes are:

- 0000 — No error
- 1000 — Divide by zero
- 0100 — Square root or log of negative number
- 1100 — Argument of inverse sine, cosine, or  $e^x$  too large
- XX10 — Underflow
- XX01 — Overflow

**CARRY:** Previous operation resulted in carry or borrow from most significant bit. (1 = Carry/Borrow, 0 = No Carry/No Borrow.)

If the BUSY bit in the status register is a one, the other status bits are not defined; if zero, indicating not busy, the operation is complete and the other status bits are defined as given above.

**READ STATUS**

The 8231A status register can be read by the CPU at any time (whether an operation is in progress or not) by bringing the chip select ( $\overline{CS}$ ) low, the command/data line ( $A_0$ ) high, and lowering  $\overline{RD}$ . The status register is then gated onto the data bus and may be input by the CPU.

**EXECUTION TIMES**

Timing for execution of the 8231A command set is contained below. All times are given in terms of clock cycles. Where substantial variation of execution times

is possible, the minimum and maximum values are quoted; otherwise, typical values are given. Variations are data dependent.

Total execution times may require allowances for operand transfer into the APU, command execution, and result retrieval from the APU. Except for command exe-

cutation, these times will be heavily influenced by the nature of the data, the control interface used, the speed of memory, the CPU used, the priority allotted to DMA and Interrupt operations, the size and number of operands to be transferred, and the use of chained calculations, etc.

**Table 5. Command Execution Times**

Command Mnemonic	Clock Cycles	Command Mnemonic	Clock Cycles	Command Mnemonic	Clock Cycles	Command Mnemonic	Clock Cycles
SADD	17	FADD	54-368	LN	4298-6956	POPF	12
SSUB	30	FSUB	70-370	EXP	3794-4878	XCHS	18
SMUL	84-94	FMUL	146-168	PWR	8290-12032	XCHD	26
SMUU	80-98						
SDIV	84-94	FDIV	154-184	NOP	4	XCHF	26
DADD	21	SORT	800	CHSS	23	PUPI	16
DSUB	38	SIN	4464	CHSD	27		
DMUL	194-210	COS	4118	CHSF	18		
DMUU	182-218						
DDIV	208	TAN	5754	PTOS	16		
FIXS	92-216	ASIN	7668	PTOD	20		
FIXD	100-346	ACOS	7734	PTOF	20		
FLTS	98-186	ATAN	6006	POPS	10		
FLTD	98-378	LOG	4474-7132	POPD	12		

## DERIVED FUNCTION DISCUSSION

Computer approximations of transcendental functions are often based on some form of polynomial equation, such as:

$$F(X) = A_0 + A_1X + A_2X^2 + A_3X^3 + A_4X^4 \dots \quad (1-1)$$

The primary shortcoming of an approximation in this form is that it typically exhibits very large errors when the magnitude of  $|X|$  is large, although the errors are small when  $|X|$  is small. With polynomials in this form, the error distribution is markedly uneven over any arbitrary interval.

A set of approximating functions exists that not only minimizes the maximum error but also provides an even distribution of errors within the selected data representation interval. These are known as Chebyshev Polynomials and are based upon cosine functions. These functions are defined as follows:

$$T_n(X) = \cos n\theta; \text{ where } n = 0, 1, 2, \dots \quad (1-2)$$

$$\theta = \cos^{-1}X$$

The various terms of the Chebyshev series can be computed as shown below:

$$T_0(X) = \cos(0 \cdot \theta) = \cos(0) = 1 \quad (1-4)$$

$$T_1(X) = \cos(\cos^{-1}X) = X \quad (1-5)$$

$$T_2(X) = \cos 2\theta = 2\cos^2\theta - 1 = 2\cos^2(\cos^{-1}X) - 1 = 2X^2 - 1 \quad (1-6)$$

In general, the next term in the Chebyshev series can be recursively derived from the previous term as follows:

$$T_{n+1}(X) = 2X[T_n(X) - T_{n-1}(X)]; n \geq 2 \quad (1-7)$$

Common logarithms are computed by multiplication of the natural logarithm by the conversion factor 0.43429448 and the error function is therefore the same as that for natural logarithm. The power function is realized by combination of natural log and exponential functions according to the equation:

$$X^Y = e^{Y \ln X}$$

The error for the power function is a combination of that for the logarithm and exponential functions.

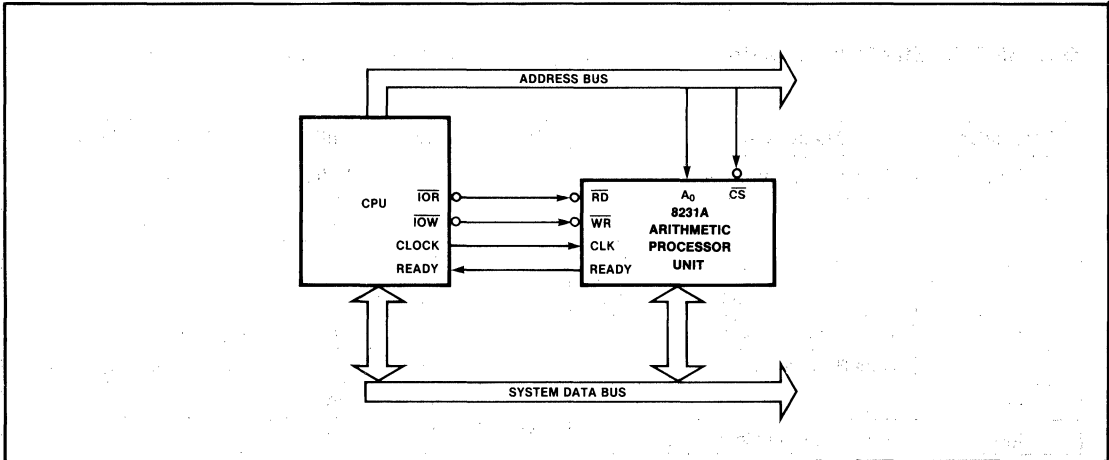
Each of the derived functions is an approximation of the true function. Thus the result of a derived function will have an error. The absolute error is the difference between the function's result and the true result. A more useful measure of the function's error is relative error (absolute error/true result). This gives a measurement of the significant digits of algorithm accuracy. For the derived functions except LN, LOG, and PWR the relative error is typically  $4 \times 10^{-7}$ . For PWR the relative error is the summation of the EXP and LN errors,  $7 \times 10^{-7}$ . For LN and LOG, the absolute error is  $2 \times 10^{-7}$ .

**APPLICATION INFORMATION**

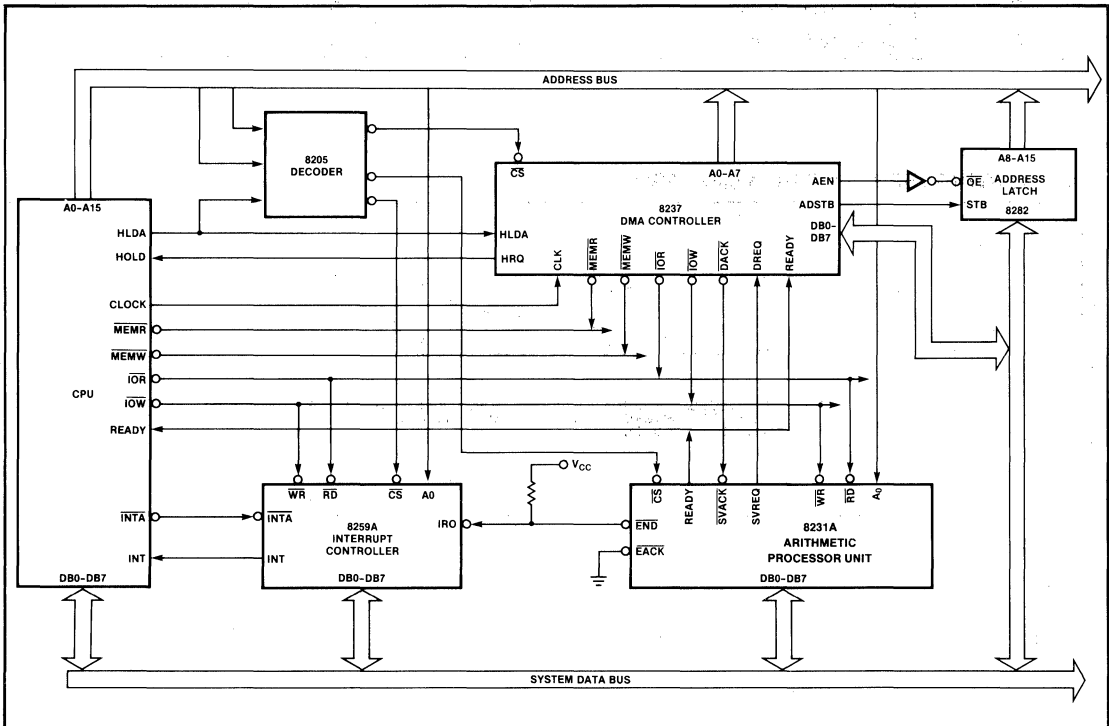
The diagram in Figure 4 shows the interface connections for the APU with operand transfers handled by an 8237 DMA controller, and CPU coordination handled by an Interrupt Controller. The APU interrupts the CPU to indicate that a command has been completed. When the performance enhancements provided by the DMA and Interrupt operations are not required, the APU interface

can be simplified as shown in Figure 3. The 8231A APU is designed with a general purpose 8-bit data bus and interface control so that it can be conveniently used with any general 8-bit processor.

In many systems it will be convenient to use the microcomputer system clock to drive the APU clock input. In the case of 8080A systems it would be the  $\phi$ 2TTL signal. Its cycle time will usually fall in the range of 250 ns to 1000 ns, depending on the system speed.



**Figure 3. Minimum Configuration Example**



**Figure 4. High Performance Configuration Example**



**ABSOLUTE MAXIMUM RATINGS\***

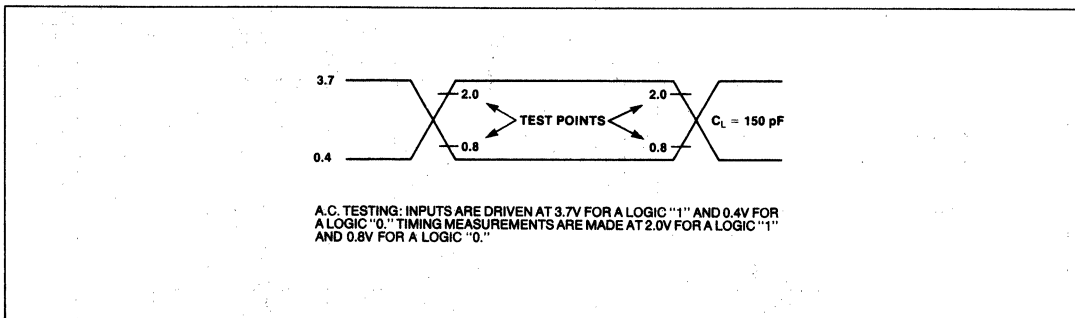
Storage Temperature..... - 65°C to + 150°C  
 Ambient Temperature Under Bias..... 0°C to 70°C  
 $V_{DD}$  with Respect to  $V_{SS}$ ..... - 0.5V to + 15.0V  
 $V_{CC}$  with Respect to  $V_{SS}$ ..... - 0.5V to + 7.0V  
 All Signal Voltages with Respect  
 to  $V_{SS}$ ..... - 0.5V to + 7.0V  
 Power Dissipation..... 2.0W

*\*NOTICE: Stresses above those listed under "Absolute Maximum Ratings" may cause permanent damage to the device. This is a stress rating only and functional operation of the device at these or any other conditions above those indicated in the operational sections of this specification is not implied. Exposure to absolute maximum rating conditions for extended periods may effect device reliability.*

**D.C. AND OPERATING CHARACTERISTICS** ( $T_A = 0^\circ\text{C}$  to  $70^\circ\text{C}$ ,  $V_{SS} = 0\text{V}$ ,  $V_{CC} = +5\text{V} \pm 10\%$ ,  $V_{DD} = +12\text{V} \pm 10\%$ )

Parameters	Description	Min.	Typ.	Max.	Units	Test Conditions
$V_{OH}$	Output HIGH Voltage	3.7			Volts	$I_{OH} = -200 \mu\text{A}$
$V_{OL}$	Output LOW Voltage			0.4	Volts	$I_{OL} = 3.2 \text{ mA}$
$V_{IH}$	Input HIGH Voltage	2.0		$V_{CC}$	Volts	
$V_{IL}$	Input LOW Voltage	- 0.5		0.8	Volts	
$I_{IL}$	Input Load Current			$\pm 10$	$\mu\text{A}$	$V_{SS} \leq V_{IN} \leq V_{CC}$
$I_{OZ}$	Data Bus Leakage			$\pm 10$	$\mu\text{A}$	$V_{SS} + 0.4 \leq V_{OUT} \leq V_{CC}$
$I_{CC}$	$V_{CC}$ Supply Current		50	95	mA	
$I_{DD}$	$V_{DD}$ Supply Current		50	95	mA	
$C_O$	Output Capacitance		8		pF	fc = 1.0 MHz, Inputs = 0V
$C_I$	Input Capacitance		5		pF	
$C_{IO}$	I/O Capacitance		10		pF	

**A.C. TESTING INPUT, OUTPUT WAVEFORM**



**A.C. CHARACTERISTICS** ( $T_A = 0^\circ\text{C}$  to  $70^\circ\text{C}$ ,  $V_{SS} = 0\text{V}$ ,  $V_{CC} = +5\text{V} \pm 10\%$ ,  $V_{DD} = +12\text{V} \pm 10\%$ )

**READ OPERATION**

Symbol	Parameter	8231A-8		8231A-3		8231A		Units
		Min.	Max.	Min.	Max.	Min.	Max.	
$t_{AR}$	$A_0$ , $\overline{CS}$ Setup to $\overline{RD}$	0		0		0		ns
$t_{RA}$	$A_0$ , $\overline{CS}$ Hold from $\overline{RD}$	0		0		0		ns
$t_{RY}$	READY $\downarrow$ from $\overline{RD}$ $\downarrow$ Delay (Note 2)		150		100		100	ns
$t_{YR}$	READY $\uparrow$ to $\overline{RD}$ $\uparrow$	0		0		0		ns
$t_{RRR}$	READY Pulse Width (Note 3)	Data	$3.5 t_{CY} + 50$		$3.5 t_{CY} + 50$		$3.5 t_{CY} + 50$	ns
		Status	$1.5 t_{CY} + 50$		$1.5 t_{CY} + 50$		$1.5 t_{CY} + 50$	ns
$t_{RDE}$	Data Bus Enable from $\overline{RD}$ $\downarrow$	50		50		50		ns
$t_{DRY}$	Data Valid to READY $\uparrow$	0		0		0		ns
$t_{DF}$	Data Float after $\overline{RD}$ $\uparrow$	50	200	50	150	50	100	ns

**WRITE OPERATION**

Symbol	Parameter	8231A-8		8231A-3		8231A		Units
		Min.	Max.	Min.	Max.	Min.	Max.	
$t_{AW}$	$A_0$ , $\overline{CS}$ Setup to $\overline{WR}$	0		0		0		ns
$t_{WA}$	$A_0$ , $\overline{CS}$ Hold after $\overline{WR}$	60		30		25		ns
$t_{WY}$	READY $\downarrow$ from $\overline{WR}$ $\downarrow$ Delay (Note 2)		150		100		100	ns
$t_{YW}$	READY $\uparrow$ to $\overline{WR}$ $\uparrow$	0		0		0		ns
$t_{RRW}$	READY Pulse Width (Note 4)		50		50		50	ns
$t_{WI}$	Write Inactive Time (Note 4)	Command	$4 t_{CY}$		$4 t_{CY}$		$4 t_{CY}$	ns
		Data	$5 t_{CY}$		$5 t_{CY}$		$5 t_{CY}$	ns
$t_{DW}$	Data Setup to $\overline{WR}$	150		100		100		ns
$t_{WD}$	Data Hold after $\overline{WR}$	20		20		20		ns

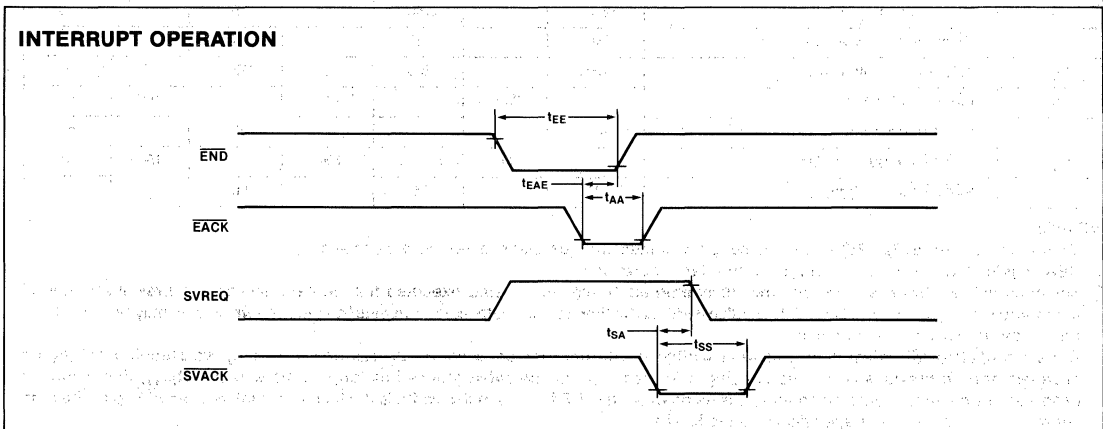
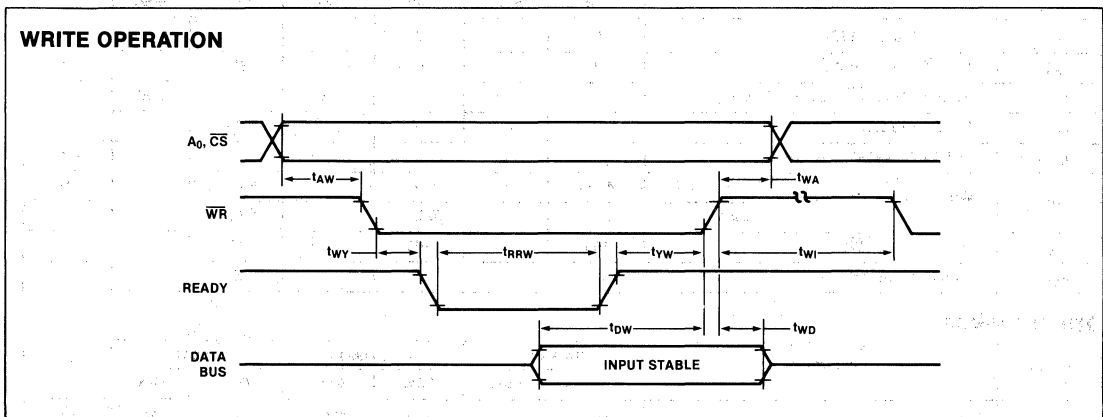
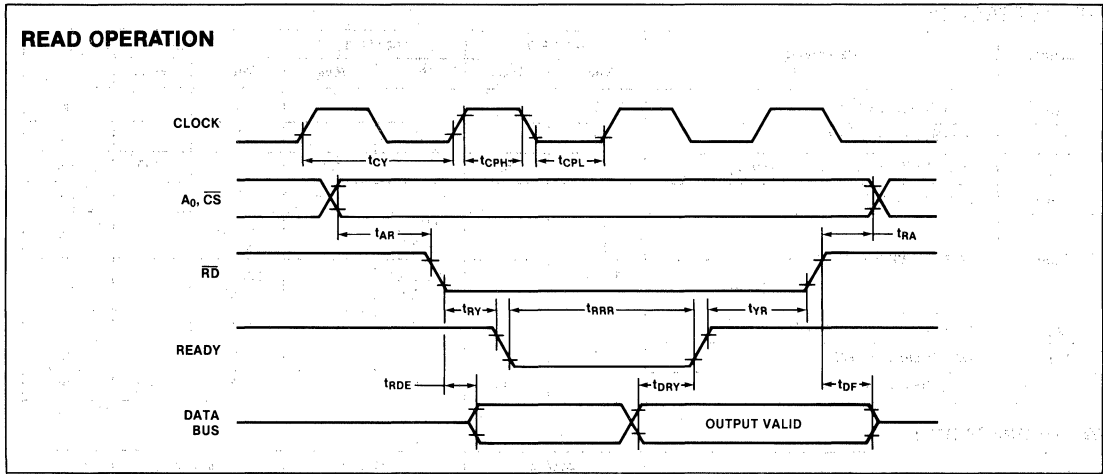
**OTHER TIMINGS**

Symbol	Parameter	8231A-8		8231A-3		8231A		Units
		Min.	Max.	Min.	Max.	Min.	Max.	
$t_{CY}$	Clock Period	480	5000	320	3300	250	2500	ns
$t_{CPH}$	Clock Pulse High Width	200		140		100		ns
$t_{CPL}$	Clock Pulse Low Width	240		160		120		ns
$t_{EE}$	$\overline{END}$ Pulse Width (Note 5)	400		300		200		ns
$t_{EAE}$	$\overline{EACK}$ $\downarrow$ to $\overline{END}$ $\uparrow$ Delay		200		175		150	ns
$t_{AA}$	$\overline{EACK}$ Pulse Width	100		75		50		ns
$t_{SA}$	$\overline{SVACK}$ $\downarrow$ to $\overline{SVREQ}$ $\downarrow$ Delay		300		200		150	ns
$t_{SS}$	$\overline{SVACK}$ Pulse Width	100		75		50		ns

**NOTES:**

1. Typical values are for  $T_A = 25^\circ\text{C}$ , nominal supply voltages and nominal processing parameters.
2. READY is pulled low for both command and data operations.
3. Minimum values shown assume no previously entered command is being executed for the data access. If a previously entered command is being executed, READY low pulse width is the time to complete execution plus the time shown. Status may be read at any time without exceeding the time shown.
4. READY low pulse width is less than 50 ns when writing into the data port or the control port as long as the duty cycle requirement ( $t_{WI}$ ) is observed and no previous command is being executed.  $t_{WI}$  may be safely violated as long as the extended  $t_{RRW}$  that results is observed. If a previously entered command is being executed, READY low pulse width is the time to complete execution plus the time shown. These timings refer specifically to the 8231A.
5.  $\overline{END}$  low pulse width is specified for  $\overline{EACK}$  tied to VSS. Otherwise  $t_{EAE}$  applies.

WAVEFORMS



# 8232 FLOATING POINT PROCESSING UNIT

- Compatible with Proposed IEEE Format and Existing Intel Floating Point Standard
- Single (32-Bit) and Double (64-Bit) Precision Capability
- Add, Subtract, Multiply and Divide Functions
- Stack Oriented Operand Storage
- General Purpose 8-Bit Data Bus Interface
- Standard 24-Pin Package
- 12V and 5V Power Supplies
- Compatible with MCS-80™, MCS-85™ and MCS-86™ Microprocessor Families
- Error Interrupt
- Direct Memory Access or Programmed I/O Data Transfers
- End of Execution Signal
- Advanced N-Channel Silicon Gate HMOS Technology

The Intel® 8232 is a high performance floating-point processor unit (FPU). It provides single precision (32-bit) and double precision (64-bit) add, subtract, multiply and divide operations. The 8232's floating point arithmetic is a subset of the proposed IEEE standard. It can be easily interfaced to enhance the computational capabilities of the host microprocessor.

The operand, result, status and command information transfers take place over an 8-bit bidirectional data bus. Operands are pushed onto an internal stack by the host processor and a command is issued to perform an operation on the data stack. The results of the operation are available to the host processor from the stack.

Information transfers between the 8232 and the host processor can be handled by using programmed I/O or direct memory access techniques. After completing an operation, the 8232 activates an "end of execution" signal that can be used to interrupt the host processor.

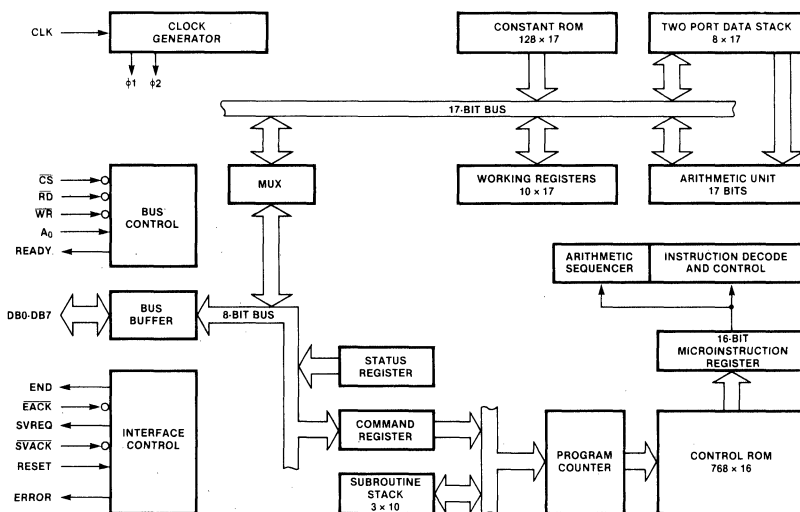


Figure 1. Block Diagram

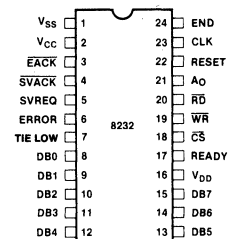


Figure 2. Pin Configuration

Table 1. Pin Description

Symbol	Pin No.	Type	Name and Description																				
V <sub>CC</sub>	2		<b>POWER SUPPLY:</b> +5V power supply																				
V <sub>DD</sub>	16		<b>POWER SUPPLY:</b> +12V power supply																				
V <sub>SS</sub>	1		<b>GROUND</b>																				
CLK	23	I	<b>CLOCK:</b> An external timing source connected to the CLK input provides the necessary clocking.																				
RESET	22	I	<b>RESET:</b> A HIGH level on this input causes initialization. Reset terminates any operation in progress, and clears the status register to zero. The internal stack pointer is initialized and the contents of the stack may be affected. After a reset the END output, the ERROR output and the SVREQ output will be LOW. For proper initialization, RESET must be HIGH for at least five CLK periods following stable power supply voltages and stable clock.																				
CS	18	I	<p><b>CHIP SELECT:</b> input must be LOW to accomplish any read or write operation to the 8232.</p> <p>To perform a write operation, appropriate data is presented on DB0 through DB7 lines, appropriate logic level on the A<sub>0</sub> input and the CS input is made LOW. Whenever WR and RD inputs are both HIGH and CS is LOW, READY goes LOW. However, actual writing into the 8232 cannot start until WR is made LOW. After initiating the write operation by the HIGH to LOW transition on the WR input, the READY output will go HIGH, indicating the write operation has been acknowledged. The WR input can go HIGH after READY goes HIGH. The data lines, the A<sub>0</sub> input and the CS input can change when appropriate hold time requirements are satisfied. See write timing diagram for details.</p> <p>To perform a read operation an appropriate logic level is established on the A<sub>0</sub> input and CS is made LOW. The READY output goes LOW because WR and RD inputs are HIGH. The read operation does not start until the RD input goes LOW. READY will go HIGH indicating that read operation is complete and the required information is available on the DB0 through DB7 lines. This information will remain on the data lines as long as RD is LOW. The RD input can return HIGH anytime after READY goes HIGH. The CS input and A<sub>0</sub> input can change anytime after RD returns HIGH. See read timing diagram for details. If the CS is tied LOW permanently, READY will remain LOW until the next 8232 read or write access.</p>																				
A <sub>0</sub>	21	I	<p><b>ADDRESS:</b> The A<sub>0</sub> input together with the RD and WR inputs determines the type of transfer to be performed on the data bus as follows:</p> <table border="1" style="margin-left: 20px;"> <thead> <tr> <th>A<sub>0</sub></th> <th>RD</th> <th>WR</th> <th>Function</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>1</td> <td>0</td> <td>Enter data byte into stack</td> </tr> <tr> <td>0</td> <td>0</td> <td>1</td> <td>Read data byte from stack</td> </tr> <tr> <td>1</td> <td>1</td> <td>0</td> <td>Enter command</td> </tr> <tr> <td>1</td> <td>0</td> <td>1</td> <td>Read status</td> </tr> </tbody> </table>	A <sub>0</sub>	RD	WR	Function	0	1	0	Enter data byte into stack	0	0	1	Read data byte from stack	1	1	0	Enter command	1	0	1	Read status
A <sub>0</sub>	RD	WR	Function																				
0	1	0	Enter data byte into stack																				
0	0	1	Read data byte from stack																				
1	1	0	Enter command																				
1	0	1	Read status																				

Symbol	Pin No.	Type	Name and Description
RD	20	I	<p><b>READ:</b> A LOW level on this input is used to read information from an internal location and gate that information onto the data bus. The CS input must be LOW to accomplish the read operation. The A<sub>0</sub> input determines what internal location is to be read. See A<sub>0</sub>, CS input descriptions and read timing diagram for details. If the END output was HIGH, performing any read operation will make the END output go LOW after the HIGH to LOW transition of the RD input (assuming CS is LOW). If the ERROR output was HIGH, performing a status register read operation will make the ERROR output LOW. This will happen after the HIGH to LOW transition of the RD input (assuming CS is LOW).</p>
WR	19	I	<p><b>WRITE:</b> A LOW level on this input is used to transfer information from the data bus into an internal location. The CS must be LOW to accomplish the write operation. A<sub>0</sub> determines which internal location is to be written. See A<sub>0</sub>, CS input descriptions and write timing diagram for details.</p> <p>If the END output was HIGH, performing any write operation will make the END output go LOW after the LOW to HIGH transition of the WR input (assuming CS is LOW).</p>
EACK	3	I	<p><b>END ACKNOWLEDGE:</b> When LOW, makes the END output go LOW. As mentioned earlier, HIGH on the END output signals completion of a command execution. The END signal is derived from an internal flip-flop which is clocked at the completion of a command. This flip-flop is clocked to the reset state when EACK is LOW. Consequently, if EACK is tied LOW, the END output will be a pulse that is approximately one CLK period wide.</p>
SVACK	4	I	<p><b>SERVICE ACKNOWLEDGE:</b> A LOW level on this input clears SVREQ. If the SVACK input is permanently tied LOW, it will conflict with the internal setting of the SVREQ output. Thus, the SVREQ indication cannot be relied upon if the SVACK is tied LOW.</p>
END	24	O	<p><b>END OF EXECUTION:</b> A HIGH on this output indicates that execution of the current command is complete. This output will be cleared LOW by activating the EACK input LOW or performing any read or write operation or device initialization using RESET. If EACK is tied LOW, the END output will be a pulse (see EACK description).</p> <p>Reading the status register while a command execution is in progress is allowed. However, any read or write operation clears the flip-flop that generates the END output. Thus, such continuous reading could conflict with internal logic setting of the END flip-flop at the end of command execution.</p>

Table 1. Pin Description (Continued)

Symbol	Pin No.	Type	Name and Description
SVREQ	5	O	<b>SERVICE REQUEST:</b> A HIGH on this output indicates completion of a command. In this sense this output is the same as the END output. However, the SVREQ output will go HIGH at the completion of a command only when the Service Request Enable bit was set to 1. The SVREQ can be cleared (i.e., go LOW) by activating the $\overline{SVACK}$ input LOW or initializing the device using the RESET. Also, the SVREQ will be automatically cleared after completion of any command that has the service request bit as 0.
ERROR	6	O	<b>ERROR:</b> Output goes HIGH to indicate that the current command execution resulted in an error condition. The error conditions are: attempt to divide by zero, exponent overflow and exponent underflow. The ERROR output is cleared LOW on a status register read operation or upon RESET.  The ERROR output is derived from the error bits in the status register. These error bits will be updated internally at an appropriate time during a command execution. Thus, ERROR output going HIGH may not coincide with the completion of a command. Reading of the status register can be performed while a command execution is in progress. However, it should be noted that reading the status register clears the ERROR output. Thus, reading the status register while a command execution is in progress may result in an internal conflict with the ERROR output.

Symbol	Pin No.	Type	Name and Description
READY	17	O	<b>READY:</b> Output is a handshake signal used while performing read or write transactions with the 8232. If the $\overline{WR}$ and $\overline{RD}$ inputs are both HIGH, the READY output goes LOW with the $\overline{CS}$ input in anticipation of a transaction. If $\overline{WR}$ goes LOW to initiate a write transaction with proper signals established on the DB0-DB7. $A_0$ inputs, the READY will return HIGH indicating that the write operation has been accomplished. The $\overline{WR}$ can be made HIGH after this event. On the other hand, if a read operation is desired, the $\overline{RD}$ input is made LOW after activating $\overline{CS}$ LOW and establishing proper $A_0$ input. (The READY will go LOW in response to $\overline{CS}$ going LOW.) The READY will return HIGH, indicating completion of read. The $\overline{RD}$ can return HIGH after this event. It should be noted that a read or write operation can be initiated without any regard to whether a command execution is in progress or not. Proper device operation is assured by obeying the READY output indication as described.
DB0-DB7	8-15	I/O	<b>DATA BUS:</b> Bidirectional lines are used to transfer command, status and operand information between the device and the host processor. DB0 is the least significant and DB7 is the most significant bit position. HIGH on a data bus line corresponds to 1 and LOW corresponds to 0.  When pushing operands on the stack using the data bus, the least significant byte must be pushed first and the most significant byte last. When popping the stack to read the result of an operation, the most significant byte will be available on the data bus first and the least significant byte will be the last. Moreover, for pushing operands and popping results, the number of transactions must be equal to the proper number of bytes appropriate for the chosen format. Otherwise, the internal byte pointer will not be aligned properly. The single precision format requires 4 bytes and double precision format requires 8 bytes.

**FUNCTIONAL DESCRIPTION**

Major functional units of the 8232 are shown in the block diagram. The 8232 employs a microprogram controlled stack oriented architecture with 17-bit wide data paths.

The Arithmetic Unit receives one of its operands from the Operand Stack. This stack is an eight word by 17-bit two port memory with last in-first out (LIFO) attributes. The second operand to the Arithmetic Unit is supplied by the internal 17-bit bus. In addition to supplying the second operand, this bidirectional bus also carries the results from the output of the Arithmetic Unit when required. Writing into the Operand Stack takes place

from this internal 17-bit bus when required. Also connected to this bus are the Constant ROM and Working Registers. The ROM provides the required constants to perform the mathematical operations while the Working Registers provide storage for the intermediate values during command execution.

Communication between the external world and the 8232 takes place on eight bidirectional input/output lines, DB0 through DB7 (Data Bus). These signals are gated to the internal 8-bit bus through appropriate interface and buffer circuitry. Multiplexing facilities exist for bidirectional communication between the internal eight

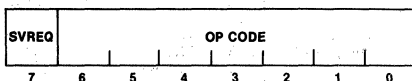
and 17-bit buses. The Status Register and Command Register are also located on the 8-bit bus.

The 8232 operations are controlled by the microprogram contained in the Control ROM. The Program Counter supplies the microprogram addresses and can be partially loaded from the Command Register. Associated with the Program Counter is the Subroutine Stack where return addresses are held during subroutine calls in the microprogram. The Microinstruction Register holds the current microinstruction being executed. The register facilitates pipelined microprogram execution. The Instruction Decode logic generates various internal control signals needed for the 8232 operation.

The Interface Control logic receives several external inputs and provides handshake related outputs to facilitate interfacing the 8232 to microprocessors.

### Command Format

The operation of the 8232 is controlled from the host processor by issuing instructions called commands. The command format is shown below.



The command consists of 8 bits; the least significant 7 bits specify the operation to be performed as detailed in Table 1. The most significant bit is the Service Request Enable bit. This bit must be a 1 if SVREQ is to go HIGH at the end of executing a command.

The commands fall into three categories: single precision arithmetic, double precision arithmetic and data manipulation. There are four arithmetic operations that can be performed with single precision (32-bit) or double precision (64-bit) floating-point numbers: add, subtract, multiply and divide. These operations require two operands. The 8232 assumes that these operands are located in the internal stack as Top of Stack (TOS) and Next on Stack (NOS). The result will always be returned to the previous NOS which becomes the new TOS. Results from an operation are of the same precision and format as the operands. The results will be rounded to preserve the accuracy. The actual data formats and rounding procedures are described in a later section. In addition to the arithmetic operations, the 8232 implements eight data manipulating operations. These include changing the sign of a double or single precision operand located in TOS, exchanging single precision operands located at TOS and NOS, as well as pushing and popping single or double precision operands. See also the sections on status register and operand formats.

The execution times of the commands are all data dependent. Table 3 shows one example of each command execution time.

### Operand Entry

The 8232 commands operate on the operands located at the TOS and NOS. Results are returned to the stack at NOS and then popped to TOS. The operands required for the 8232 are one of two formats — single precision floating-point (4 bytes) or double precision floating-point (8 bytes). The result of an operation has the same format as the operands. In other words, operations using single precision quantities always result in a single precision result, while operations involving double precision quantities will result in double precision result.

Operands are always entered into the stack least significant byte first and most significant byte last. The following procedure must be followed to enter operands into the stack:

1. The lower significant operand byte is established on the DB0-DB7 lines.
2. A LOW is established on the A<sub>0</sub> input to specify that data is to be entered into the stack.
3. The  $\overline{CS}$  input is made LOW. Whenever the  $\overline{WR}$  and  $\overline{RD}$  inputs are HIGH, the READY output will follow the  $\overline{CS}$  input. Thus, READY output will become LOW.
4. After appropriate set up time (see timing diagrams), the  $\overline{WR}$  input is made LOW.
5. Sometime after this event, READY will return HIGH to indicate that the write operation has been acknowledged.
6. Any time after the READY output goes HIGH, the  $\overline{WR}$  input can be made HIGH. The DB0-DB7, A<sub>0</sub> and  $\overline{CS}$  inputs can change after appropriate hold time requirements are satisfied (see timing diagrams).

The above procedure must be repeated until all bytes of the operand are pushed into the stack. It should be noted that for single precision operands 4 bytes should be pushed and 8 bytes must be pushed for double precision. Not pushing all the bytes of a quantity will result in byte pointer misalignment.

The 8232 stack can accommodate four single precision quantities or two double precision quantities. Pushing more quantities than the capacity of the stack will result in loss of data which is usual with any LIFO stack.

The stack can be visualized as shown below:

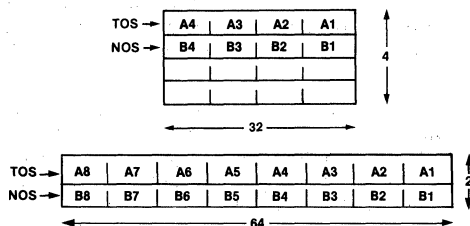


Table 2. 8232 Command Set

Single Precision Instructions

Instruction	Description	Hex <sup>1</sup> Code	Stack Contents <sup>2</sup> After Execution				Status Flags Affected <sup>4</sup>
			A	B	C	D	
SADD	Add A and B	01	R	C	D	U	S, Z, U, V
SSUB	Subtract A from B	02	R	C	D	U	S, Z, U, V
SMUL	Multiply A by B	03	R	C	D	U	S, Z, U, V
SDIV	Divide B by A. If A exponent = 0, then R = B.	04	R	C	D	U	S, Z, U, V, D
CHSS	Change sign of A <sup>5</sup>	05	R	B	C	D	S, Z
PTOS	Push stack <sup>5</sup>	06	A*	A	B	C	S, Z
POPS	Pop stack	07	B	C	D	A	S, Z
XCHS	Exchange	08	B	A	C	D	S, Z

Double Precision Instructions

Instruction	Description	Hex <sup>1</sup> Code	Stack Contents <sup>3</sup> After Execution		Status Flags Affected <sup>4</sup>
			A	B	
DADD	Add A and B	29	R	U	S, Z, U, V
DSUB	Subtract A from B	2A	R	U	S, Z, U, V
DMUL	Multiply A by B	2B	R	U	S, Z, U, V
DDIV	Divide B by A. If A = 0, then R = B.	2C	R	U	S, Z, U, V, D
CHSD	Change sign of A <sup>5</sup>	2D	R	B	S, Z
PTOD	Push stack <sup>5</sup>	2E	A*	A	S, Z
POPD	Pop stack	2F	B	A	S, Z
CLR	CLR status	00	A	B	

Notes:

1. In the hex code column, SVREQ bit is a 0.
2. The stack initially is composed of four 32-bit numbers (A, B, C, D). A is equivalent to Top Of Stack (TOS) and B is Next on Stack (NOS). Upon completion of a command the stack is composed of: the result (R); undefined (U); or the initial contents (A,B,C, or D).
3. The stack initially is composed of two 64-bit numbers (A, B). A is equivalent to Top Of Stack (TOS) and B is Next On Stack (NOS). Upon completion of a command the stack is composed of: the result (R); undefined (U); or the initial contents (A, B).
4. Any status bit(s) not affected are set to 0. Nomenclature: Sign (S); Zero (Z); Exponent Underflow (U); Exponent Overflow (V); Divide Exception (D).
5. If the exponent field of A is zero, R or A\* will be zero.



Table 3. Execution Times

Command	TOS	NOS	Result	Clock Periods
SADD	3F800000	3F800000	40000000	58
SSUB	3F800000	3F800000	00000000	56
SMUL	40400000	3FC00000	40900000	198
SDIV	3F800000	40000000	3F000000	228
CHSS	3F800000	—	BF800000	10
PTOS	3F800000	—	—	16
POPS	3F800000	—	—	14
XCHS	3F800000	40000000	—	26
CHSD	3FF00000 00000000	—	BFF00000 00000000	24
PTOD	3FF00000 00000000	—	—	40
POPD	3FF00000 00000000	—	—	26
CLR	3FF00000 00000000	—	—	4
DADD	3FF00000 0A000000	3FF00000 00000000	3FF00000 0A000000	578
DSUB	3FF00000 A0000000	3FF00000 00000000	3FF00000 A0000000	578
DMUL	BFF80000 00000000	3FF80000 00000000	C0020000 00000000	1748
DDIV	BFF80000 00000000	3FF80000 00000000	BFF00000 00000000	4560

Note: TOS, NOS and result are in hexadecimal; clock period is in decimal.

### Command Initiation

After properly positioning the required operands in the stack, a command may be issued. The procedure for initiating a command execution is the same as that described above for operand entry, except that the  $A_0$  input is HIGH.

An attempt to issue a new command while the current command execution is in progress is allowed. Under these circumstances, the READY output will not go HIGH until the current command execution is completed.

### Removing the Results

Result from an operation will be available at the TOS. Results can be transferred from the stack to the data bus by reading the stack.

When the stack is read for results, the most significant byte is available first and the least significant byte last.

A result is always of the same precision as the operands that produced it. Thus, when the result is taken from the stack, the total number of bytes popped out should be appropriate with the precision — single precision results are 4 bytes and double precision results are 8 bytes. The following procedure must be used for reading the result from the stack:

1. A LOW is established on the  $A_0$  input.
2. The  $\overline{CS}$  input is made LOW. When  $\overline{WR}$  and  $\overline{RD}$  inputs are both HIGH, the READY output follows the  $\overline{CS}$  input, thus READY will be LOW.
3. After appropriate set up time (see timing diagrams), the  $\overline{RD}$  input is made LOW.

4. Sometime after this, READY will return HIGH, indicating that the data is available on the DB0-DB7 lines. This data will remain on the DB0-DB7 lines as long as the  $\overline{RD}$  input remains LOW.
5. Any time after READY goes HIGH, the  $\overline{RD}$  input can return HIGH to complete the transaction.
6. The  $\overline{CS}$  and  $A_0$  inputs can change after appropriate hold time requirements are satisfied (see timing diagram).
7. Repeat this procedure until all bytes appropriate for the precision of the result are popped out.

Reading of the stack does not alter its data; it only adjusts the byte pointer. Note data must be removed in even byte multiples to avoid a byte pointer misalignment. If more data is popped than the capacity of the stack, the internal byte pointer will wrap around and older data will be read again, consistent with the LIFO stack.

### Reading Status Register

The 8232 status register can be read without any regard to whether a command is in progress or not. The only implication that has to be considered is the effect this might have on the END and ERROR outputs discussed in the signal descriptions.

The following procedure must be followed to accomplish status register reading:

1. Establish HIGH on the  $A_0$  input.
2. Establish LOW on the  $\overline{CS}$  input. Whenever  $\overline{WR}$  and  $\overline{RD}$  inputs are HIGH, READY will follow the  $\overline{CS}$  input. Thus, READY will go LOW.
3. After appropriate set up time (see timing diagram),  $\overline{RD}$  is made LOW.

4. Sometime after the HIGH to LOW transition of  $\overline{RD}$ , READY will become HIGH, indicating that status register contents are available on the DB0-DB7 lines. These lines will contain this information as long as  $\overline{RD}$  is LOW.
5. The  $\overline{RD}$  input can be returned HIGH any time after READY goes HIGH.
6. The  $A_0$  input and  $\overline{CS}$  input can change after satisfying appropriate hold time requirements (see timing diagram).

**Status Register**

The 8232 contains an 8-bit status register with the following format:

BUSY	SIGN S	ZERO Z	RESERVED	DIVIDE EXCEPTION D	EXPONENT UNDERFLOW U	EXPONENT OVERFLOW V	RESERVED
7	6	5	4	3	2	1	0

All the bits are initialized to zero upon reset. Also, executing a CLR (Clear Status) command will result in all zero status register bits. A zero in bit 7 indicates that the 8232 is not busy and a new command may be initiated. As soon as a new command is issued, bit 7 becomes 1 to indicate the device is busy and remains 1 until the command execution is complete, at which time it will become 0. As soon as a new command is issued, status register bits 0-6 are cleared to zero. The status bits will be set as required during the command execution. Hence, as long as bit 7 is 1, the remainder of the status register bit indications should not be relied upon unless the ERROR occurs. The following is a detailed status bit description.

Bit 0 Reserved.

Bit 1 Exponent overflow (V). When 1, this bit indicates that the result exponent is more positive than +127 (+1023). The exponent is "wrapped" into the negative exponent range, skipping the end values.

Bit 2 Exponent Underflow (U). When 1, this bit indicates that the result exponent is more negative than -126 (-1022). The exponent is "wrapped" into the positive range by the number of underflow bits, skipping -127 (-1023) and +128 (+1024).

Bit 3 Divide Exception (D). When 1, this bit indicates that an attempt to divide by zero is made. Cleared to zero otherwise.

Bit 4 Reserved.

Bit 5 Zero (Z). When 1, this bit indicates that the result returned to TOS after a command is zero. Cleared to zero otherwise.

Bit 6 Sign (S). When 1, this bit indicates that the result returned to TOS is negative. Cleared to zero otherwise.

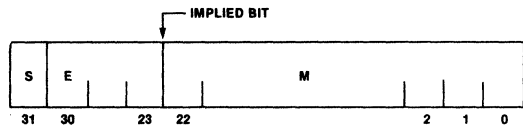
Bit 7 Busy. When 1, this bit indicates the 8232 is in the process of executing a command. It will become zero after the command execution is complete.

All other status register bits are valid when the Busy bit is zero.

**Data Formats**

The 8232 handles floating-point quantities in two different formats — single precision and double precision. These formats are the same as those used by Intel in other products and those proposed by the IEEE Subcommittee on floating point arithmetic.

The single precision quantities are 32 bits long, as shown below:



**Bit 31:**

S = Sign of the mantissa. One represents negative and 0 represents positive.

**Bits 23-30:**

E = These 8 bits represent a biased exponent. The bias is  $2^7 - 1 = 127$ .

**Bits 0-22:**

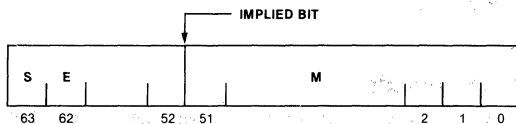
M = 23-bit mantissa. Together with the sign bit, the mantissa represents a signed fraction in sign-magnitude notation. There is an implied 1 beyond the most significant bit (bit 22) of the mantissa. In other words, the mantissa is assumed to be a 24-bit normalized quantity and the most significant bit, which will always be a 1 due to normalization, is implied. The 8232 restores this implied bit internally before performing arithmetic, normalizes the result and strips the implied bit before returning the results to the external data bus. The binary point is between the implied bit and bit 22 of the mantissa.

The quantity N represented by the above notation is

$$N = (-1)^S 2^{E - (2^7 - 1)} (1.M)$$

Provided  $E \neq 0$  (reserved for 0) or all 1's (illegal). The approximate decimal range for this format is  $\pm 1.17 \times 10^{-38}$  to  $\pm 3.40 \times 10^{38}$ . The format supports 7 significant decimal digits.

A double precision quantity consists of the mantissa sign bit, an 11-bit biased exponent (E), and a 52-bit mantissa (M). The bias for double precision quantities is  $2^{10} - 1$ . The double precision format is illustrated below.



**Bit 63:**

S = Sign of the mantissa. One represents negative and 0 represents positive.

**Bits 52-62:**

E = These 11 bits represent a biased exponent. The bias is  $2^{10} - 1 = 1023$ .

**Bits 0-51:**

M = 52-bit mantissa. Together with the sign bit the mantissa represents a signed fraction in sign-magnitude notation. There is an implied 1 beyond the most significant bit (bit 51) of the mantissa. In other words, the mantissa is assumed to be a 53-bit normalized quantity and the most significant bit, which will always be a 1 due to normalization, is implied. The 8232 restores this implied bit internally before performing arithmetic, normalizes the result and strips the implied bit before returning the result to the external data bus. The binary point is between the implied bit and bit 51 of the mantissa.

The quantity N represented by the above notation is

$$N = (-1)^S 2^{E - (2^{10} - 1)} (1.M)$$

↑ BIAS
↑ BINARY POINT

Provided E ≠ 0 (reserved for 0) or all 1s (illegal). The approximate decimal range is  $\pm 2.22 \times 10^{-308}$  to  $\pm 1.80 \times 10^{308}$ . The format supports 16 significant decimal digits.

The following are some examples of single precision floating point representations:

Decimal	S	E	M	Binary Floating Point
0	0	0	0	0000 0000H
1	0	127	0	3F80 0000H
-1	1	127	0	BF80 0000H
255	0	134	.9922	437F 0000H
π	0	128	.5708	4049 0FDBH

**Rounding**

One of the main objectives in choosing the 8232's Intel/IEEE proposed floating point arithmetic was to provide maximum accuracy with no anomalies. This means that a mathematically unsophisticated user will not be "surprised" by some of the results. It is probably possible for a sophisticated user to obtain reliable results from almost any floating point arithmetic. However, in that case there will be an additional burden on the software.

The best example of what might be called the 8232's "safety factor" is the inclusion of guard bits for rounding. The absence of guard bits leads to the problem demonstrated by the following four-bit multiplication:

$$\begin{array}{r} .1111 \times 2^0 \\ .1000 \times 2^1 \\ \hline .01111000 \times 2^1 \end{array}$$

Since the last four bits are lost, the normalized result is:

$$.1110 \times 2^0$$

and the identify function is not valid. In the past this problem has been avoided (hopefully) by relying on excess precision.

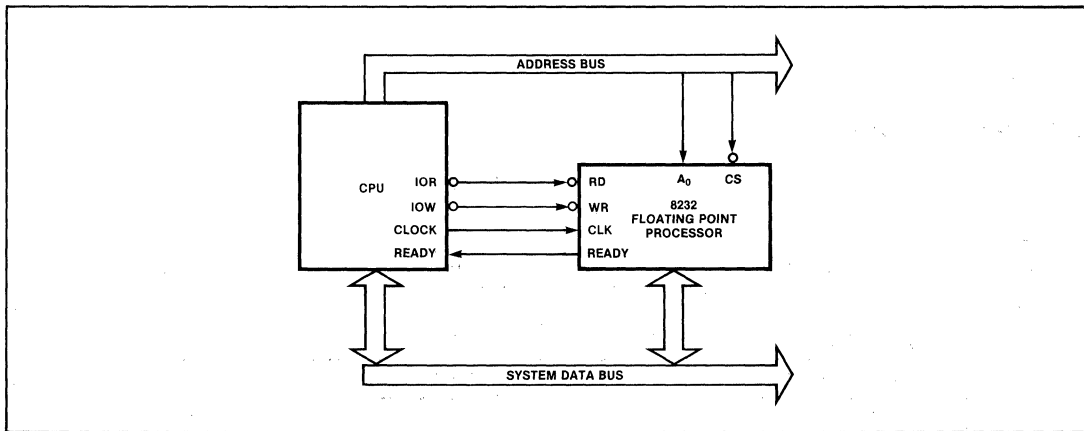
Instead the 8232 uses a form of rounding known as "round to even." There are other types of rounding provided for in the proposed IEEE standard, but "round to even," an unbiased rounding scheme, is required. "Round to even" comes into play when a result is exactly halfway between two floating point numbers. In this case the arithmetic produces the "even" number, the one whose last mantissa bit is zero. The 8232 uses three additional bits—the Guard bit (G), the Rounding bit (R), and the "Sticky" bit (S)—to do the rounding. These are bits which hold data shifted out (right) of the accumulator. Rounding is carried out by the following rules, as shown in the following figure, after the result is normalized.

G		Bit R		S	Rule
0	0	0	0	0	No Round
0	0	0	1	1	Round Down
0	1	0	0	0	
0	1	0	1	1	
1	0	0	0	0	Round to Even
1	0	1	0	1	Round Up
1	1	0	0	0	
1	1	1	0	1	

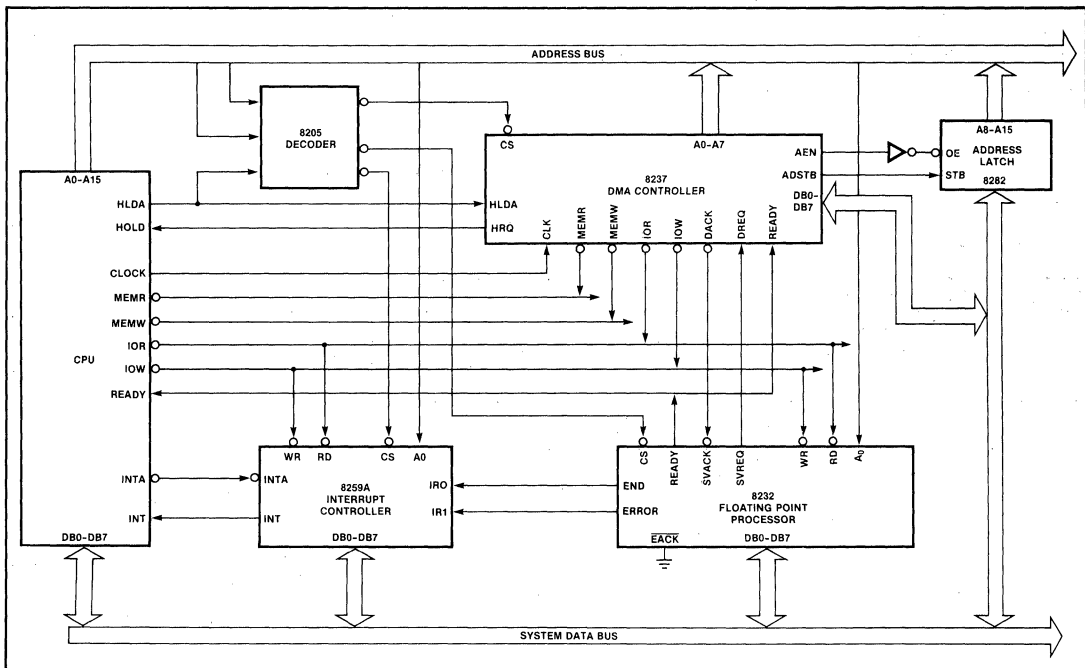
**APPLICATIONS INFORMATION**

The diagram in Figure 3 represents the minimum configuration of an 8232 system. The CPU transfers data to and from the 8232 Floating Point Processor using the READY line. The 8232 status is checked using polling by the CPU.

In a high performance configuration (Figure 4), interrupts are used in place of polling. The interrupts are generated for an error condition and to signal the end of execution. Operand transfers are handled by the DMA controller.



**Figure 3. Minimum Configuration Example**



**Figure 4. High Performance Configuration Example**

**ABSOLUTE MAXIMUM RATINGS\***

Storage Temperature ..... -65°C to +150°C  
 Ambient Temperature Under Bias ..... 0°C to +70°C  
 V<sub>DD</sub> with Respect to V<sub>SS</sub> ..... -0.5V to +15.0V  
 V<sub>CC</sub> with Respect to V<sub>SS</sub> ..... -0.5V to +7.0V  
 All Signal Voltages with Respect  
 to V<sub>SS</sub> ..... -0.5V to +7.0V  
 Power Dissipation ..... 2.0W

*\*NOTICE: Stresses above those listed under "Absolute Maximum Ratings" may cause permanent damage to the device. This is a stress rating only and functional operation of the device at these or any other conditions above those indicated in the operational sections of this specification is not implied. Exposure to absolute maximum rating conditions for extended periods may affect device reliability.*

**D.C. CHARACTERISTICS** (T<sub>A</sub> = 0°C to 70°C, V<sub>SS</sub> = 0V, V<sub>CC</sub> = +5V ± 10%, V<sub>DD</sub> = +12V ± 10%)

Symbol	Parameter	Min.	Typ.	Max.	Units	Test Conditions
V <sub>OH</sub>	Output HIGH Voltage	3.7			V	I <sub>OH</sub> = -200 μA
V <sub>OL</sub>	Output LOW Voltage			0.4	V	I <sub>OL</sub> = 3.2 mA
V <sub>IH</sub>	Input HIGH Voltage	2.0		V <sub>CC</sub>	V	
V <sub>IL</sub>	Input LOW Voltage	-0.5		0.8	V	
I <sub>IL</sub>	Input Load Current			±10	μA	V <sub>SS</sub> ≤ V <sub>IN</sub> ≤ V <sub>CC</sub>
I <sub>OZ</sub>	Data Bus Leakage			±10	μA	V <sub>SS</sub> + 0.4 ≤ V <sub>OUT</sub> ≤ V <sub>CC</sub>
I <sub>CC</sub>	V <sub>CC</sub> Supply Current		50	95	mA	
I <sub>DD</sub>	V <sub>DD</sub> Supply Current		50	95	mA	
C <sub>O</sub>	Output Capacitance		8		pF	f <sub>C</sub> = 1.0 MHz, Inputs = 0V
C <sub>I</sub>	Input Capacitance		5		pF	
C <sub>IO</sub>	I/O Capacitance		10		pF	

**A.C. CHARACTERISTICS** (T<sub>A</sub> = 0°C to 70°C, V<sub>SS</sub> = 0V, V<sub>CC</sub> = +5V ± 10%, V<sub>DD</sub> = +12V ± 10%)

**READ OPERATION**

Symbol	Parameter		8232		8232-3		8232-8		Units
			Min.	Max.	Min.	Max.	Min.	Max.	
t <sub>AR</sub>	A <sub>0</sub> , $\overline{CS}$ Setup to $\overline{RD}$		0		0		0		ns
t <sub>RA</sub>	A <sub>0</sub> , $\overline{CS}$ Hold from $\overline{RD}$		0		0		0		ns
t <sub>ARY</sub>	READY↑ from A <sub>0</sub> , $\overline{CS}$ ↓ Delay (Note 2)			100		100		150	ns
t <sub>YR</sub>	READY↑ to $\overline{RD}$ ↑		0		0		0		ns
t <sub>RRR</sub>	READY Pulse Width (Note 3)		Data	3.5 t <sub>CY</sub> + 50		3.5 t <sub>CY</sub> + 50		3.5 t <sub>CY</sub> + 50	ns
			Status	1.5 t <sub>CY</sub> + 50		1.5 t <sub>CY</sub> + 50		1.5 t <sub>CY</sub> + 50	ns
t <sub>RDE</sub>	Data Bus Enable from $\overline{RD}$ ↓		50		50		50		ns
t <sub>DRY</sub>	Data Valid to READY↑		0		0		0		ns
t <sub>DF</sub>	Data Float after $\overline{RD}$ ↑		20	100	20	150	20	200	ns

**A.C. CHARACTERISTICS (Continued)**

**WRITE OPERATION**

Symbol	Parameter	8232		8232-3		8232-8		Units
		Min.	Max.	Min.	Max.	Min.	Max.	
$t_{AW}$	$A_0, \overline{CS}$ Setup to $\overline{WR}$	25		25		25		ns
$t_{WA}$	$A_0, \overline{CS}$ Hold after $\overline{WR}$	30		30		60		ns
$t_{AWY}$	READY↓ from $A_0, \overline{CS}$ ↓ Delay (Note 2)		100		100		150	ns
$t_{YW}$	READY↑ to $\overline{WR}$ ↑	0		0		0		ns
$t_{RRW}$	READY Pulse Width		$t_{AW} + 50$		$t_{AW} + 50$		$t_{AW} + 50$	ns
$t_{DW}$	Data Setup to $\overline{WR}$ ↑	100		100		150		ns
$t_{WD}$	Data Hold after $\overline{WR}$ ↑	20		20		20		ns

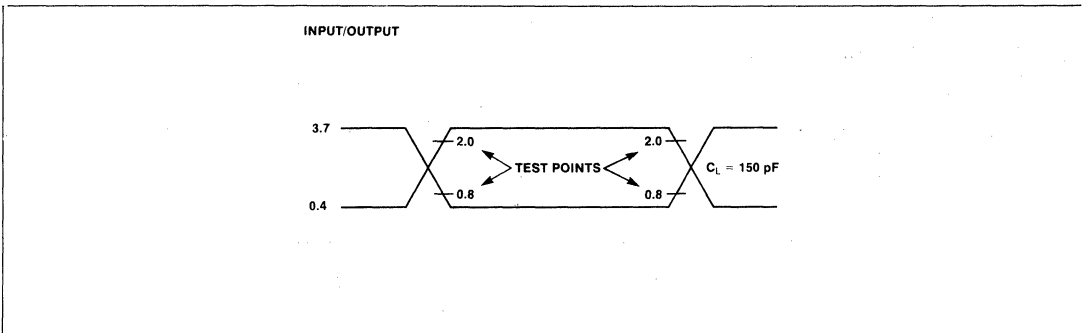
**OTHER TIMINGS**

Symbol	Parameter	8232		8232-3		8232-8		Units
		Min.	Max.	Min.	Max.	Min.	Max.	
$t_{CY}$	Clock Period	250	2500	320	3300	480	5000	ns
$t_{CPH}$	Clock Pulse HIGH Width	100		140		200		ns
$t_{CPL}$	Clock Pulse LOW Width	120		160		240		ns
$t_{EE}$	END Pulse Width (Note 4)	200		300		400		ns
$t_{EAE}$	$\overline{EACK}$ ↓ to $\overline{END}$ ↓ Delay		150		175		200	ns
$t_{AA}$	$\overline{EACK}$ Pulse Width	50		75		100		ns
$t_{SA}$	$\overline{SVACK}$ ↓ to $\overline{SVREQ}$ ↓ Delay		100		200		300	ns
$t_{SS}$	$\overline{SVACK}$ Pulse Width	50		75		100		ns

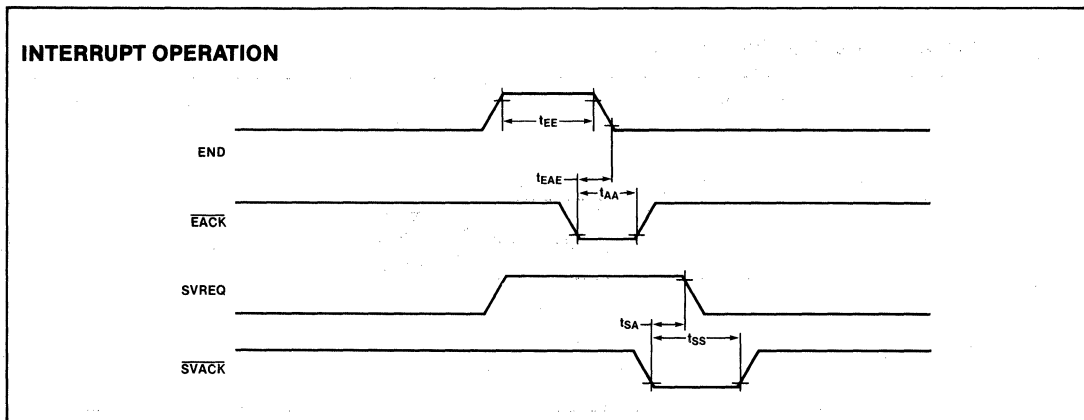
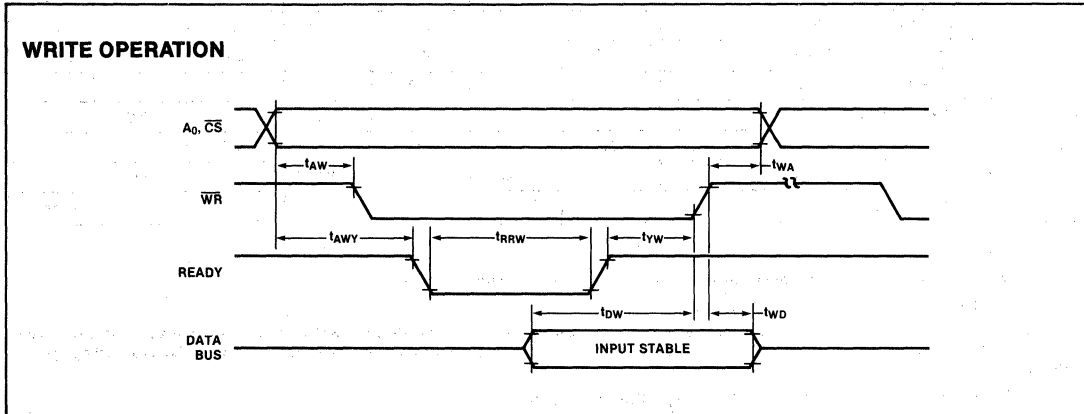
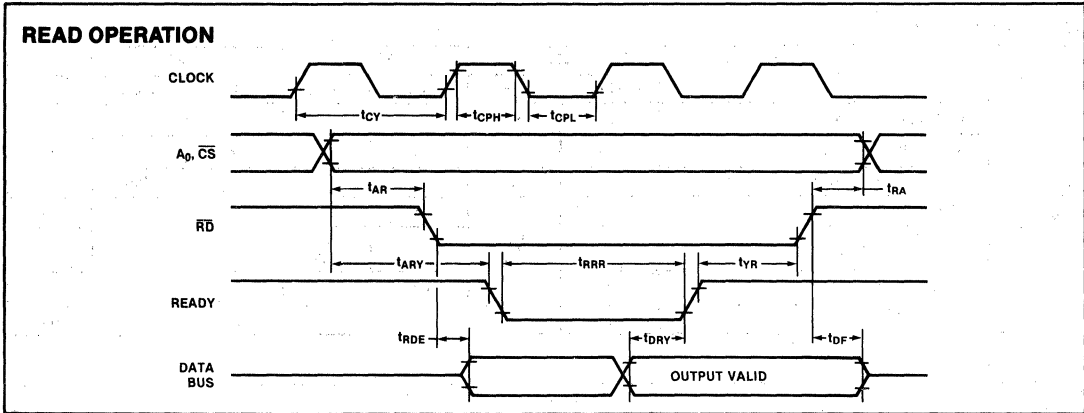
**NOTES:**

1. Typical values are for  $T_A = 25^\circ\text{C}$ , nominal supply voltages and nominal processing parameters.
2. READY is pulled low for both command and data operations.
3. Minimum values shown assume no previously entered command is being executed for the data access. If a previously entered command is being executed, READY low pulse width is the time to complete execution plus the time shown. Status may be read at any time without exceeding the time shown.
4. END high pulse width is specified for  $\overline{EACK}$  tied to  $V_{SS}$ . Otherwise  $t_{EAE}$  applies.

**A.C. TESTING INPUT, OUTPUT WAVEFORM**



WAVEFORMS





# 8294 DATA ENCRYPTION UNIT

- Certified by National Bureau of Standards
- 80 Byte/Sec Data Conversion Rate
- 64-Bit Data Encryption Using 56-Bit Key
- DMA Interface
- 3 Interrupt Outputs to Aid in Loading and Unloading Data
- 7-Bit User Output Port
- Single 5V ± 10% Power Supply
- Peripheral to MCS-86™, MCS-85™, MCS-80™ and MCS-48™ Processors
- Implements Federal Information Processing Data Encryption Standard
- Encrypt and Decrypt Modes Available

The Intel® 8294 Data Encryption Unit (DEU) is a microprocessor peripheral device designed to encrypt and decrypt 64-bit blocks of data using the algorithm specified in the Federal Information Processing Data Encryption Standard. The DEU operates on 64-bit text words using a 56-bit user-specified key to produce 64-bit cipher words. The operation is reversible: if the cipher word is operated upon, the original text word is produced. The algorithm itself is permanently contained in the 8294; however, the 56-bit key is user-defined and may be changed at any time.

The 56-bit key and 64-bit message data are transferred to and from the 8294 in 8-bit bytes by way of the system data bus. A DMA interface and three interrupt outputs are available to minimize software overhead associated with data transfer. Also, by using the DMA interface two or more DEUs may be operated in parallel to achieve effective system conversion rates which are virtually any multiple of 80 bytes/second. The 8294 also has a 7-bit TTL compatible output port for user-specified functions.

Because the 8294 implements the NBS encryption algorithm it can be used in a variety of Electronic Funds Transfer applications as well as other electronic banking and data handling applications where data must be encrypted.

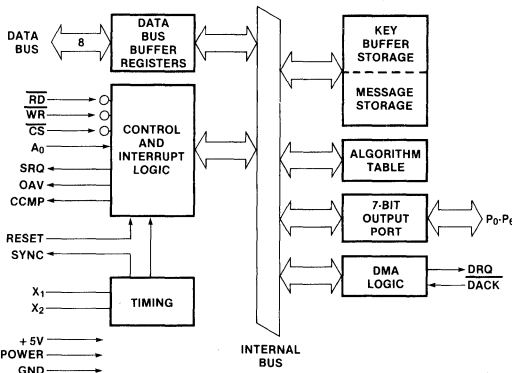


Figure 1. Block Diagram

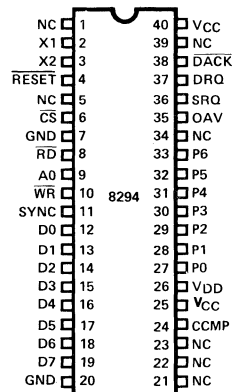


Figure 2. Pin Configuration



Table 1. Pin Description

Symbol	Pin No.	Type	Name and Function
NC	1		<b>No Connection.</b>
X1 X2	2 3	I	<b>Crystal:</b> Inputs for crystal, L-C or external timing signal to determine internal oscillator frequency.
$\overline{\text{RESET}}$	4	I	<b>Reset:</b> A low signal to this pin resets the 8294.
NC	5		<b>No Connection:</b> No connection or tied high.
$\overline{\text{CS}}$	6	I	<b>Chip Select:</b> A low signal to this pin enables reading and writing to the 8294.
GND	7		<b>Ground:</b> This pin must be tied to ground.
$\overline{\text{RD}}$	8	I	<b>Read:</b> An active low read strobe at this pin enables the CPU to read data and status from the internal DEU registers.
A <sub>0</sub>	9	I	<b>Address:</b> Address input used by the CPU to select DEU registers during read and write operations.
$\overline{\text{WR}}$	10	I	<b>Write:</b> An active low write strobe at this pin enables the CPU to send data and commands to the DEU.
SYNC	11	O	<b>Sync:</b> High frequency (Clock ÷ 15) output. Can be used as a strobe for external circuitry.
D <sub>0</sub> D <sub>1</sub> D <sub>2</sub> D <sub>3</sub> D <sub>4</sub> D <sub>5</sub> D <sub>6</sub> D <sub>7</sub>	12 13 14 15 16 17 18 19	I/O	<b>Data Bus:</b> Three-state, bi-directional data bus lines used to transfer data between the CPU and the 8294.
GND	20		<b>Ground:</b> This pin must be tied to ground.
V <sub>CC</sub>	40		<b>Power:</b> +5 volt power input: +5V ± 10%.

Symbol	Pin No.	Type	Name and Function
NC	39		<b>No Connection.</b>
$\overline{\text{DACK}}$	38	I	<b>DMA Acknowledge:</b> Input signal from the 8257 DMA Controller acknowledging that the requested DMA cycle has been granted.
DRQ	37	O	<b>DMA Request:</b> Output signal to the 8257 DMA Controller requesting a DMA cycle.
SRQ	38	O	<b>Service Request:</b> Interrupt to the CPU indicating that the 8294 is awaiting data or commands at the input buffer. SRQ=1 implies IBF=0.
OAV	35	O	<b>Output Available:</b> Interrupt to the CPU indicating that the 8294 has data or status available in its output buffer. OAV=1 implies OBF=1.
NC	34		<b>No Connection.</b>
P6 P5 P4 P3 P2 P1 P0	33 32 31 30 29 28 27	O	<b>Output Port:</b> User output port lines. Output lines available to the user via a CPU command which can assert selected port lines. These lines have nothing to do with the encryption function. At power-on, each line is in a 1 state.
V <sub>DD</sub>	26		<b>Power:</b> +5V power input. (+5V ±10%) Low power standby pin.
V <sub>CC</sub>	25		<b>Power:</b> Tied high.
CCMP	24	O	<b>Conversion Complete:</b> Interrupt to the CPU indicating that the encryption/decryption of an 8-byte block is complete.
NC	23		<b>No Connection.</b>
NC	22		<b>No Connection.</b>
NC	21		<b>No Connection.</b>

**FUNCTIONAL DESCRIPTION**

**OPERATION**

The data conversion sequence is as follows:

1. A Set Mode command is given, enabling the desired interrupt outputs.
2. An Enter New Key command is issued, followed by 8 data inputs which are retained by the DEU for encryption/decryption. Each byte must have odd parity.
3. An Encrypt Data or Decrypt Data command sets the DEU in the desired mode.

After this, data conversions are made by writing 8 data bytes and then reading back 8 converted data bytes. Any of the above commands may be issued between data conversions to change the basic operation of the DEU; e.g., a Decrypt Data command could be issued to change the DEU from **encrypt** mode to **decrypt** mode without changing either the key or the interrupt outputs enabled.

**INTERNAL DEU REGISTERS**

Four internal registers are addressable by the master processor: 2 for input, and 2 for output. The following table describes how these registers are accessed.

RD	WR	CS	A <sub>0</sub>	Register
1	0	0	0	Data input buffer
0	1	0	0	Data output buffer
1	0	0	1	Command input buffer
0	1	0	1	Status output buffer
X	X	1	X	Don't care

The functions of each of these registers are described below.

**Data Input Buffer** — Data written to this register is interpreted in one of three ways, depending on the preceding command sequence.

1. Part of a key.
2. Data to be encrypted or decrypted.
3. A DMA block count.

**Data Output Buffer** — Data read from this register is the output of the encryption/decryption operation.

**Command Input Buffer** — Commands to the DEU are written into this register. (See command summary below.)

**Status Output Buffer** — DEU status is available in this register at all times. It is used by the processor for poll-driven command and data transfer operations.

STATUS BIT:	7	6	5	4	3	2	1	0
FUNCTION:	X	X	X	KPE	CF	DEC	IBF	OBF

**OBF** Output Buffer Full; OBF = 1 indicates that output from the encryption/decryption function is available in the Data Output Buffer. It is reset when the data is read.

**IBF** Input Buffer Full; A write to the Data Input Buffer or to the Command Input Buffer sets IBF = 1. The DEU resets this flag when it has accepted the input byte. Nothing should be written when IBF = 1.

**DEC** Decrypt; indicates whether the DEU is in an encrypt or a decrypt mode. DEC = 1 implies the decrypt mode. DEC = 0 implies the encrypt mode.

**CF** Completion Flag; This flag may be used to indicate any or all of three events in the data transfer protocol.

1. It may be used in lieu of a counter in the processor routine to flag the end of an 8-byte transfer.
2. It must be used to indicate the validity of the KPE flag.
3. It may be used in lieu of the CCMP interrupt to indicate the completion of a DMA operation.

**KPE** Key Parity Error; After a new key has been entered, the DEU uses this flag in conjunction with the CF flag to indicate correct or incorrect parity.

**COMMAND SUMMARY**

**1 — Enter New Key**

OP CODE: 

0	1	0	0	0	0	0	0
---	---	---	---	---	---	---	---

  
MSB LSB

This command is followed by 8 data byte inputs which are retained in the key buffer (RAM) to be used in encrypting and decrypting data. These data bytes must have odd parity represented by the LSB.

**2 — Encrypt Data**

OP CODE: 

0	0	1	1	0	0	0	0
---	---	---	---	---	---	---	---

  
MSB LSB

This command puts the 8294 into the encrypt mode.

**3 — Decrypt Data**

OP CODE: 

0	0	1	0	0	0	0	0
---	---	---	---	---	---	---	---

  
MSB LSB

This command puts the 8294 into the decrypt mode.

**4 — Set Mode**

OP CODE: 

0	0	0	0	A	B	C	D
---	---	---	---	---	---	---	---

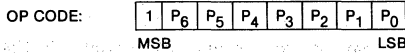
  
MSB LSB

where:

- A is the OAV (Output Available) interrupt enable
- B is the SRQ (Service Request) interrupt enable
- C is the DMA (Direct Memory Access) transfer enable
- D is the CCMP (Conversion Complete) interrupt enable

This command determines which interrupt outputs will be enabled. A "1" in bits A, B, or D will enable the OAV, SRQ, or CCMP interrupts respectively. A "1" in bit C will allow DMA transfers. When bit C is set the OAV and SRQ interrupts should also be enabled (bits A,B = 1). Following the command in which bit C, the DMA bit, is set, the 8294 will expect one data byte to specify the number of 8-byte blocks to be converted using DMA.

**5 — Write to Output Port**



This command causes the 7 least significant bits of the command byte to be latched as output data on the 8294 output port. The initial output data is 1111111. Use of this port is independent of the encryption/decryption function.

**PROCESSOR/DEU INTERFACE PROTOCOL  
ENTERING A NEW KEY**

The timing sequence for entering a new key is shown in Figure 3. A flowchart showing the CPU software to accommodate this sequence is given in Figure 4.

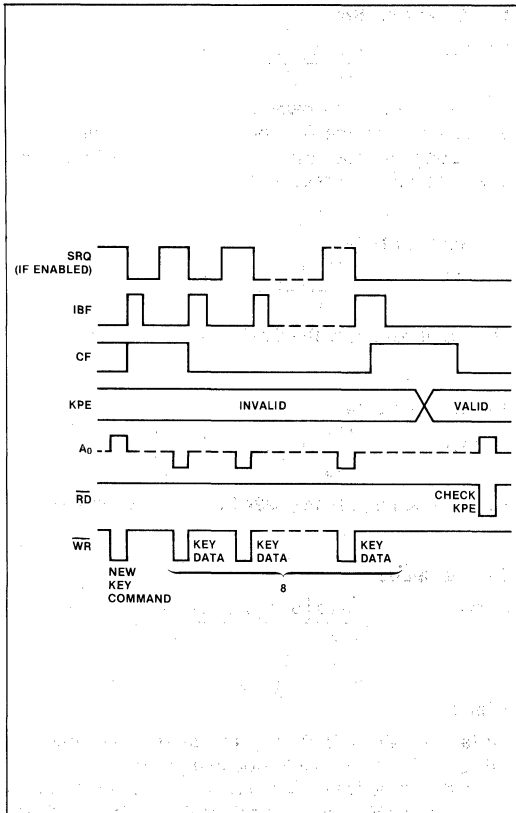


Figure 3. Entering a New Key

After the Enter New Key command is issued, 8 data bytes representing the new key are written to the data input buffer (most significant byte first). After the eighth byte is accepted by the DEU, CF goes true (CF = 1). The CF bit goes false again when KPE is valid. The CPU can then check the KPE flag. If KPE = 1, a parity error has been detected and the DEU has not accepted the key. Each byte is checked for odd parity, where the parity bit is the LSB of each byte.

Since the CF bit is used in this protocol to indicate the validity of the KPE flag, it may not be used to flag the end of the 8 byte key entry. CF = 1 only as long as KPE is invalid. Therefore, the CPU might not detect that CF = 1 and the key entry is complete before KPE becomes valid. Thus, a counter should be used, as in Figure 4, to flag the end of the new key entry. Then, CF is used to indicate a valid KPE flag.

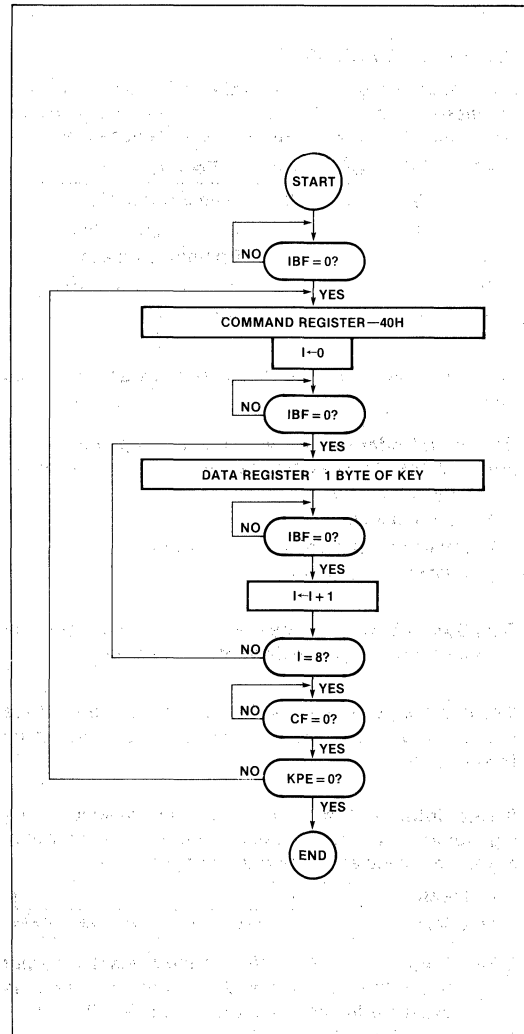
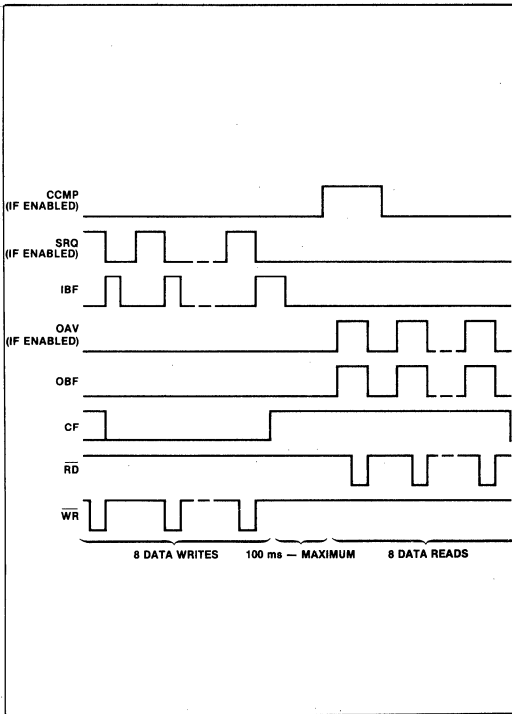


Figure 4. Flowchart for Entering a New Key

**ENCRYPTING OR DECRYPTING DATA**

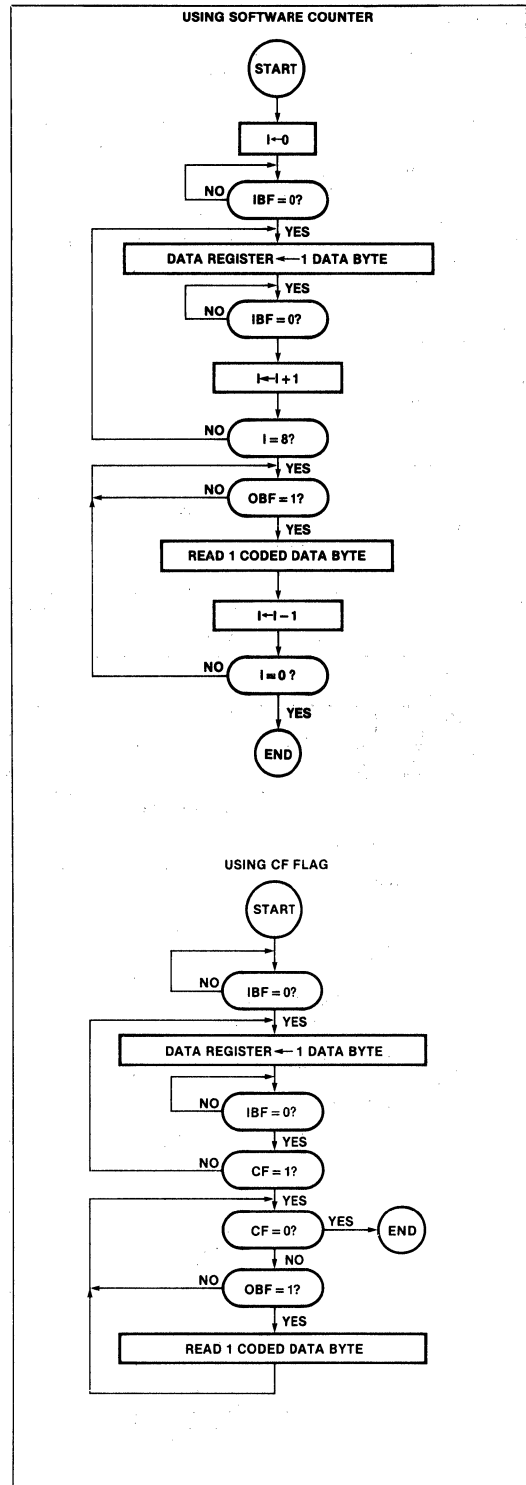
Figure 5 shows the timing sequence for encrypting or decrypting data. The CPU writes 8 data bytes to the DEU's data input buffer for encryption/decryption. CF then goes true (CF = 1) to indicate that the DEU has accepted the 8-byte block. Thus, the CPU may test for  $IBF = 0$  and  $CF = 1$  to terminate the input mode, or it may use a software counter. When the encryption/decryption is complete, the CCMP and OAV interrupts are asserted and the OBF flag is set true (OBF = 1). OAV and OBF are set false again after each of the converted data bytes is read back by the CPU. The CCMP interrupt is set false, and remains false, after the first read. After 8 bytes have been read back by the CPU, CF goes false (CF = 0). Thus, the CPU may test for  $CF = 0$  to terminate the read mode. Also, the CCMP interrupt may be used to initiate a service routine which performs the next series of 8 data reads and 8 data writes.



**Figure 5. Encrypting/Decrypting Data**

Figure 6 offers two flowcharts outlining the alternative means of implementing the data conversion protocol. Either the CF flag or a software counter may be used to end the read and write modes.

$SRQ = 1$  implies  $IBF = 0$ ,  $OAV = 1$  implies  $OBF = 1$ . This allows interrupt routines to do data transfers without checking status first. However, the OAV service routine must detect and flag the end of a data conversion.



**Figure 6. Data Conversion Flowcharts**

### USING DMA

The timing sequence for data conversions using DMA is shown in Figure 7. This sequence can be better understood when considered in conjunction with the hardware DMA interface in Figure 8. Note that the use of the DMA feature requires 3 external AND gates and 2 DMA channels (one for input, one for output). Since the DEU has only one DMA request pin, the SRQ and OAV outputs are used in conjunction with two of the AND gates to create separate DMA request outputs for the 2 DMA channels. The third AND gate combines the two active-low  $\overline{\text{DACK}}$  inputs.

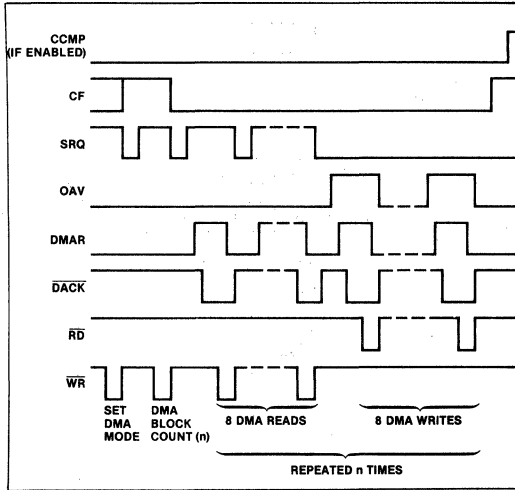


Figure 7. DMA Sequence

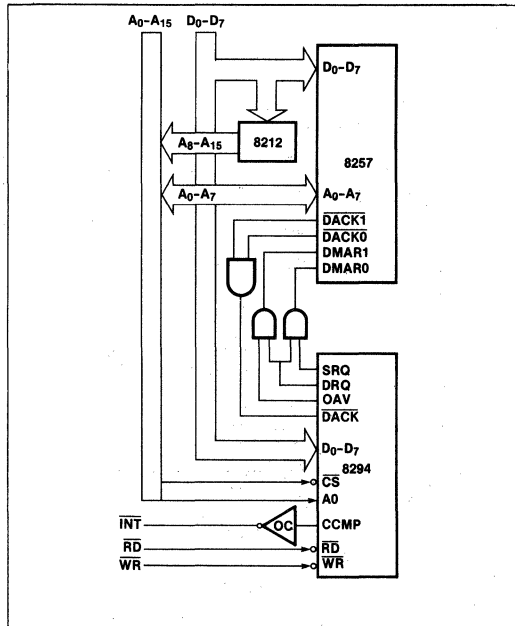


Figure 8. DMA Interface

To initiate a DMA transfer, the CPU must first initialize the two DMA channels as shown in the flowchart in Figure 9. It must then issue a Set Mode command to the DEU enabling the OAV, SRQ, and DMA outputs. The CCMP interrupt may be enabled or disabled, depending on whether that output is desired. Following the Set Mode command, there must be a data byte giving the number of 8-byte blocks of data ( $n < 256$ ) to be converted. The DEU then generates the required number of DMA requests to the 2 DMA channels with no further CPU intervention. When the requested number of blocks has been converted, the DEU will set CF and assert the CCMP interrupt (if enabled). CCMP then goes false again with the next write to the DEU (command or data). Upon completion of the conversion, the DMA mode is disabled and the DEU returns to the encrypt/decrypt mode. The enabled interrupt outputs, however, will remain enabled until another Set Mode command is issued.

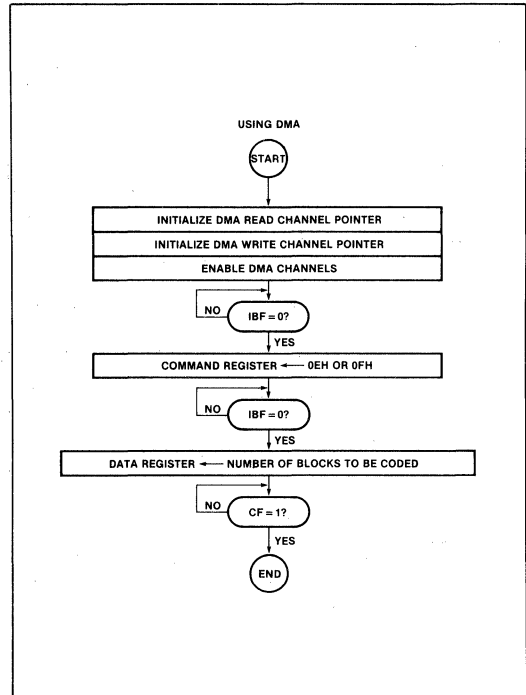


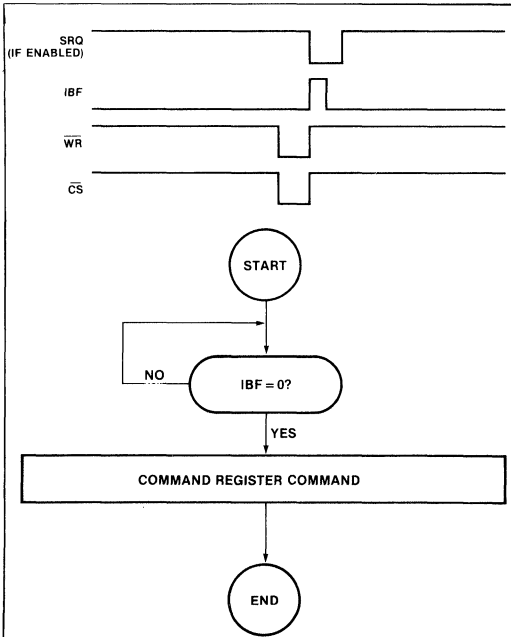
Figure 9. DMA Flowchart

### SINGLE BYTE COMMANDS

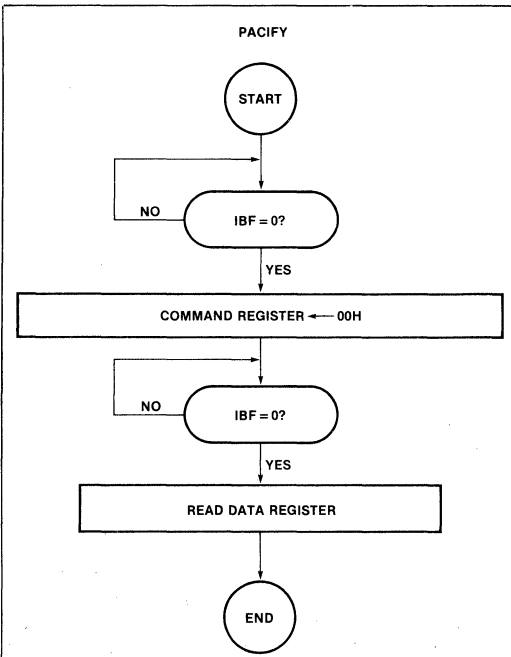
Figure 10 shows the timing and protocol for single byte commands. Note that any of the commands is effective as a pacify command in that they may be entered at any time, except during a DMA conversion. The DEU is thus set to a known state. However, if a command is issued out of sequence, an additional protocol is required (Figure 11). The CPU must wait until the command is accepted ( $\text{IBF} = 0$ ). A data read must then be issued to clear anything the preceding command sequence may have left in the Data Output Buffer.

**CPU/DEU INTERFACES**

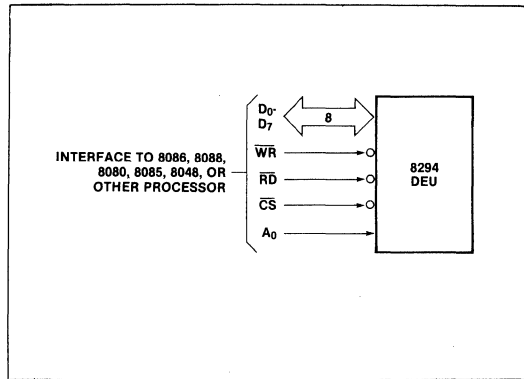
Figures 12 through 15 illustrate four interface configurations used in the CPU/DEU data transfers. In all cases SRQ will be true (if enabled) and IBF will be false when the DEU is ready to accept data or commands.



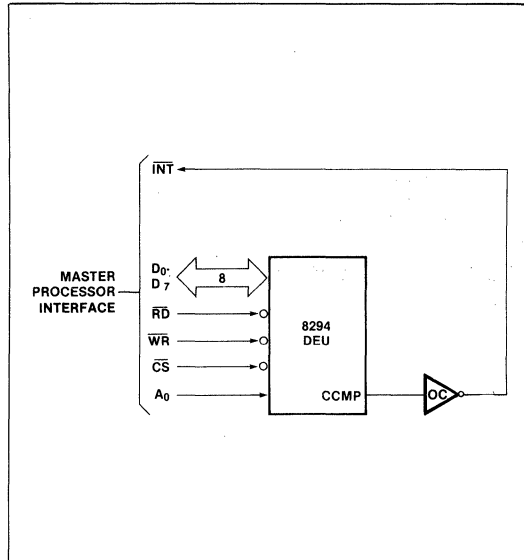
**Figure 10. Single Byte Commands**



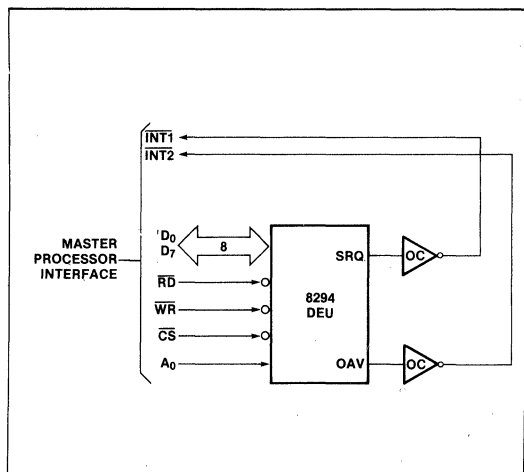
**Figure 11. Pacify Protocol**



**Figure 12. Polling Interface**



**Figure 13. Single Interrupt Interface**



**Figure 14. Dual Interrupt Interface**

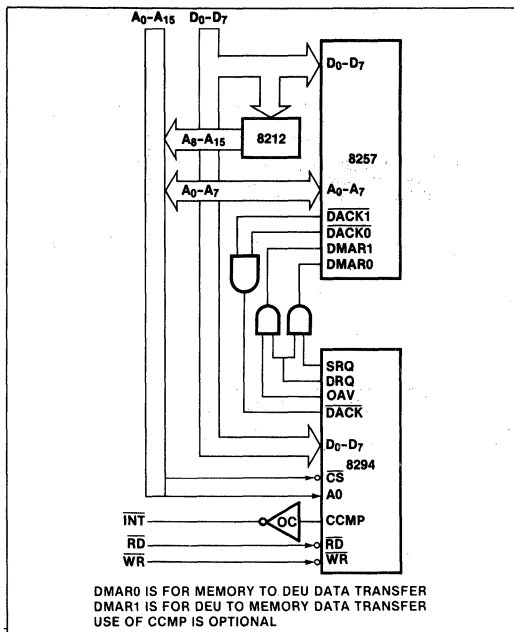


Figure 15. DMA Interface

**OSCILLATOR AND TIMING CIRCUITS**

The 8294's internal timing generation is controlled by a self-contained oscillator and timing circuit. A choice of crystal, L-C or external clock can be used to derive the basic oscillator frequency.

The resident timing circuit consists of an oscillator, a state counter and a cycle counter as illustrated in Figure 16.

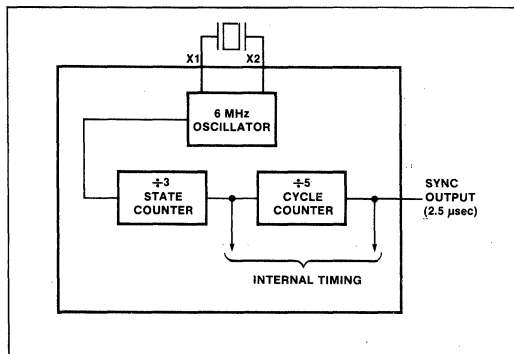


Figure 16. Oscillator Configuration

**OSCILLATOR**

The on-board oscillator is a series resonant circuit with a frequency range of 1 to 6 MHz. Pins X1 and X2 are input and output (respectively) of a high gain amplifier stage. A crystal or inductor and capacitor connected between X1 and X2 provide the feedback and proper phase shift for oscillation. Recommended connections for crystal or L-C are shown in Figure 17.

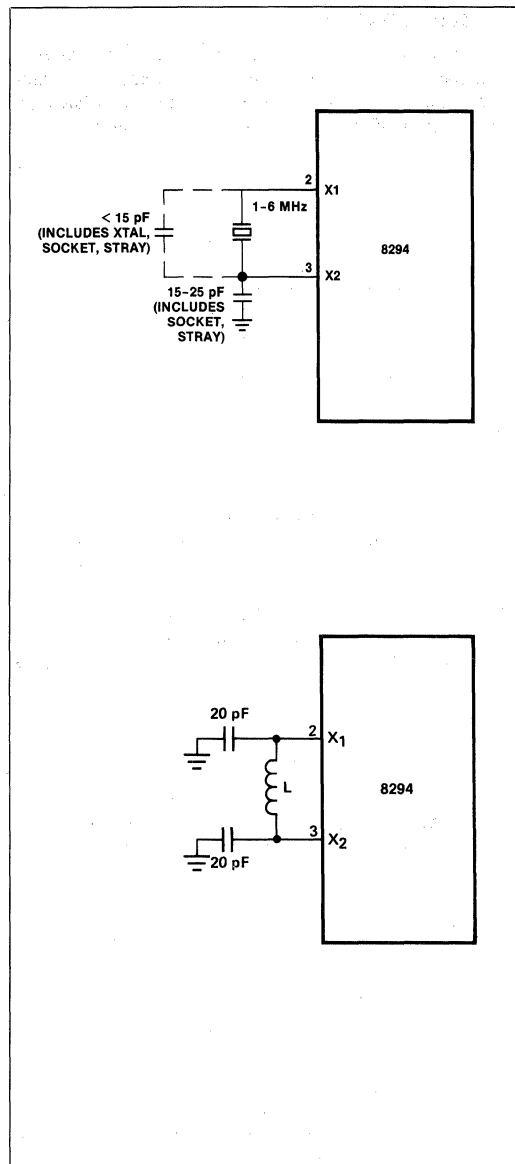


Figure 17. Recommended Crystal and L-C Connections

A recommended range of inductance and capacitance combinations is given below:

$L = 120 \mu\text{H}$  corresponds to 3MHz

$L = 45 \mu\text{H}$  corresponds to 5MHz

An external clock signal can also be used as a frequency reference to the 8294; however, the levels are *not* compatible. The signal must be in the 1MHz-6MHz frequency range and must be connected to pins X1 and X2 by buffers with a suitable pull-up resistor to guarantee that a logic "1" is above 3.8 volts. The recommended connection is shown in Figure 18.

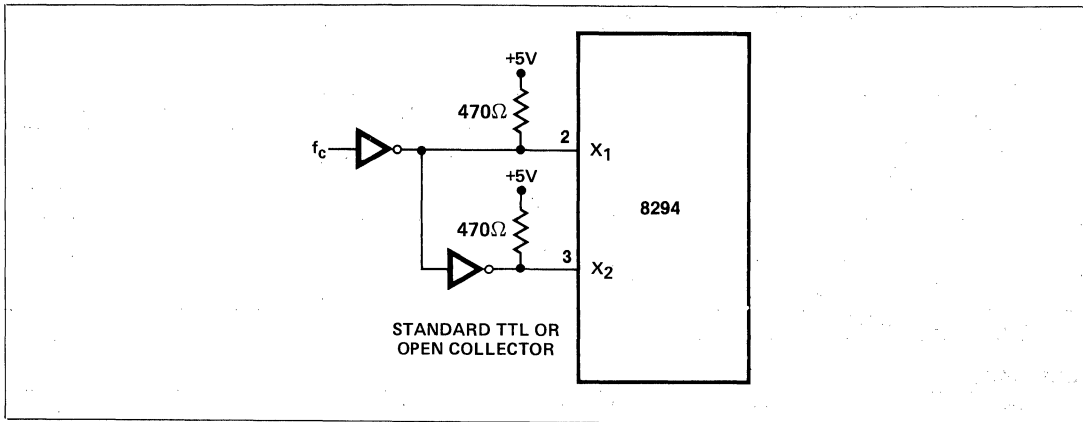


Figure 18. Recommended Connection for External Clock Signal

**ABSOLUTE MAXIMUM RATINGS\***

Ambient Temperature Under Bias ..... 0°C to 70°C  
 Storage Temperature ..... - 65°C to + 150°C  
 Voltage on Any Pin With  
 Respect to Ground ..... -0.5V to +7V  
 Power Dissipation ..... 1.5 Watt

\*NOTICE: Stresses above those listed under "Absolute Maximum Ratings" may cause permanent damage to the device. This is a stress rating only and functional operation of the device at these or any other conditions above those indicated in the operational sections of this specification is not implied. Exposure to absolute maximum rating conditions for extended periods may affect device reliability.

**D.C. AND OPERATING CHARACTERISTICS** (T<sub>A</sub> = 0°C to 70°C, V<sub>CC</sub> = +5V ± 10%, V<sub>SS</sub> = 0V)

Symbol	Parameter	Limits			Unit	Test Conditions
		Min.	Typ.	Max.		
V <sub>IL</sub>	Input Low Voltage (All Except X <sub>1</sub> , X <sub>2</sub> , RESET)	-0.5		0.8	V	
V <sub>IL1</sub>	Input Low Voltage (X <sub>1</sub> , X <sub>2</sub> , RESET)	-0.5		0.6	V	
V <sub>IH</sub>	Input High Voltage (All Except X <sub>1</sub> , X <sub>2</sub> , RESET)	2.2		V <sub>CC</sub>	V	
V <sub>IH1</sub>	Input High Voltage (X <sub>1</sub> , X <sub>2</sub> , RESET)	3.8		V <sub>CC</sub>	V	
V <sub>OL</sub>	Output Low Voltage (D <sub>0</sub> -D <sub>7</sub> )			0.45	V	I <sub>OL</sub> = 2.0 mA
V <sub>OL1</sub>	Output Low Voltage (All Other Outputs)			0.45	V	I <sub>OL</sub> = 1.6 mA
V <sub>OH</sub>	Output High Voltage (D <sub>0</sub> -D <sub>7</sub> )	2.4			V	I <sub>OH</sub> = -400 μA
V <sub>OH1</sub>	Output High Voltage (All Other Outputs)	2.4			V	I <sub>OH</sub> = -50 μA
I <sub>IL</sub>	Input Leakage Current (RD, WR, CS, A <sub>0</sub> )			± 10	μA	V <sub>SS</sub> ≤ V <sub>IN</sub> ≤ V <sub>CC</sub>
I <sub>oZ</sub>	Output Leakage Current (D <sub>0</sub> -D <sub>7</sub> , High Z State)			± 10	μA	V <sub>SS</sub> + 0.45 ≤ V <sub>OUT</sub> ≤ V <sub>CC</sub>
I <sub>DD</sub>	V <sub>DD</sub> Supply Current		5	15	mA	
I <sub>DD</sub> + I <sub>CC</sub>	Total Supply Current		60	125	mA	
I <sub>LI</sub>	Low Input Load Current (Pins 24, 27-38)			0.5	mA	V <sub>IL</sub> = 0.8V
I <sub>LI1</sub>	Low Input Load Current (RESET)			0.2	mA	V <sub>IL</sub> = 0.8V
I <sub>IH</sub>	Input High Leakage Current (Pins 24, 27-38)			100	μA	V <sub>IN</sub> = V <sub>CC</sub>
C <sub>IN</sub>	Input Capacitance			10	pF	
C <sub>I/O</sub>	I/O Capacitance			20	pF	



**A.C. CHARACTERISTICS** ( $T_A = 0^\circ\text{C}$  to  $70^\circ\text{C}$ ,  $V_{CC} = V_{DD} = +5V \pm 10\%$ ,  $V_{SS} = 0V$ )

**DBB READ**

Symbol	Parameter	Min.	Max.	Unit	Test Conditions
$t_{AR}$	$\overline{CS}$ , $A_0$ Setup to $\overline{RD} \downarrow$	0		ns	
$t_{RA}$	$\overline{CS}$ , $A_0$ Hold After $\overline{RD} \uparrow$	0		ns	
$t_{RR}$	$\overline{RD}$ Pulse Width	250		ns	
$t_{AD}$	$\overline{CS}$ , $A_0$ to Data Out Delay		225	ns	$C_L = 150\text{ pF}$
$t_{RD}$	$\overline{RD} \downarrow$ to Data Out Delay		225	ns	$C_L = 150\text{ pF}$
$t_{DF}$	$\overline{RD} \uparrow$ to Data Float Delay		100	ns	
$t_{CY}$	Cycle Time	2.5	15	$\mu\text{s}$	6 MHz Crystal

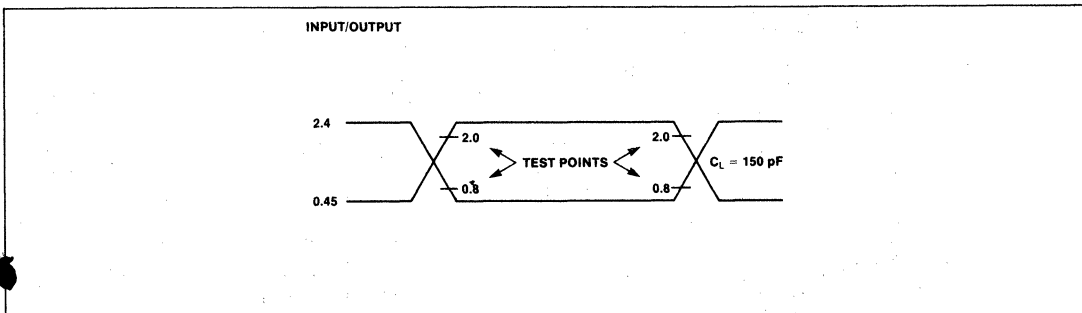
**DBB WRITE**

Symbol	Parameter	Min.	Max.	Unit	Test Conditions
$t_{AW}$	$\overline{CS}$ , $A_0$ Setup to $\overline{WR} \downarrow$	0		ns	
$t_{WA}$	$\overline{CS}$ , $A_0$ Hold After $\overline{WR} \uparrow$	0		ns	
$t_{WW}$	$\overline{WR}$ Pulse Width	250		ns	
$t_{DW}$	Data Setup to $\overline{WR} \uparrow$	150		ns	
$t_{WD}$	Data Hold to $\overline{WR} \uparrow$	0		ns	

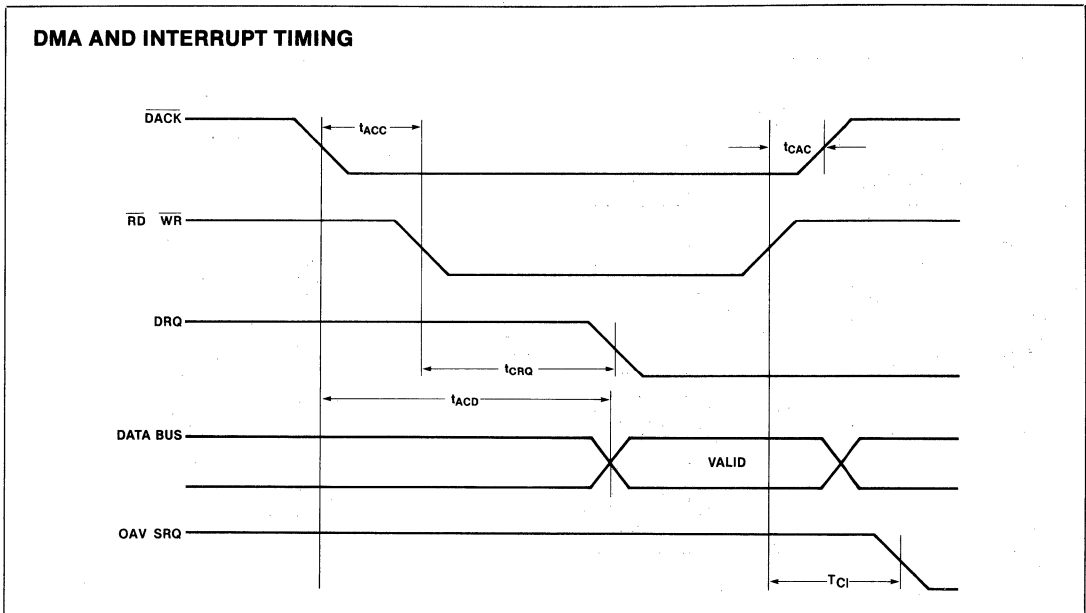
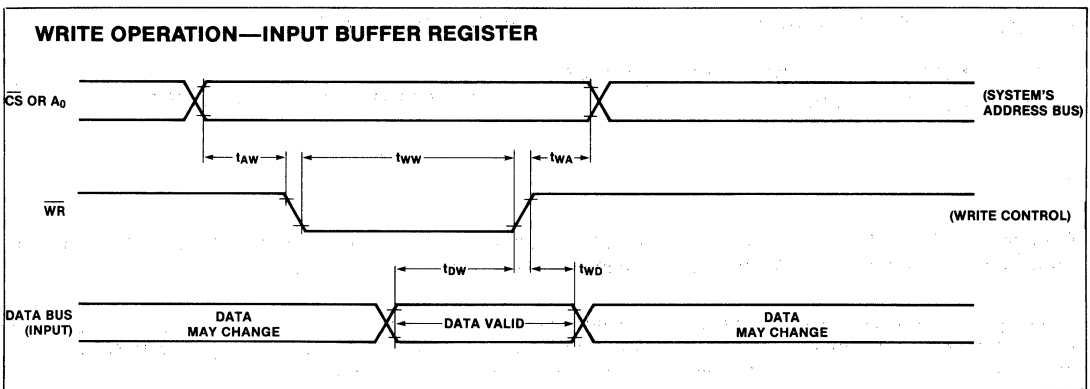
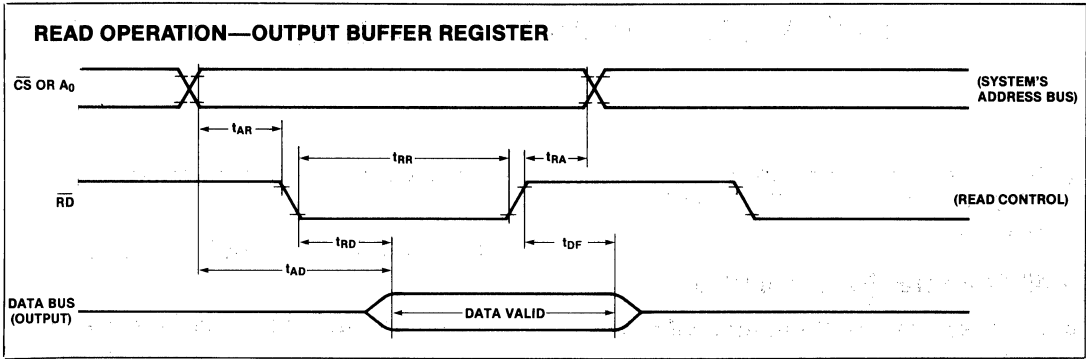
**DMA AND INTERRUPT TIMING**

Symbol	Parameter	Min.	Max.	Unit	Test Conditions
$t_{ACC}$	$\overline{DACK}$ Setup to Control	0		ns	
$t_{CAC}$	$\overline{DACK}$ Hold After Control	0		ns	
$t_{ACD}$	$\overline{DACK}$ to Data Valid		225	ns	$C_L = 150\text{ pF}$
$t_{CRQ}$	Control L.E. to $\overline{DRQ}$ T.E.		200	ns	
$t_{CI}$	Control T.E. to Interrupt T.E.		$t_{CY} + 500$	ns	

**A.C. TESTING INPUT, OUTPUT WAVEFORM**



WAVEFORMS





# 8295 DOT MATRIX PRINTER CONTROLLER

- Interfaces Dot Matrix Printers to MCS-48™, MCS-80/85™, MCS-86™ Systems
- 40 Character Buffer On Chip
- Serial or Parallel Communication with Host
- DMA Transfer Capability
- Programmable Character Density (10 or 12 Characters/Inch)
- Programmable Print Intensity
- Single or Double Width Printing
- Programmable Multiple Line Feeds
- 3 Tabulations
- 2 General Purpose Outputs

The Intel® 8295 Dot Matrix Printer Controller provides an interface for microprocessors to the LRC 7040 Series dot matrix impact printers. It may also be used as an interface to other similar printers.

The chip may be used in a serial or parallel communication mode with the host processor. In parallel mode, data transfers are based on polling, interrupts, or DMA. Furthermore, it provides internal buffering of up to 40 characters and contains a 7 × 7 matrix character generator accommodating 64 ASCII characters.

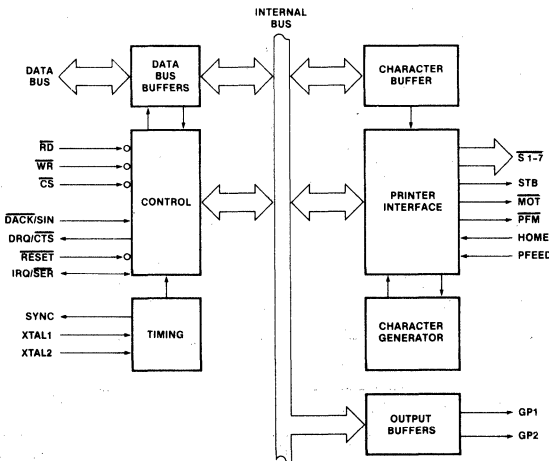


Figure 1. Block Diagram

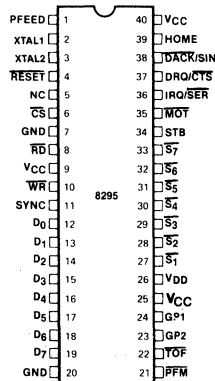


Figure 2. Pin Configuration

Table 1. Pin Description

Symbol	Pin No.	Type	Name and Function
PFEED	1	I	<b>Paper Feed:</b> Paper feed input switch.
XTAL1 XTAL2	2 3	I	<b>Crystal:</b> Inputs for a crystal to set internal oscillator frequency. For proper operation use 6 MHz crystal.
RESET	4	I	<b>Reset:</b> Reset input, active low. After reset the 8295 will be set for 12 characters/inch single width printing, solenoid strobe at 320 msec.
NC	5		<b>No Connection:</b> No connection or tied high.
$\overline{CS}$	6	I	<b>Chip Select:</b> Chip select input used to enable the RD and WR inputs except during DMA.
GND	7		<b>Ground:</b> This pin must be tied to ground.
$\overline{RD}$	8	I	<b>Read:</b> Read input which enables the master CPU to read data and status. In the serial mode this pin must be tied to $V_{CC}$ .
$V_{CC}$	9		<b>Power:</b> +5 volt power input: +5V $\pm$ 10%.
$\overline{WR}$	10	I	<b>Write:</b> Write input which enables the master CPU to write data and commands to the 8295. In the serial mode this pin must be tied to $V_{SS}$ .
SYNC	11	O	<b>Sync:</b> 2.5 $\mu$ s clock output. Can be used as a strobe for external circuitry.
D <sub>0</sub> D <sub>1</sub> D <sub>2</sub> D <sub>3</sub> D <sub>4</sub> D <sub>5</sub> D <sub>6</sub> D <sub>7</sub>	12 13 14 15 16 17 18 19	I/O	<b>Data Bus:</b> Three-state bidirectional data bus buffer lines used to interface the 8295 to the host processor in the parallel mode. In the serial mode D <sub>0</sub> —D <sub>2</sub> sets up the baud rate.
GND	20		<b>Ground:</b> This pin must be tied to ground.
$V_{CC}$	40		<b>Power:</b> +5 volt power input: +5 $\pm$ 10%.

Symbol	Pin No.	Type	Name and Function
HOME	39	I	<b>Home:</b> Home input switch, used by the 8295 to detect that the print head is in the home position.
DACK/SIN	38	I	<b>DMA Acknowledge/Serial Input:</b> In the parallel mode used as DMA acknowledgment; in the serial mode, used as input for data.
DRQ/CTS	37	O	<b>DMA Request/Clear to Send:</b> In the parallel mode used as DMA request output pin to indicate to the 8257 that a DMA transfer is requested; in the serial mode used as clear-to-send signal.
IRQ/SER	36	O	<b>Interrupt Request/Serial Mode:</b> In parallel mode it is an interrupt request input to the master CPU; in serial mode it should be strapped to $V_{SS}$ .
$\overline{MOT}$	35	O	<b>Motor:</b> Main motor drive, active low.
STB	34	O	<b>Solenoid Strobe:</b> Solenoid strobe output. Used to determine duration of solenoids activation.
$\overline{S_7}$ $\overline{S_6}$ $\overline{S_5}$ $\overline{S_4}$ $\overline{S_3}$ $\overline{S_2}$ $\overline{S_1}$	33 32 31 30 29 28 27	O	<b>Solenoid:</b> Solenoid drive outputs; active low.
$V_{DD}$	26		<b>Power:</b> +5V power input (+5V $\pm$ 10%). Low power standby pin.
$V_{CC}$	25		<b>Power:</b> Tied high.
GP1 GP2	24 23	O O	<b>General Purpose:</b> General purpose output pins.
$\overline{TOF}$	22	I	<b>Top of Form:</b> Top of form input, used to sense top of form signal for type T printer.
PFM	21	O	<b>Paper Feed Motor Drive:</b> Paper feed motor drive, active low.

## FUNCTIONAL DESCRIPTION

The 8295 interfaces microcomputers to the LRC 7040 Series dot matrix impact printers, and to other similar printers. It provides internal buffering of up to 40 characters. Printing begins automatically when the buffer is full or when a carriage return character is received. It provides a modified 7x7 matrix character generator. The character set includes 64 ASCII characters.

Communication between the 8295 and the host processor can be implemented in either a serial or parallel mode. The parallel mode allows for character transfers into the buffer via DMA cycles. The serial mode features selectable data rates from 110 to 4800 baud.

The 8295 also offers two general purpose output pins which can be set or cleared by the host processor. They can be used with various printers to implement such functions as ribbon color selection, enabling form release solenoid, and reverse document feed.

## COMMAND SUMMARY

Hex Code	Description	Hex Code	Description
00	Set GP1. This command brings the GP1 pin to a logic high state. After power on it is automatically set high.	09	Tab character.
01	Set GP2. Same as the above but for GP2.	0A	Line feed.
02	Clear GP1. Sets GP1 pin to logic low state, inverse of command 00.	0B	Multiple Line Feed; must be followed by a byte specifying the number of line feeds.
03	Clear GP2. Same as above but for GP2. Inverse command 01.	0C	Top of Form. Enables the line feed output until the Top of Form input is activated.
04	Software Reset. This is a pacify command. This command is not effective immediately after commands requiring a parameter, as the Reset command will be interpreted as a parameter.	0D	Carriage Return. Signifies end of a line and enables the printer to start printing.
05	Print 10 characters/in. density.	0E	Set Tab #1, followed by tab position byte.
06	Print 12 characters/in. density.	0F	Set Tab #2, followed by tab position byte. Should be greater than Tab #1.
07	Print double width characters. This command prints characters at twice the normal width, that is, at either 17 or 20 characters per line.	10	Set Tab #3, followed by tab position byte. Should be greater than Tab #2.
08	Enable DMA mode; must be followed by two bytes specifying the number of data characters to be fetched. Least significant byte accepted first.	11	Print Head Home on Right. On some printers the print head home position is on the right. This command would enable normal left to right printing with such printers.
		12	Set Strobe Width; must be followed by strobe width selection byte. This command adjusts the duration of the strobe activation.

## PROGRAMMABLE PRINTING OPTIONS

### CHARACTER DENSITY

The character density is programmable at 10 or 12 characters/inch (32 or 40 characters/line). The 8295 is automatically set to 12 characters/inch at power-up. Invoking the Print Double-Width command halves the character density (5 or 6 characters/inch). The 10 char/in or 12 char/in command must be re-issued to cancel the Double-Width mode. Different character density modes may not be mixed within a single line of printing.

### PRINT INTENSITY

The intensity of the printed characters is determined by the amount of time during which the solenoid is on. This on-time is programmable via the Set Strobe-Width command. A byte following this command sets the solenoid on-time according to Table 2. Note that only the three least significant bits of this byte are important.

Table 2. Solenoid On-Time

D7—D3	D2	D1	D0	Solenoid On (microsec)
x	0	0	0	200
x	0	0	1	240
x	0	1	0	280
x	0	1	1	320
x	1	0	0	360
x	1	0	1	400
x	1	1	0	440
x	1	1	1	480

### TABULATIONS

Up to three tabulation positions may be specified with the 8295. The column position of each tabulation is selected by issuing the Set Tab commands, each fol-

lowed by a byte specifying the column. The tab positions will then remain valid until new Set Tab commands are issued.

Sending a tab character (09H) will automatically fill the character buffer with blanks up to the next tab position. The character sent immediately after the tab character will thus be stored and printed at that position.

### CPU TO 8295 INTERFACE

Communication between the CPU and the 8295 may take place in either a serial or parallel mode. However, the selection of modes is inherent in the system hardware; it is not software programmable. Thus, the two modes cannot be mixed in a single 8295 application.

### PARALLEL INTERFACE

Two internal registers on the 8295 are addressable by the CPU: one for input, one for output. The following table describes how these registers are accessed.

$\overline{RD}$	$\overline{WR}$	$\overline{CS}$	Register
1	0	0	Input Data Register
0	1	0	Output Status Register

**Input Data Register**—Data written to this register is interpreted in one of two ways, depending on how the data is coded.

1. A command to be executed (0XH or 1XH).
2. A character to be stored in the character buffer for printing (2XH, 3XH, 4XH, or 5XH). See the character set, Table 2.

**Output Status Register**—8295 status is available in this register at all times.

STATUS BIT:	7	6	5	4	3	2	1	0
FUNCTION:	x	x	PA	DE	x	x	IBF	x

**PA**—Parameter Required; PA = 1 indicates that a command requiring a parameter has been received. After the necessary parameters have been received by the 8295, the PA flag is cleared.

**DE**—DMA Enabled; DE = 1 whenever the 8295 is in DMA mode. Upon completion of the required DMA transfers, the DE flag is cleared.

**IBF**—Input Buffer Full; IBF = 1 whenever data is written to the Input Data Register. No data should be written to the 8295 when IBF = 1.

A flow chart describing communication with the 8295 is shown in Figure 3.

The interrupt request output (IRQ, Pin 36) is available on the 8295 for interrupt driven systems. This output is asserted true whenever the 8295 is ready to receive data. To improve bus efficiency and CPU overhead, data may be transferred from main memory to the 8295 via DMA cycles. Sending the Enable DMA command (08H) activates the DMA channel of the 8295. This command must be followed by two bytes specifying the length of the data string to be transferred (least significant byte first). The 8295 will then assert the required DMA requests to

the 8257 DMA controller without further CPU intervention. Figure 4 shows a block diagram of the 8295 in DMA mode.

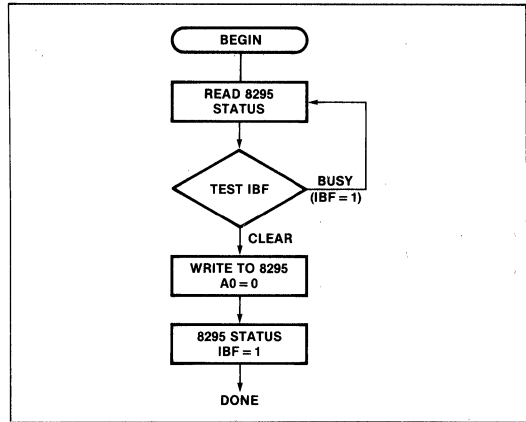


Figure 3. Host to 8295 Protocol Flowchart

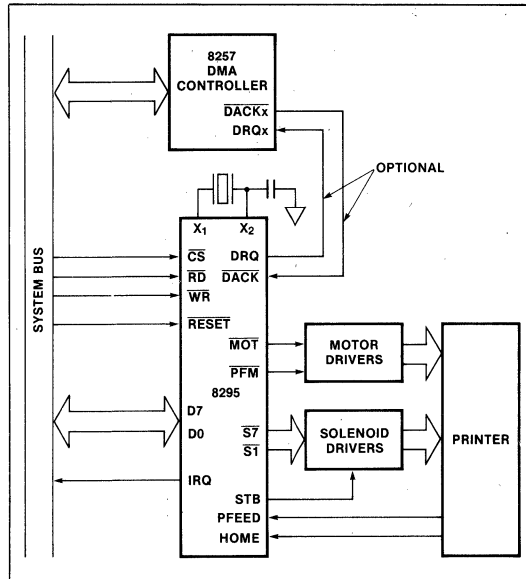


Figure 4. Parallel System Interface

Data transferred in the DMA mode may be either commands or characters or a mixture of both. The procedure is as follows:

1. Set up the 8257 DMA controller channel by sending a starting address and a block length.
2. Set up the 8295 by issuing the "Enable DMA" command (08H) followed by two bytes specifying the block length (least significant byte first).

The DMA enabled flag (DE) will be true until the assigned data transfer is completed. Upon completion of the transfer, the flag is cleared and the interrupt request (IRQ) signal is asserted. The 8295 then returns to the non-DMA mode of operation.

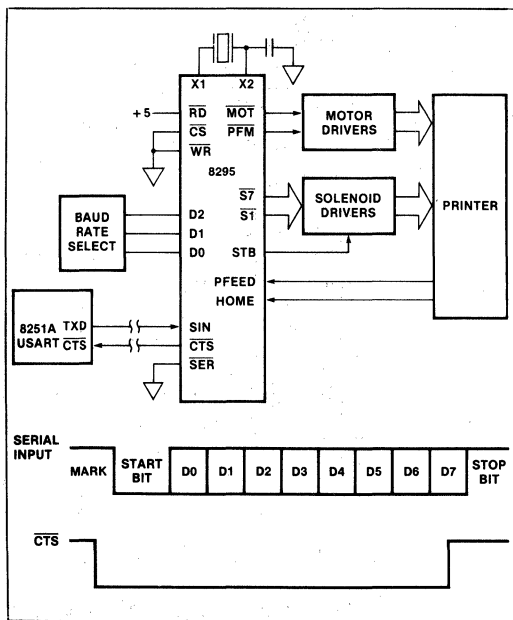
**SERIAL INTERFACE**

The 8295 may be hardware programmed to operate in a serial mode of communication. By connecting the IRQ/SER pin (pin 36) to logic zero, the serial mode is enabled immediately upon power-up. The serial mode baud rate is also hardware programmable; by strapping pins 14, 13, and 12 according to Table 3, the rate is selected. CS, RD, and WR must be strapped as shown in Figure 5.

**Table 3. Serial Baud Rate**

Pin 14	Pin 13	Pin 12	Baud Rate
0	0	0	110
0	0	1	150
0	1	0	300
0	1	1	600
1	0	0	1200
1	0	1	2400
1	1	0	4800
1	1	1	4800

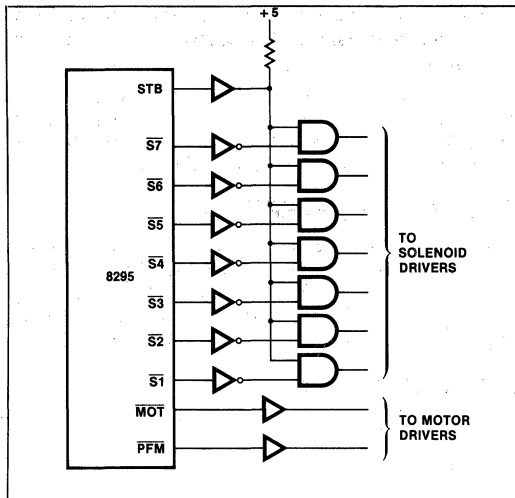
The serial data format is shown in Figure 5. The CPU should wait for a clear to send signal (CTS) from the 8295 before sending data.



**Figure 5. Serial Interface to UART (8251A)**

**8295 TO PRINTER INTERFACE**

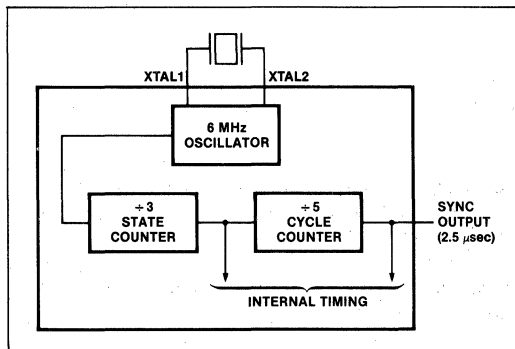
The strobe output signal of the 8295 determines the duration of the solenoid outputs, which hold the data to the printer. These solenoid outputs cannot drive the printer solenoids directly. They should be buffered through solenoid drivers as shown in Figure 6. Recommended solenoid and motor driver circuits may be found in the printer manufacturer's interface guide.



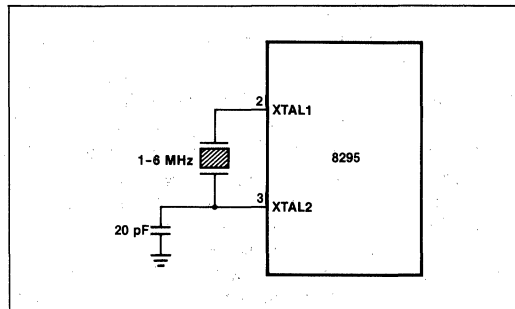
**Figure 6. 8295 To Printer Solenoid Interface**

**OSCILLATOR AND TIMING CIRCUITS**

The 8295's internal timing generation is controlled by a self-contained oscillator and timing circuit. A 6 MHz crystal is used to derive the basic oscillator frequency. The resident timing circuit consists of an oscillator, a state counter and a cycle counter as illustrated in Figure 7. The recommended crystal connection is shown in Figure 8.



**Figure 7. Oscillator Configuration**



**Figure 8. Recommended Crystal Connection**

**8295 CHARACTER SET**

Hex Code	Print Char.	Hex Code	Print Char.	Hex Code	Print Char.	Hex Code	Print Char.
20	space	30	0	40	@	50	P
21	!	31	1	41	A	51	Q
22	"	32	2	42	B	52	R
23	#	33	3	43	C	53	S
24	\$	34	4	44	D	54	T
25	%	35	5	45	E	55	U
26	&	36	6	46	F	56	V
27	,	37	7	47	G	57	W
28	(	38	8	48	H	58	X
29	)	39	9	49	I	59	Y
2A	*	3A	:	5A	J	5A	Z
2B	+	3B	;	4B	K	5B	[
2C	,	3C	<	4C	L	5C	\
2D	-	3D	=	4D	M	5D	]
2E	.	3E	>	4E	N	5E	↑
2F	/	3F	?	4F	O	5F	—

**ABSOLUTE MAXIMUM RATINGS\***

Ambient Temperature Under Bias ..... 0°C to 70°C  
 Storage Temperature ..... - 65° to + 150°C  
 Voltage on Any Pin With  
 Respect to Ground ..... -0.5V to +7V  
 Power Dissipation ..... 1.5 Watt

*\*NOTICE: Stresses above those listed under "Absolute Maximum Ratings" may cause permanent damage to the device. This is a stress rating only and functional operation of the device at these or any other conditions above those indicated in the operational sections of this specification is not implied. Exposure to absolute maximum rating conditions for extended periods may affect device reliability.*

**D.C. AND OPERATING CHARACTERISTICS** ( $T_A = 0^\circ\text{C to } 70^\circ\text{C}$ ,  $V_{CC} = V_{DD} = +5V \pm 10\%$ ,  $V_{SS} = 0V$ )

Symbol	Parameter	Limits			Unit	Test Conditions
		Min.	Typ.	Max.		
$V_{IL}$	Input Low Voltage (All Except $X_1, X_2, \text{RESET}$ )	-0.5		0.8	V	
$V_{IL1}$	Input Low Voltage ( $X_1, X_2, \text{RESET}$ )	-0.5		0.6	V	
$V_{IH}$	Input High Voltage (All Except $X_1, X_2, \text{RESET}$ )	2.2		$V_{CC}$	V	
$V_{IH1}$	Input High Voltage ( $X_1, X_2, \text{RESET}$ )	3.8		$V_{CC}$	V	
$V_{OL}$	Output Low Voltage ( $D_0-D_7$ )			0.45	V	$I_{OL} = 2.0 \text{ mA}$
$V_{OL1}$	Output Low Voltage (All Other Outputs)			0.45	V	$I_{OL} = 1.6 \text{ mA}$
$V_{OH}$	Output High Voltage ( $D_0-D_7$ )	2.4			V	$I_{OH} = -400 \mu\text{A}$
$V_{OH1}$	Output High Voltage (All Other Outputs)	2.4			V	$I_{OH} = -50 \mu\text{A}$
$I_{IL}$	Input Leakage Current ( $R_D, W_R, CS, \bar{A}_0$ )			$\pm 10$	$\mu\text{A}$	$V_{SS} \leq V_{IN} \leq V_{CC}$
$I_{OZ}$	Output Leakage Current ( $D_0-D_7, \text{High Z State}$ )			$\pm 10$	$\mu\text{A}$	$V_{SS} + 0.45 \leq V_{OUT} \leq V_{CC}$
$I_{DD}$	$V_{DD}$ Supply Current		5	15	mA	
$I_{DD} + I_{CC}$	Total Supply Current		60	125	mA	
$I_{LI}$	Low Input Load Current (Pins 24, 27-38)			0.5	mA	$V_{IL} = 0.8V$
$I_{LI1}$	Low Input Load Current (RESET)			0.2	mA	$V_{IL} = 0.8V$
$I_{IH}$	Input High Leakage Current (Pins 22, 38)			100	$\mu\text{A}$	$V_{IN} = V_{CC}$
$C_{IN}$	Input Capacitance			10	pF	
$C_{I/O}$	I/O Capacitance			20	pF	



**A.C. CHARACTERISTICS** ( $T_A = 0^\circ\text{C}$  to  $70^\circ\text{C}$ ,  $V_{CC} = V_{DD} = +5\text{V} \pm 10\%$ ,  $V_{SS} = 0\text{V}$ )

**DBB READ**

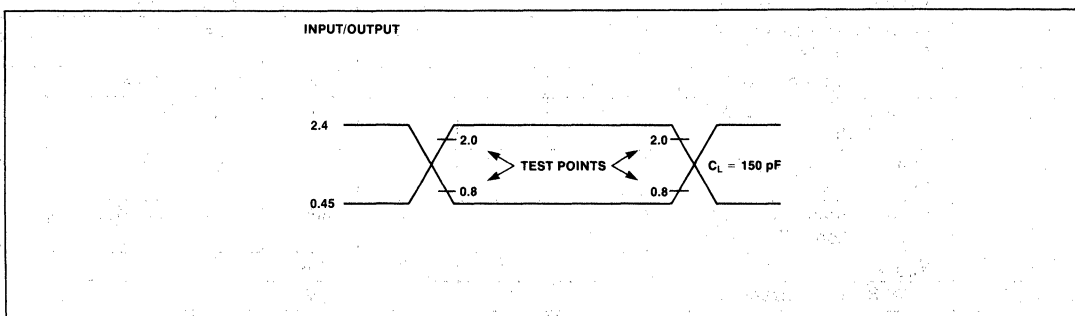
Symbol	Parameter	Min.	Max.	Unit	Test Conditions
$t_{AR}$	$\overline{CS}$ , $A_0$ Setup to $\overline{RD} \downarrow$	0		ns	
$t_{RA}$	$\overline{CS}$ , $A_0$ Hold After $\overline{RD} \uparrow$	0		ns	
$t_{RR}$	$\overline{RD}$ Pulse Width	250		ns	
$t_{AD}$	$\overline{CS}$ , $A_0$ to Data Out Delay		225	ns	$C_L = 150 \text{ pF}$
$t_{RD}$	$\overline{RD} \downarrow$ to Data Out Delay		225	ns	$C_L = 150 \text{ pF}$
$t_{DF}$	$\overline{RD} \uparrow$ to Data Float Delay		100	ns	
$t_{CY}$	Cycle Time	2.5	15	$\mu\text{s}$	

**DBB WRITE**

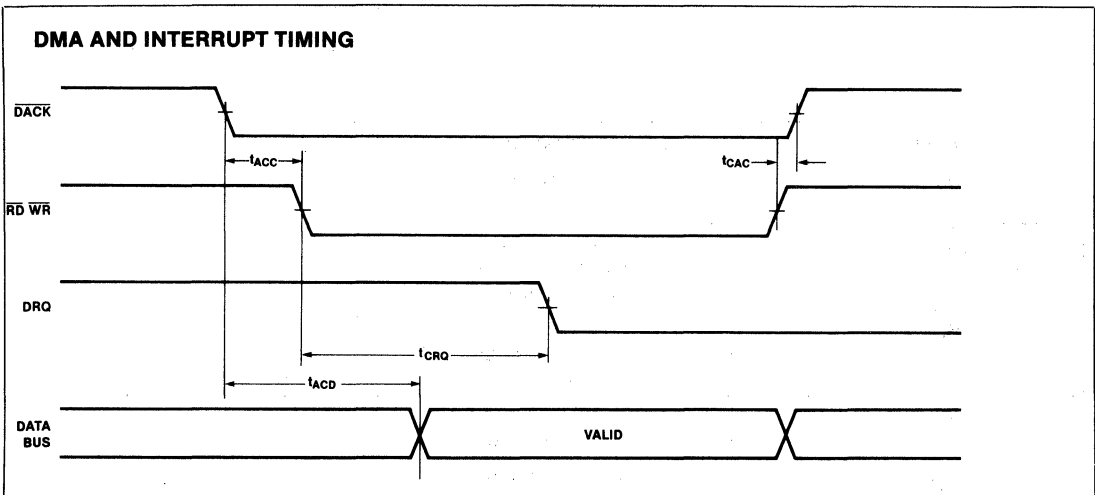
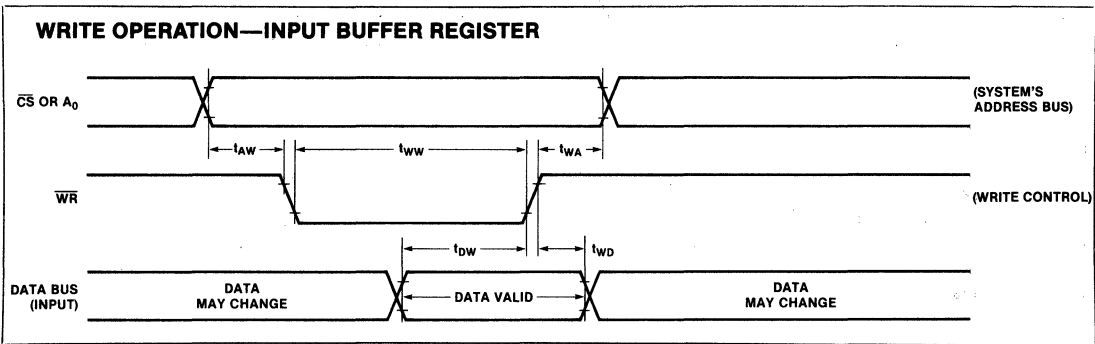
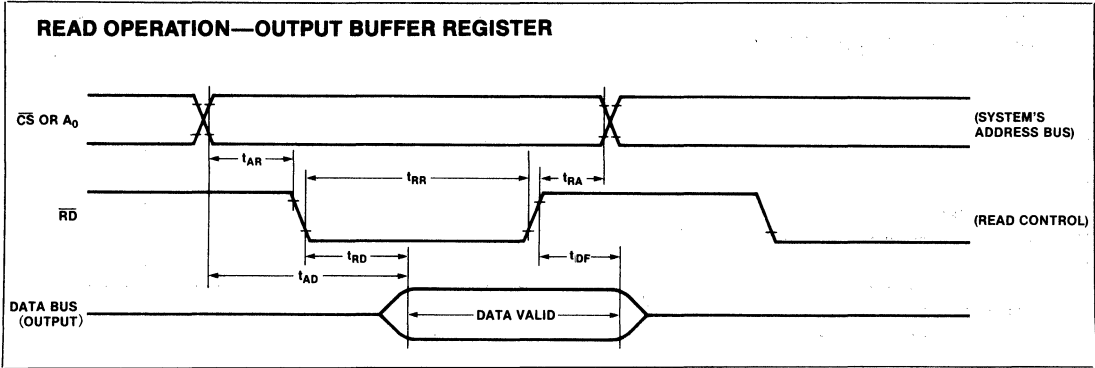
Symbol	Parameter	Min.	Max.	Unit	Test Conditions
$t_{AW}$	$\overline{CS}$ , $A_0$ Setup to $\overline{WR} \downarrow$	0		ns	
$t_{WA}$	$\overline{CS}$ , $A_0$ Hold After $\overline{WR} \uparrow$	0		ns	
$t_{WW}$	$\overline{WR}$ Pulse Width	250		ns	
$t_{DW}$	Data Setup to $\overline{WR} \uparrow$	150		ns	
$t_{WD}$	Data Hold to $\overline{WR} \uparrow$	0		ns	

**DMA AND INTERRUPT TIMING**

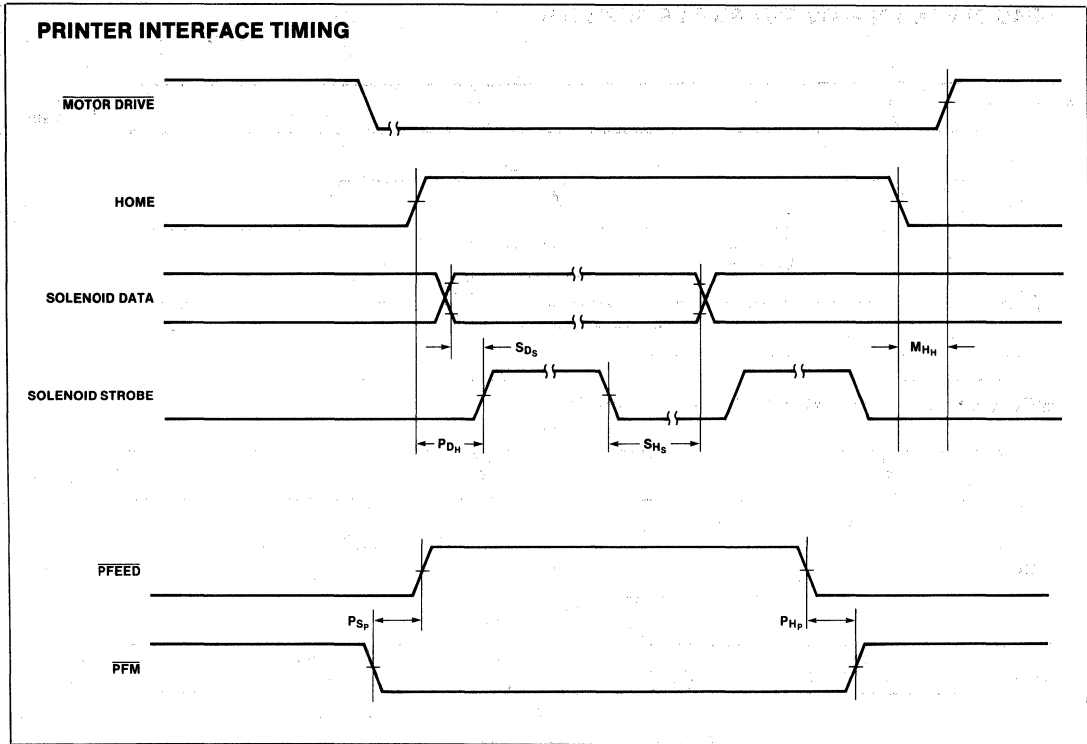
Symbol	Parameter	Min.	Max.	Unit	Test Conditions
$t_{ACC}$	$\overline{DACK}$ Setup to Control	0		ns	
$t_{CAC}$	$\overline{DACK}$ Hold After Control	0		ns	
$t_{CRQ}$	$\overline{WR}$ to $\overline{DRQ}$ Cleared		200	ns	
$t_{ACD}$	$\overline{DACK}$ to Data Valid		225	ns	$C_L = 150 \text{ pF}$

**A.C. TESTING INPUT, OUTPUT WAVEFORM**


WAVEFORMS



WAVEFORMS (Continued)



Symbol	Parameter	Typical
$P_{DH}$	Print delay from home inactive	1.8 ms
$S_{DS}$	Solenoid data setup time before strobe active	25 $\mu$ s
$S_{HS}$	Solenoid data hold after strobe inactive	>1 ms
$M_{HA}$	Motor hold time after home active	3.2 ms
$P_{SP}$	PFEED setup time after PFM active	58 ms
$P_{HP}$	PFM hold time after PFEED active	9.75 ms

---

# *Memory Controllers*

---

## 8202A DYNAMIC RAM CONTROLLER

- Provides All Signals Necessary to Control 2104A, 2117, or 2118 Dynamic Memories
- Directly Addresses and Drives Up to 64K Bytes Without External Drivers
- Provides Address Multiplexing and Strobes
- Provides a Refresh Timer and a Refresh Counter
- Refresh Cycles May be Internally or Externally Requested
- Provides Transparent Refresh Capability
- Fully Compatible with Intel® 8080A, 8085A, iAPX 88, and iAPX 86 Family Microprocessors
- Decodes CPU Status for Advanced Read Capability
- Provides System Acknowledge and Transfer Acknowledge Signals
- Internal Clock Capability with the 8202A-1

The Intel® 8202A is a Dynamic Ram System Controller designed to provide all signals necessary to use 2104A, 2117, or 2118 Dynamic RAMs in microcomputer systems. The 8202A provides multiplexed addresses and address strobes, as well as refresh/access arbitration. The 8202A-1 supports an internal crystal oscillator.

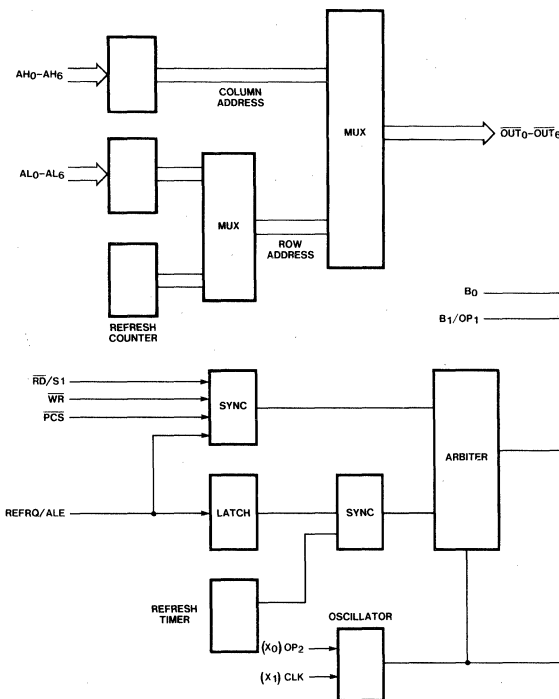


Figure 1. 8202A Block Diagram

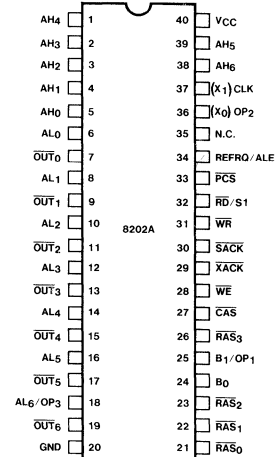


Figure 2. Pin Configuration

Table 1. Pin Descriptions

Symbol	Pin No.	Type	Name and Function
AL <sub>0</sub> AL <sub>1</sub> AL <sub>2</sub> AL <sub>3</sub> AL <sub>4</sub> AL <sub>5</sub> AL <sub>6</sub> /OP <sub>3</sub>	6 8 10 12 14 16 18	I I I I I I I	<b>Address Low:</b> CPU address inputs used to generate memory row address. AL <sub>6</sub> /OP <sub>3</sub> used to select 4K RAM mode.
AH <sub>0</sub> AH <sub>1</sub> AH <sub>2</sub> AH <sub>3</sub> AH <sub>4</sub> AH <sub>5</sub> AH <sub>6</sub>	5 4 3 2 1 39 38	I I I I I I I	<b>Address High:</b> CPU address inputs used to generate memory column address.
BO B <sub>1</sub> /OP <sub>1</sub>	24 25	I I	<b>Bank Select Inputs:</b> Used to gate the appropriate RAS <sub>0</sub> -RAS <sub>3</sub> output for a memory cycle. B <sub>1</sub> /OP <sub>1</sub> option used to select the Advanced Read Mode.
PCS	33	I	<b>Protected Chip Select:</b> Used to enable the memory read and write inputs. Once a cycle is started, it will not abort even if PCS goes inactive before cycle completion.
WR	31	I	<b>Memory Write Request.</b>
RD/S1	32	I	<b>Memory Read Request:</b> S1 function used in Advanced Read mode selected by OP <sub>1</sub> (pin 25).
REFRQ/ ALE	34	I	<b>External Refresh Request:</b> ALE function used in Advanced Read mode, selected by OP <sub>1</sub> (pin 25).
OUT <sub>0</sub> OUT <sub>1</sub> OUT <sub>2</sub> OUT <sub>3</sub> OUT <sub>4</sub> OUT <sub>5</sub> OUT <sub>6</sub>	7 9 11 13 15 17 19	O O O O O O O	<b>Output of the Multiplexer:</b> These outputs are designed to drive the addresses of the Dynamic RAM array. For 4K RAM operation, OUT <sub>6</sub> is designed to drive the 2104A CS input. (Note that the OUT <sub>0-6</sub> pins do not require inverters or drivers for proper operation.)
WE	28	O	<b>Write Enable:</b> Drives the Write Enable inputs of the Dynamic RAM array.
CAS	27	O	<b>Column Address Strobe:</b> This output is used to latch the Column Address into the Dynamic RAM array.

Symbol	Pin No.	Type	Name and Function
RAS <sub>0</sub> RAS <sub>1</sub> RAS <sub>2</sub> RAS <sub>3</sub>	21 22 23 26	O O O O	<b>Row Address Strobe:</b> Used to latch the Row Address into the bank of dynamic RAMs, selected by the 8202A Bank Select pins (B <sub>0</sub> , B <sub>1</sub> /OP <sub>1</sub> ).
XACK	29	O	<b>Transfer Acknowledge:</b> This output is a strobe indicating valid data during a read cycle or data written during a write cycle. XACK can be used to latch valid data from the RAM array.
SACK	30	O	<b>System Acknowledge:</b> This output indicates the beginning of a memory access cycle. It can be used as an advanced transfer acknowledge to eliminate wait states. (Note: If a memory access request is made during a refresh cycle, SACK is delayed until XACK in the memory access cycle).
(X <sub>0</sub> ) OP <sub>2</sub> (X <sub>1</sub> ) CLK	36 37	I/O I/O	<b>Oscillator Inputs:</b> These inputs are designed for a quartz crystal to control the frequency of the oscillator. If X <sub>0</sub> /OP <sub>2</sub> is connected to a 1KΩ resistor pulled to +12V then X <sub>1</sub> /CLK becomes a TTL input for an external clock.
N.C.	35		Reserved for future use.
VCC	40		<b>Power Supply:</b> +5V.
GND	20		<b>Ground.</b>

NOTE: Crystal mode for the 8202A-1 only.

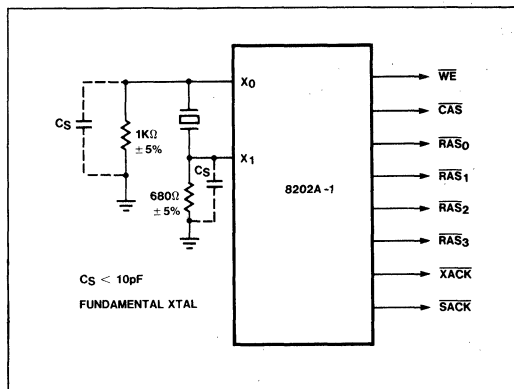


Figure 3. Crystal Operation for the 8202A-1

## Functional Description

The 8202A provides a complete dynamic RAM controller for microprocessor systems as well as expansion memory boards. All of the necessary control signals are provided for 2104A, 2117, and 2118 dynamic RAM's.

All 8202A timing is generated from a single reference clock. This clock is provided via an external oscillator or an on chip crystal oscillator. All output signal transitions are synchronous with respect to this clock reference, except for the CPU handshake signals  $\overline{SACK}$  and  $\overline{XACK}$  (trailing edge).

CPU memory requests normally use the  $\overline{RD}$  and  $\overline{WR}$  inputs. The advanced READ mode allows ALE and S1 to be used in place of the  $\overline{RD}$  input.

Failsafe refresh is provided via an internal refresh timer which generates internal refresh requests. Refresh requests can also be generated via the REFRQ input.

An on-chip synchronizer / arbiter prevents memory and refresh requests from affecting a cycle in progress. The READ, WRITE, and external REFRESH requests may be asynchronous to the 8202A clock; on-chip logic will synchronize the requests, and the arbiter will decide if the requests should be delayed, pending completion of a cycle in progress.

## Option Selection

The 8202A has three strapping options. When  $OP_1$  is selected (16K mode only), pin 32 changes from a  $\overline{RD}$  input to an S1 input, and pin 34 changes from a REFREQ input to an ALE input. See "Refresh Cycles" and "Read Cycles" for more detail.  $OP_1$  is selected by tying pin 25 to +12V through a 5.1K ohm resistor.

When  $OP_2$  is selected, by connecting pin 36 to +12V through a 1K ohm resistor, pin 37 changes from a crystal input ( $X_1$ ) to the CLK input for an external TTL clock.

$OP_3$  is selected by connecting Pin 18 to +12V through a 5.1K ohm resistor. The 8202A will change its internal refresh timer from 128-row refresh (2118, 2117) to 64-row refresh (2104A).

## Refresh Timer

The refresh timer is used to monitor the time since the last refresh cycle occurred. When the appropriate amount of time has elapsed, the refresh timer will request a refresh cycle. External refresh requests will reset the refresh timer.

## Refresh Counter

The refresh counter is used to sequentially refresh all of the memory's rows. The 8-bit counter is incremented after every refresh cycle.

## Address Multiplexer

The address multiplexer takes the address inputs and the refresh counter outputs, and gates them onto the address outputs at the appropriate time. The address outputs, in conjunction with the  $\overline{RAS}$  and  $\overline{CAS}$  outputs, determine the address used by the dynamic RAMs for read, write, and refresh cycles. During the first part of a read or write cycle,  $AL_0$ - $AL_6$  are gated to  $\overline{OUT_0}$ - $\overline{OUT_6}$ , then  $AH_0$ - $AH_6$  are gated to the address outputs.

During a refresh cycle, the refresh counter is gated onto the address outputs. All refresh cycles are RAS-only refresh ( $\overline{CAS}$  inactive,  $\overline{RAS}$  active).

To minimize buffer delay, the information on the address outputs is inverted from that on the address inputs.

$\overline{OUT_0}$ - $\overline{OUT_6}$  do not need inverters or buffers unless additional drive is required.

## Synchronizer / Arbiter

The 8202A has three inputs, REFRQ/ALE (pin 34),  $\overline{RD}$  (pin 32) and  $\overline{WR}$  (pin 31). The  $\overline{RD}$  and  $\overline{WR}$  inputs allow an external CPU to request a memory read or write cycle, respectively. The REFRQ/ALE allows refresh requests to be requested external to the 8202A.

All three of these inputs may be asynchronous with respect to the 8202A's clock. The arbiter will resolve conflicts between refresh and memory requests, for both pending cycles and cycles in progress. Read and write requests will be given priority over refresh requests.

## System Operation

The 8202A is always in one of the following states:

- a) IDLE
- b) TEST Cycle
- c) REFRESH Cycle
- d) READ Cycle
- e) WRITE Cycle

The 8202A is normally in the IDLE state. Whenever one of the other cycles is requested, the 8202A will leave the IDLE state to perform the desired cycle. If no other cycles are pending, the 8202A will return to the IDLE state.

Description	Pin #	Normal Function	Option Function
B1/ $OP_1$	25	Bank (RAS) Select	Advanced-Read Mode
$X_0$ / $OP_2$	36	Crystal Oscillator (8202A-1)	External Oscillator
$AL_6$ / $OP_3$	18	Address Input	64-ROW Refresh

Figure 4. 8202A Option Selection

**Test Cycle**

The TEST Cycle is used to check operation of several 8202A internal functions. TEST cycles are requested by activating the RD and WR inputs, independent of PCS. The TEST Cycle will reset the refresh address counter and perform a WRITE Cycle. The TEST Cycle should not be used in normal system operation, since it would affect the dynamic RAM refresh.

**Refresh Cycles**

The 8202A has two ways of providing dynamic RAM refresh:

- 1) Internal (failsafe) refresh
- 2) External (hidden) refresh

Both types of 8202A refresh cycles activate all of the  $\overline{\text{RAS}}$  outputs, while  $\overline{\text{CAS}}$ ,  $\overline{\text{WE}}$ ,  $\overline{\text{SACK}}$ , and  $\overline{\text{XACK}}$  remain inactive.

Internal refresh is generated by the on-chip refresh timer. The timer uses the 8202A clock to ensure that refresh of all rows of the dynamic RAM occurs every 2 milliseconds. If REFRQ is inactive, the refresh timer will request a refresh cycle every 10-16 microseconds.

External refresh is requested via the REFRQ input (pin 34). External refresh control is not available when the Advanced-Read mode is selected. External refresh requests are latched, then synchronized to the 8202A clock.

The arbiter will allow the refresh request to start a refresh cycle only if the 8202A is not in the middle of a cycle.

Simultaneous memory request and external refresh request will result in the memory request being honored first. This 8202A characteristic can be used to "hide" refresh cycles during system operation. A circuit similar to Figure 5 can be used to decode the CPU's instruction fetch status to generate an external refresh request. The refresh request is latched while the 8202A performs the instruction fetch; the refresh cycle will start immediately after the memory cycle is completed, even if the  $\overline{\text{RD}}$  input has not gone inactive. If the CPU's instruction decode time is long enough, the 8202A can complete the refresh cycle before the next memory request is generated.

Certain system configurations require complete external refresh requests. If external refresh is requested faster than the minimum internal refresh timer ( $t_{\text{REF}}$ ), then, in effect, all refresh cycles will be caused by the external refresh request, and the internal refresh timer will never generate a refresh request.

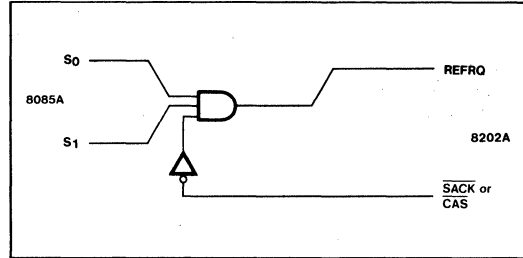


Figure 5. Hidden Refresh

**Read Cycles**

The 8202A can accept two different types of memory Read requests:

- 1) Normal Read, via the  $\overline{\text{RD}}$  input
- 2) Advanced Read, using the S1 and ALE inputs

The user can select the desired Read request configuration via the B1 / OP1 hardware strapping option on pin 25.

	Normal Read	Advanced Read
Pin 25	B1 input	+12 Volt Option
Pin 32	RD input	S1 input
Pin 34	REFRQ input	ALE input
# RAM banks	4 (RAS 0-3)	2 (RAS 2-3)
Ext. Refresh	Yes	No

Figure 6. 8202A Read Options

Normal Reads are requested by activating the  $\overline{\text{RD}}$  input, and keeping it active until the 8202A responds with an  $\overline{\text{XACK}}$  pulse. The  $\overline{\text{RD}}$  input can go inactive as soon as the command hold time ( $t_{\text{CHS}}$ ) is met.

Advanced Read cycles are requested by pulsing ALE while S1 is active; if S1 is inactive (low) ALE is ignored. Advanced Read timing is similar to Normal Read timing, except the falling edge of ALE is used as the cycle start reference.

If a Read cycle is requested while a refresh cycle is in progress, then the 8202A will set the internal delayed-SACK latch. When the Read cycle is eventually started, the 8202A will delay the active SACK transition until  $\overline{\text{XACK}}$  goes active, as shown in the AC timing diagrams. This delay was designed to compensate for the CPU's READY setup and hold times. The delayed-SACK latch is cleared after every READ cycle.

Based on system requirements, either  $\overline{\text{SACK}}$  or  $\overline{\text{XACK}}$  can be used to generate the CPU READY signal.  $\overline{\text{XACK}}$  will



normally be used; if the CPU can tolerate an advanced READY, then SACK can be used, but only if the CPU can tolerate the amount of advance provided by SACK. If SACK arrives too early to provide the appropriate number of WAIT states, then either XACK or a delayed form of SACK should be used.

**Write Cycles**

Write cycles are similar to Normal Read cycles, except for the WE output. WE is held inactive for Read cycles, but goes active for Write cycles. All 8202A Write cycles are "early-write" cycles; WE goes active before CAS goes active by an amount of time sufficient to keep the dynamic RAM output buffers turned off.

**General System Considerations**

All memory requests (Normal Reads, Advanced Reads, Writes) are qualified by the PCS input. PCS should be stable, either active or inactive, prior to the leading edge of RD, WR, or ALE. Systems which use battery backup should pullup PCS to prevent erroneous memory requests, and should also pullup WR to keep the 8202A out of its test mode.

In order to minimize propagation delay, the 8202A uses an inverting address multiplexer without latches. The system must provide adequate address setup and hold times to guarantee RAS and CAS setup and hold times for the RAM. The 8202A tAD AC parameter should be used for this system calculation.

The B0-B1 inputs are similar to the address inputs in that they are not latched. B0 and B1 should not be changed during a memory cycle, since they directly control which RAS output is activated.

The 8202A uses a two-stage synchronizer for the memory request inputs (RD, WR, ALE), and a separate two stage synchronizer for the external refresh input (REFRQ). As with any synchronizer, there is always a finite probability of metastable states inducing system errors. The 8202A synchronizer was designed to have a system error rate less than 1 memory cycle every three years based on the full operating range of the 8202A.

A microprocessor system is concerned with the time data is valid after RD goes low. See Figure 7. In order to calculate memory read access times, the dynamic RAM's A.C. specifications must be examined, especially the RAS-access time (tRAC) and the CAS-access time (tCAC). Most configurations will be CAS-access limited; i.e., the data from the RAM will be stable tCC,max (8202A) + tCAC (RAM) after a memory read cycle is started. Be sure to add any delays (due to buffers, data latches, etc.) to calculate the overall read access time.

Since the 8202A normally performs "early-write" cycles, the data must be stable at the RAM data inputs by the time CAS goes active, including the RAM's data setup time. If the system does not normally guarantee sufficient write data setup, you must either delay the WR input signal or delay the 8202A WE output.

Delaying the WR input will delay all 8202A timing, including the READY handshake signals, SACK and XACK, which may increase the number of WAIT states generated by the CPU.

If the WE output is externally delayed beyond the CAS active transition, then the RAM will use the falling edge of WE to strobe the write data into the RAM. This WE transition should not occur too late during the CAS active transition, or else the WE to CAS requirements of the RAM will not be met.

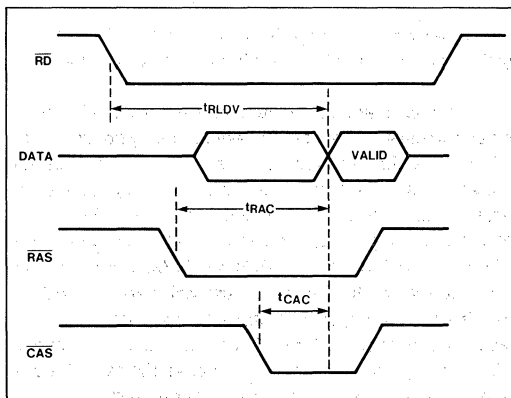


Figure 7. Read Access Time

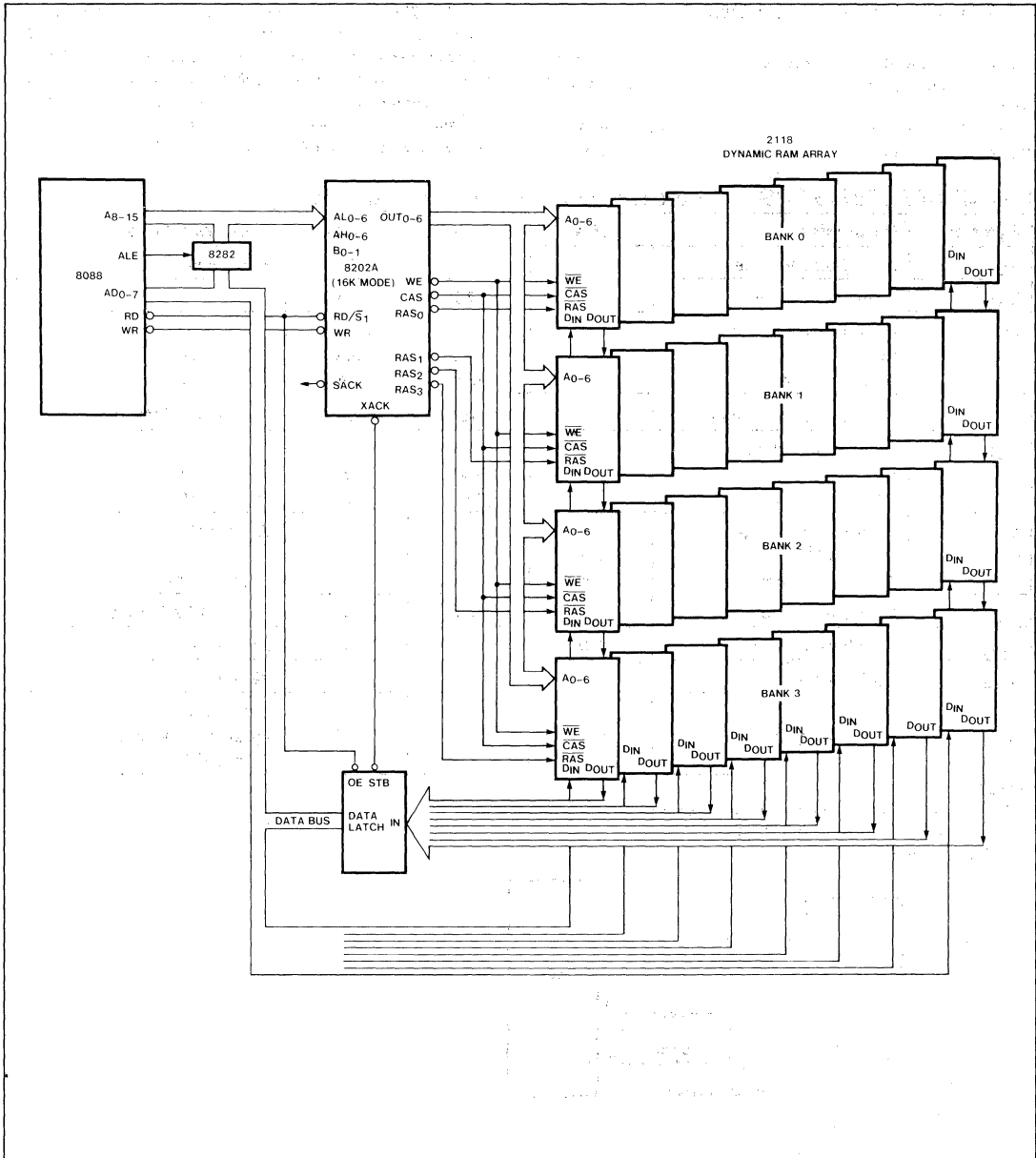


Figure 8. Typical 8088 System

**ABSOLUTE MAXIMUM RATINGS\***

Ambient Temperature Under Bias ..... 0°C to 70°C  
 Storage Temperature ..... -65°C to +150°C  
 Voltage On any Pin  
     With Respect to Ground ..... -0.5V to +7V<sup>4</sup>  
 Power Dissipation ..... 1.5 Watts

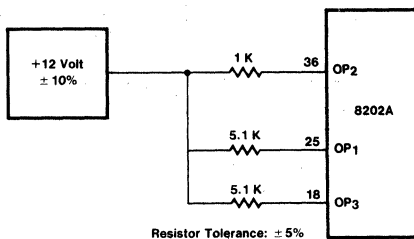
*\*NOTE: Stresses above those listed under "Absolute Maximum Ratings" may cause permanent damage to the device. This is a stress rating only and functional operation of the device at these or any other conditions above those indicated in the operational sections of this specification is not implied. Exposure to absolute maximum rating conditions for extended periods may affect device reliability.*

**D.C. CHARACTERISTICS**     $T_A = 0^\circ\text{C to } 70^\circ\text{C}$ ;  $V_{CC} = 5.0\text{V} \pm 10\%$ ;  $GND = 0\text{V}$

Symbol	Parameter	Min	Max	Units	Test Conditions
V <sub>C</sub>	Input Clamp Voltage		-1.0	V	I <sub>C</sub> = -5 mA
I <sub>CC</sub>	Power Supply Current		270	mA	
I <sub>F</sub>	Forward Input Current CLK All Other Inputs <sup>3</sup>		-2.0 -320	mA μA	V <sub>F</sub> = 0.45V V <sub>F</sub> = 0.45V
I <sub>R</sub>	Reverse Input Current <sup>3</sup>		40	μA	V <sub>R</sub> = V <sub>CC</sub> (Note 1)
V <sub>OL</sub>	Output Low Voltage SACK, XACK All Other Outputs		0.45 0.45	V V	I <sub>OL</sub> = 5 mA I <sub>OL</sub> = 3 mA
V <sub>OH</sub>	Output High Voltage SACK, XACK All Other Outputs	2.4 2.6		V V	V <sub>IL</sub> = 0.65V I <sub>OH</sub> = -1 mA I <sub>OH</sub> = -1 mA
V <sub>IL</sub>	Input Low Voltage		0.8	V	V <sub>CC</sub> = 5.0V (Note 2)
V <sub>IH1</sub>	Input High Voltage	2.0		V	V <sub>CC</sub> = 5.0V
V <sub>IH2</sub>	Option Voltage			V	(Note 4)
C <sub>IN</sub>	Input Capacitance		30	pF	F = 1 MHz V <sub>BIAS</sub> = 2.5V, V <sub>CC</sub> = 5V T <sub>A</sub> = 25°C

**NOTES:**

1. I<sub>R</sub> = 200 mA for pin 37 (CLK) for external clock mode.
2. For test mode  $\overline{RD}$  &  $\overline{WR}$  must be held at GND.
3. Except for pin 36.
- 4.



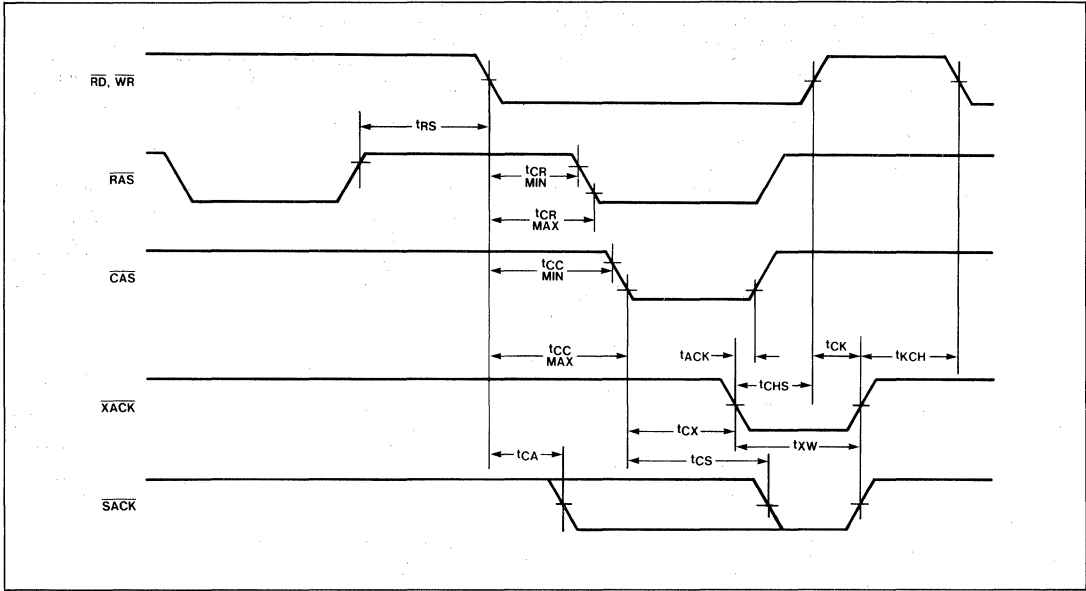
**A.C. CHARACTERISTICS**
 $T_A = 0^\circ\text{C to } 70^\circ\text{C}, V_{CC} = 5\text{V} \pm 10\%$ 

 Measurements made with respect to  $\overline{\text{RAS}}_0\text{--}\overline{\text{RAS}}_3, \overline{\text{CAS}}, \overline{\text{WE}}, \overline{\text{OUT}}_0\text{--}\overline{\text{OUT}}_6$  are at 2.4V and 0.8V. All other pins are measured at 1.5V. All times are in nsec.

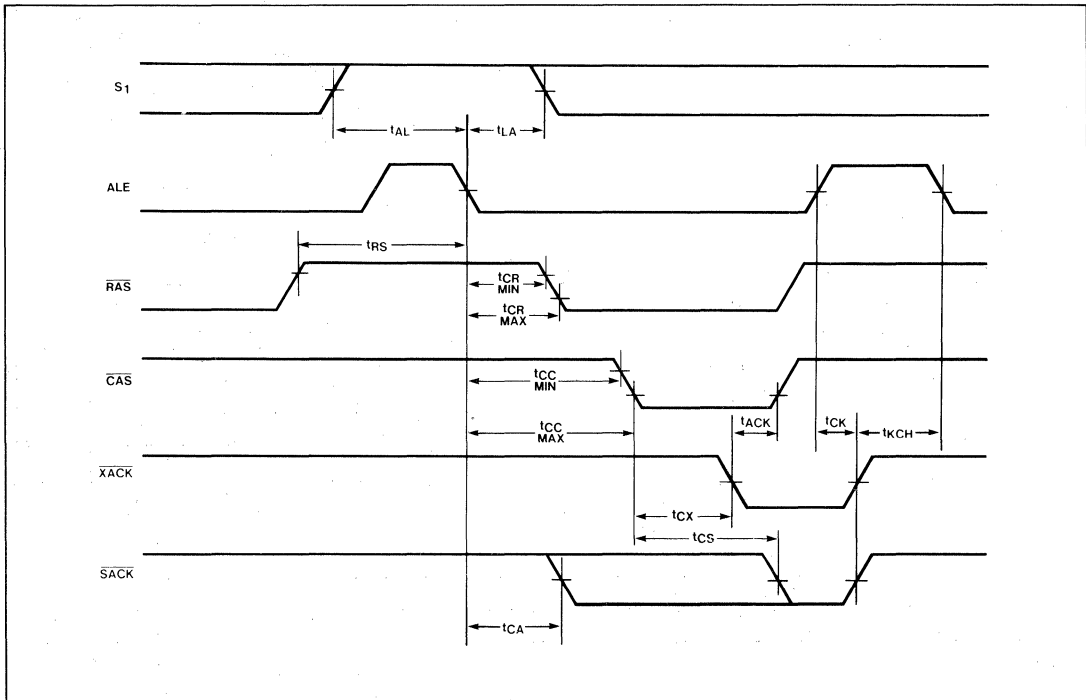
Symbol	Parameter	Min	Max	Notes
$t_p$	Clock Period	40	54	
$t_{PH}$	External Clock High Time	20		
$t_{PL}$	External Clock Low Time—above ( $>$ ) 20 MHz	17		
$t_{PL}$	External Clock Low Time—below ( $<$ ) 20 MHz	20		
$t_{RC}$	Memory Cycle Time	$10t_p - 30$	12 $t_p$	4, 5
$t_{REF}$	Refresh Time (64 cycles—4K mode)	548 $t_p$	576 $t_p$	
$t_{REF}$	Refresh Time (128 cycles—16K mode)	264 $t_p$	288 $t_p$	
$t_{RP}$	$\overline{\text{RAS}}$ Precharge Time	$4t_p - 30$		
$t_{RSH}$	$\overline{\text{RAS}}$ Hold After $\overline{\text{CAS}}$	$5t_p - 30$		3
$t_{ASR}$	Address Setup to $\overline{\text{RAS}}$	$t_p - 30$		3
$t_{RAH}$	Address Hold From $\overline{\text{RAS}}$	$t_p - 10$		3
$t_{ASC}$	Address Setup to $\overline{\text{CAS}}$	$t_p - 30$		3
$t_{CAH}$	Address Hold from $\overline{\text{CAS}}$	$5t_p - 20$		3
$t_{CAS}$	$\overline{\text{CAS}}$ Pulse Width	$5t_p - 10$		
$t_{WCS}$	$\overline{\text{WE}}$ Setup to $\overline{\text{CAS}}$	$t_p - 40$		
$t_{WCH}$	$\overline{\text{WE}}$ Hold After $\overline{\text{CAS}}$	$5t_p - 35$		8
$t_{RS}$	$\overline{\text{RD}}, \overline{\text{WR}}, \text{ALE}, \text{REFRQ}$ delay from $\overline{\text{RAS}}$	5 $t_p$		
$t_{MRP}$	$\overline{\text{RD}}, \overline{\text{WR}}$ setup to $\overline{\text{RAS}}$	0		5
$t_{RMS}$	REFRQ setup to $\overline{\text{RD}}, \overline{\text{WR}}$	2 $t_p$		
$t_{RMP}$	REFRQ setup to $\overline{\text{RAS}}$	2 $t_p$		5
$t_{PCS}$	$\overline{\text{PCS}}$ Setup to $\overline{\text{RD}}, \overline{\text{WR}}, \text{ALE}$	20		
$t_{AL}$	S1 Setup to ALE	15		
$t_{LA}$	S1 Hold from ALE	30		
$t_{CR}$	$\overline{\text{RD}}, \overline{\text{WR}}, \text{ALE}$ to $\overline{\text{RAS}}$ Delay	$t_p + 30$	$2t_p + 70$	2
$t_{CC}$	$\overline{\text{RD}}, \overline{\text{WR}}, \text{ALE}$ to $\overline{\text{CAS}}$ Delay	$3t_p + 25$	$4t_p + 85$	2
$t_{SC}$	CMD Setup to Clock	15		1
$t_{MRS}$	$\overline{\text{RD}}, \overline{\text{WR}}$ setup to REFRQ	5		
$t_{CA}$	$\overline{\text{RD}}, \overline{\text{WR}}, \text{ALE}$ to $\overline{\text{SACK}}$ Delay		$2t_p + 47$	2
$t_{CX}$	$\overline{\text{CAS}}$ to $\overline{\text{XACK}}$ Delay	$5t_p - 25$	$5t_p + 20$	
$t_{CS}$	$\overline{\text{CAS}}$ to $\overline{\text{SACK}}$ Delay	$5t_p - 25$	$5t_p + 40$	2
$t_{ACK}$	$\overline{\text{XACK}}$ to $\overline{\text{CAS}}$ Setup	10		
$t_{XW}$	$\overline{\text{XACK}}$ Pulse Width	$t_p - 25$		7
$t_{CK}$	$\overline{\text{SACK}}, \overline{\text{XACK}}$ turn-off Delay		35	
$t_{KCH}$	CMD Inactive Hold after $\overline{\text{SACK}}, \overline{\text{XACK}}$	10		
$t_{LL}$	REFRQ Pulse Width	20		
$t_{CHS}$	CMD Hold Time	30		
$t_{RFR}$	REFRQ to $\overline{\text{RAS}}$ Delay		$4t_p + 100$	6
$t_{WW}$	$\overline{\text{WR}}$ to $\overline{\text{WE}}$ Delay	0	50	8
$t_{AD}$	CPU Address Delay	0	40	3

WAVEFORMS

Normal Read or Write Cycle

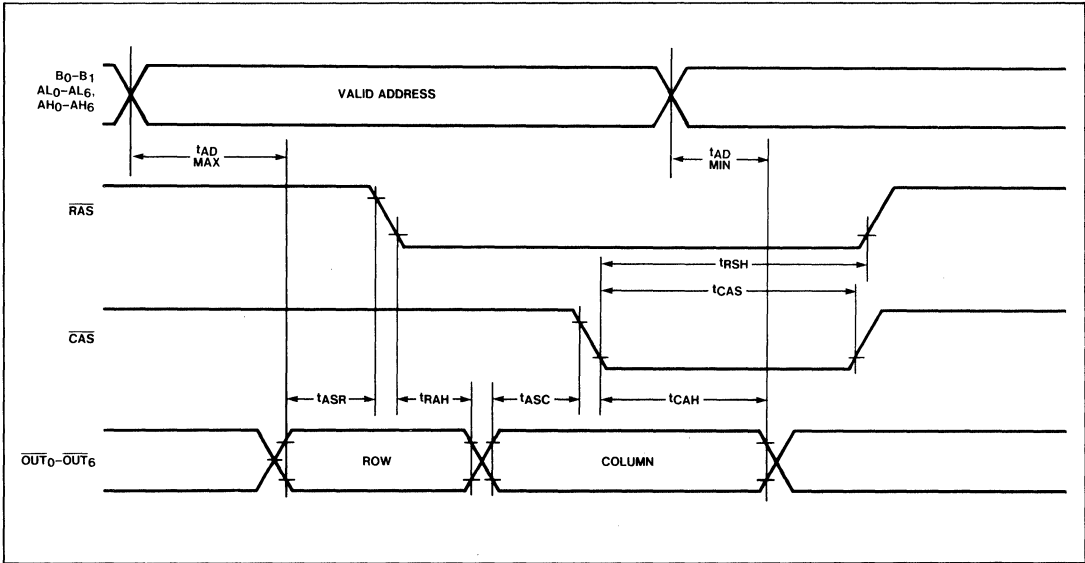


Advanced Read Mode

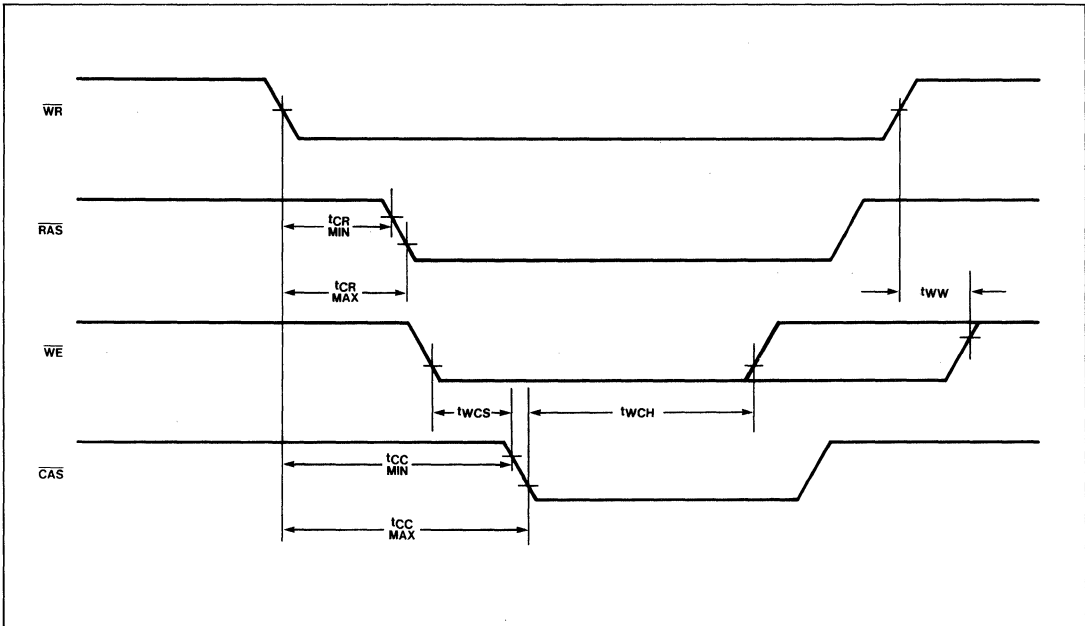


WAVEFORMS (cont'd)

Memory Compatibility Timing

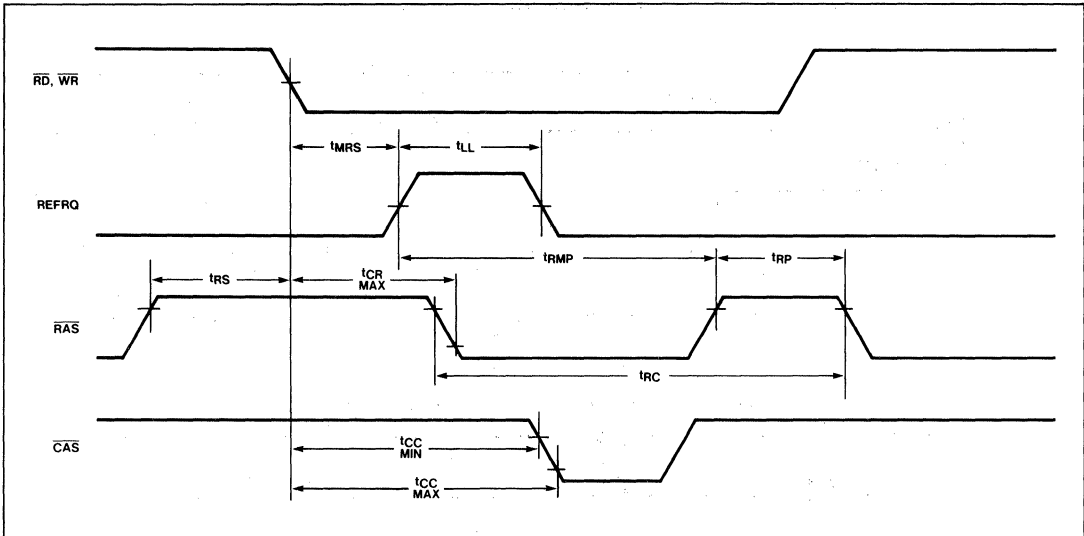


Write Cycle Timing

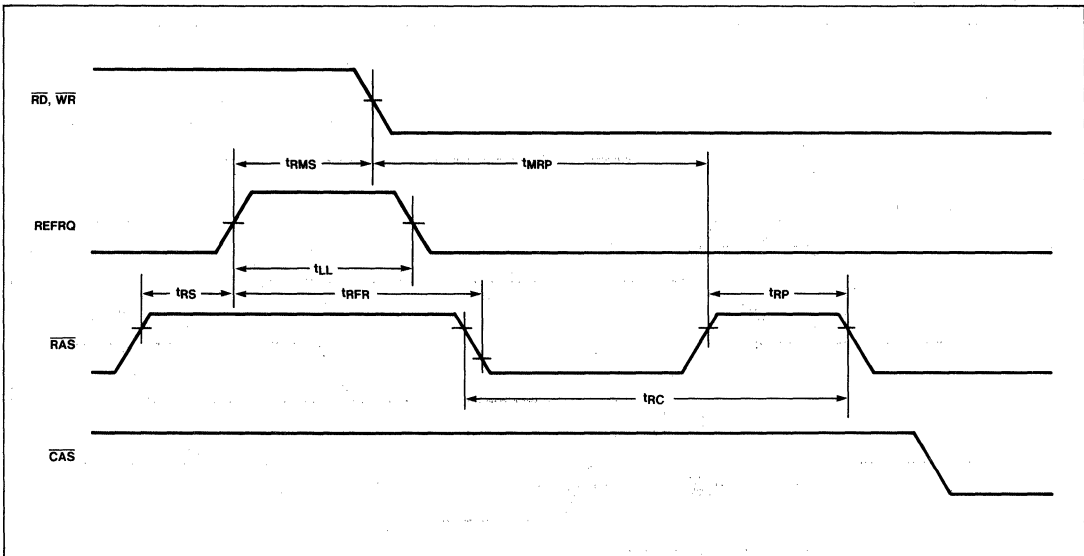


WAVEFORMS (cont'd)

Read or Write Followed By External Refresh



External Refresh Followed By Read or Write



WAVEFORMS (cont'd)

Clock And System Timing

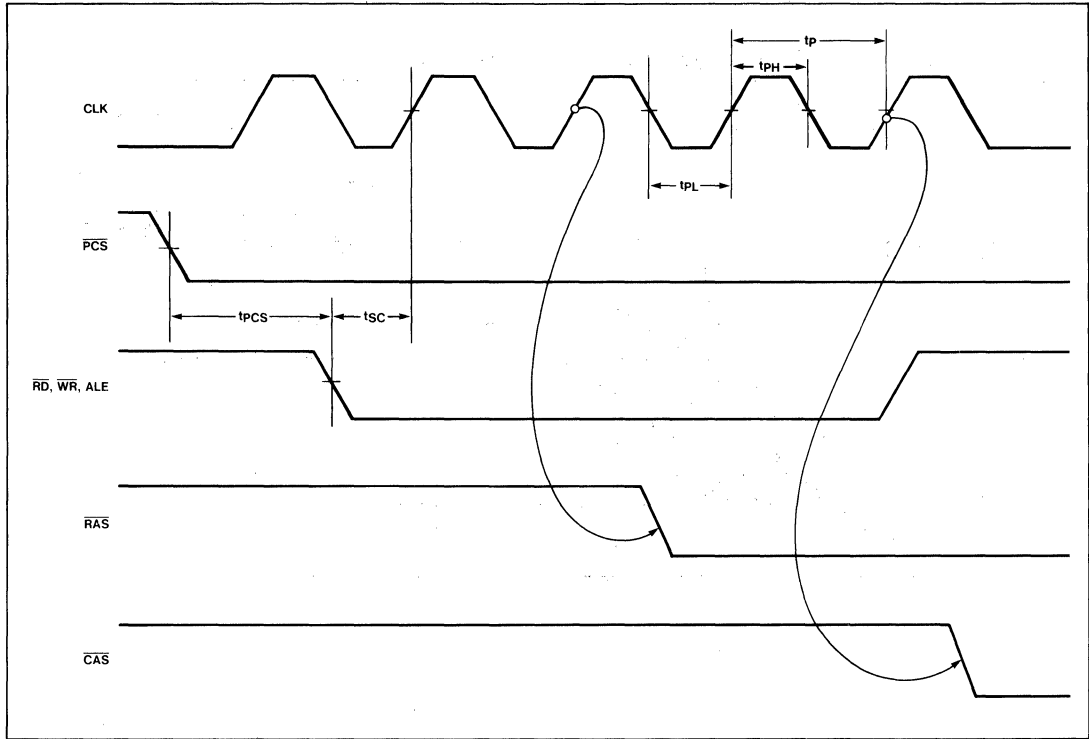


Table 2 8202A Output Test Loading.

Pin	Test Load
SACK, XACK	C <sub>L</sub> = 30 pF
OUT <sub>0</sub> -OUT <sub>6</sub>	C <sub>L</sub> = 160 pF
RAS <sub>0</sub> -RAS <sub>3</sub>	C <sub>L</sub> = 60 pF
WE	C <sub>L</sub> = 224 pF
CAS	C <sub>L</sub> = 320 pF

NOTES:

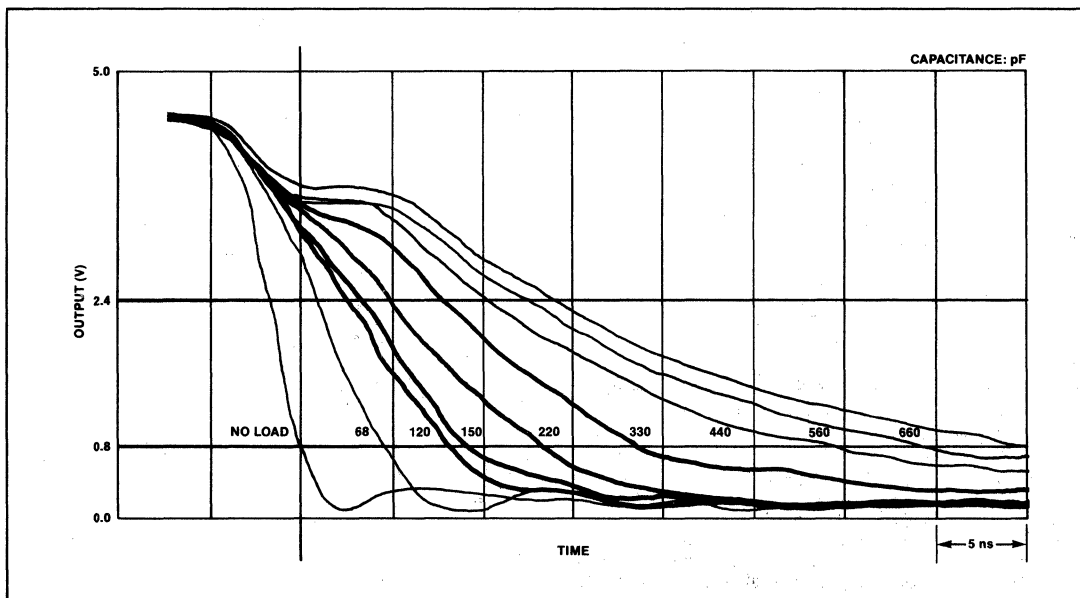
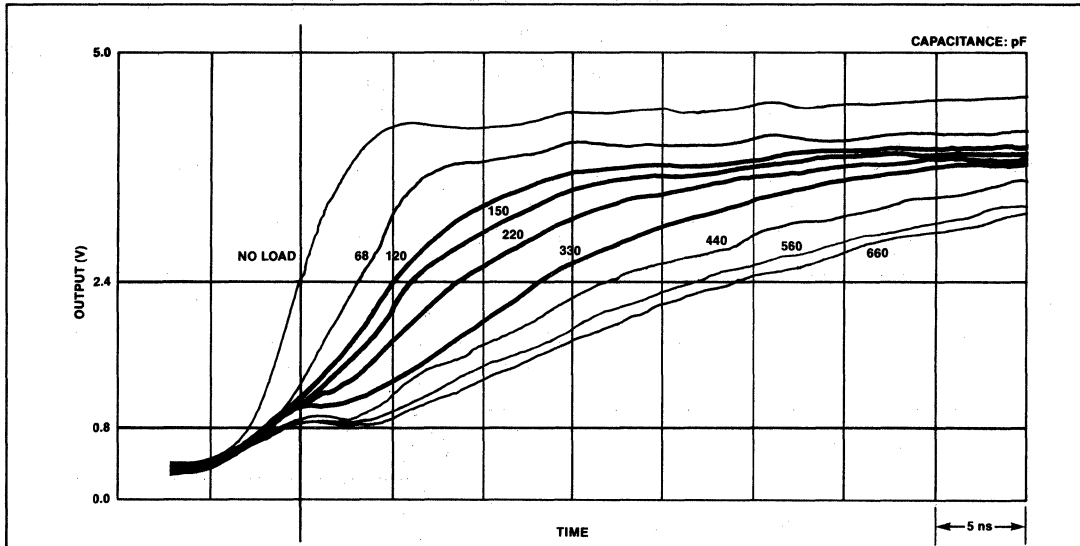
1. t<sub>SC</sub> is a reference point only. ALE, RD, WR, and REFRQ inputs do not have to be externally synchronized to 8202A clock.
2. If t<sub>RS</sub> min and t<sub>MRS</sub> min are met then, t<sub>CA</sub>, t<sub>CR</sub>, and t<sub>CC</sub> are valid, otherwise t<sub>CS</sub> is valid.
3. t<sub>ASR</sub>, t<sub>RAH</sub>, t<sub>ASC</sub>, t<sub>CAH</sub>, and t<sub>RS</sub> depend upon B0-B1 and CPU address remaining stable throughout the memory cycle. The address inputs are not latched by the 8202A.
4. For back-to-back refresh cycles, t<sub>RC</sub> max = 13tp
5. t<sub>RC</sub> max is valid only if t<sub>RMP</sub> min is met (READ, WRITE followed by REFRESH) or t<sub>MRP</sub> min is met (REFRESH followed by READ, WRITE).
6. t<sub>RFR</sub> is valid only if t<sub>RS</sub> min and t<sub>RMS</sub> min are met.
7. t<sub>XW</sub> min applies when RD, WR has already gone high. Otherwise XACK follows RD, WR.
8. WE goes high according to t<sub>WCH</sub> or t<sub>WW</sub>, whichever occurs first.



The typical rising and falling characteristic curves can be used to determine the effects of capacitive loading on the A.C. Timing Parameters. Using this

design tool in conjunction with the timing waveforms, the designer can determine typical timing shifts based on system capacitive load.

**A.C. CHARACTERISTICS FOR DIFFERENT CAPACITIVE LOADS**



**NOTE:**  
Use the Test Load as the base capacitance for estimating timing shifts for system critical timing parameters.

**TYPICAL CONDITIONS:**  
 $T_A = 25^\circ\text{C}$  Pins not under test are loaded with Test Load capacitance.  
 $V_{CC} = +5\text{V}$   
 $t_p = 50 \text{ ns}$

Example: Find the effect on  $t_{CR}$  and  $t_{CC}$  using 64 2118 Dynamic RAMs configured in 4 banks.

1. Determine the typical RAS and CAS capacitance:  
From the data sheet RAS = 4 pF and CAS = 4 pF.

- ∴ RAS load = 64 pF + board capacitance.
- CAS load = 256 pF + board capacitance.
- Assume 2 pF/in (trace length) for board capacitance.

2. From the waveform diagrams, we determine that the falling edge timing is needed for  $t_{CR}$  and  $t_{CC}$ . Next find the curve that *best* approximates the test load; i.e., 68 pF for RAS and 330 pF for CAS.

3. If we use 72 pF for RAS loading, then the  $t_{CR}$  (max.) spec should be increased by *about* 1 ns. Similarly if we use 288 pF for CAS, then  $t_{CC}$  (min.) and (max.) should decrease about 1 ns.

## 8203 64K DYNAMIC RAM CONTROLLER

- Provides All Signals Necessary to Control 64K (2164), 16K (2117, 2118) and 4K (2104A) Dynamic Memories
- Fully Compatible with Intel® 8080A, 8085A, iAPX 88, and iAPX 86 Family Microprocessors
- Directly Addresses and Drives Up to 64 Devices Without External Drivers
- Decodes CPU Status for Advanced Read Capability in 16K mode
- Provides Address Multiplexing and Strobes
- Provides System Acknowledge and Transfer Acknowledge Signals
- Provides a Refresh Timer and a Refresh Counter
- Refresh Cycles May be Internally or Externally Requested (For Transparent Refresh)
- Provides Refresh/Access Arbitration
- Internal Series Damping Resistors on All Outputs
- External or Internal Clock Capability

The Intel® 8203 is a Dynamic Ram System Controller designed to provide all signals necessary to use 2164, 2118, 2117, or 2104A Dynamic RAMs in microcomputer systems. The 8203 provides multiplexed addresses and address strobes, refresh logic, refresh/access arbitration. Refresh cycles can be started internally or externally.

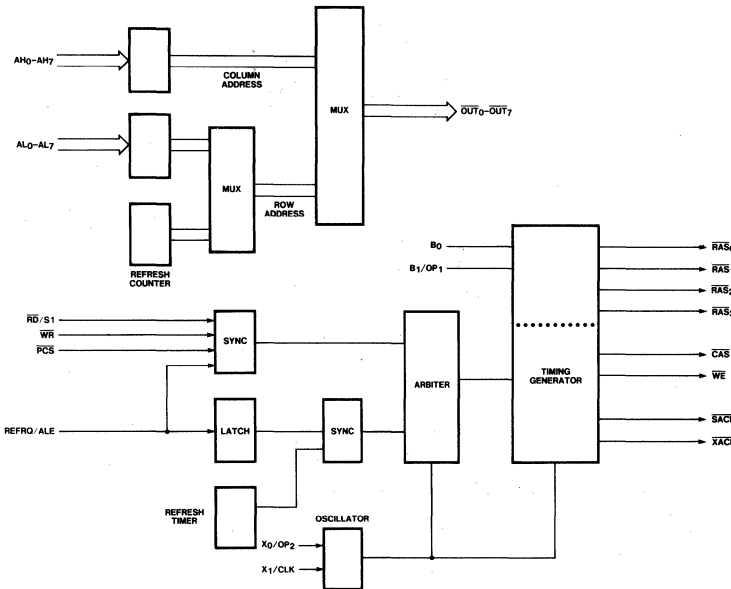


Figure 1. 8203 Block Diagram

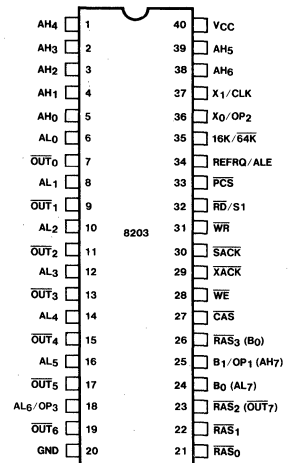


Figure 2. Pin Configuration

Table 1. Pin Descriptions

Symbol	Pin No.	Type	Name and Function
AL <sub>0</sub>	6	I	<b>Address Low:</b> CPU address inputs used to generate memory row address. AL <sub>6</sub> /OP <sub>3</sub> used to select 4K RAM mode.
AL <sub>1</sub>	8	I	
AL <sub>2</sub>	10	I	
AL <sub>3</sub>	12	I	
AL <sub>4</sub>	14	I	
AL <sub>5</sub>	16	I	
AL <sub>6</sub> /OP <sub>3</sub>	18	I	
AH <sub>0</sub>	5	I	<b>Address High:</b> CPU address inputs used to generate memory column address.
AH <sub>1</sub>	4	I	
AH <sub>2</sub>	3	I	
AH <sub>3</sub>	2	I	
AH <sub>4</sub>	1	I	
AH <sub>5</sub>	39	I	
AH <sub>6</sub>	38	I	
B <sub>0</sub> /AL <sub>7</sub>	24	I	<b>Bank Select Inputs:</b> Used to gate the appropriate RAS output for a memory cycle. B <sub>1</sub> /OP <sub>1</sub> option used to select the Advanced Read Mode. (Not available in 64K mode.) See Figure 5. When in 64K RAM Mode, pins 24 and 25 operate as the AL <sub>7</sub> and AH <sub>7</sub> address inputs.
B <sub>1</sub> /OP <sub>1</sub> /AH <sub>7</sub>	25	I	
PCS	33	I	<b>Protected Chip Select:</b> Used to enable the memory read and write inputs. Once a cycle is started, it will not abort even if PCS goes inactive before cycle completion.
WR	31	I	<b>Memory Write Request.</b>
RD/S1	32	I	<b>Memory Read Request:</b> S1 function used in Advanced Read mode selected by OP <sub>1</sub> (pin 25).
REFRQ/ALE	34	I	<b>External Refresh Request:</b> ALE function used in Advanced Read mode, selected by OP <sub>1</sub> (pin 25).
OUT <sub>0</sub>	7	O	<b>Output of the Multiplexer:</b> These outputs are designed to drive the addresses of the Dynamic RAM array. (Note that the OUT <sub>0-7</sub> pins do not require inverters or drivers for proper operation.
OUT <sub>1</sub>	9	O	
OUT <sub>2</sub>	11	O	
OUT <sub>3</sub>	13	O	
OUT <sub>4</sub>	15	O	
OUT <sub>5</sub>	17	O	
OUT <sub>6</sub>	19	O	
WE	28	O	<b>Write Enable:</b> Drives the Write Enable inputs of the Dynamic RAM array.
CAS	27	O	<b>Column Address Strobe:</b> This output is used to latch the Column Address into the Dynamic RAM array.

Symbol	Pin No.	Type	Name and Function
RAS <sub>0</sub>	21	O	<b>Row Address Strobe:</b> Used to latch the Row Address into the bank of dynamic RAMs, selected by the 8203 Bank Select pins (B <sub>0</sub> , B <sub>1</sub> /OP <sub>1</sub> ). In 64K mode, only RAS <sub>0</sub> and RAS <sub>1</sub> are available; pin 23 operates as $\overline{\text{OUT}}_7$ and pin 26 operates as the B <sub>0</sub> bank select input.
RAS <sub>1</sub>	22	O	
RAS <sub>2</sub> / $\overline{\text{OUT}}_7$	23	O	
RAS <sub>3</sub> /B <sub>0</sub>	26	I/O	
XACK	29	O	<b>Transfer Acknowledge:</b> This output is a strobe indicating valid data during a read cycle or data written during a write cycle. XACK can be used to latch valid data from the RAM array.
SACK	30	O	<b>System Acknowledge:</b> This output indicates the beginning of a memory access cycle. It can be used as an advanced transfer acknowledge to eliminate wait states. (Note: If a memory access request is made during a refresh cycle, SACK is delayed until XACK in the memory access cycle).
X <sub>0</sub> /OP <sub>2</sub>	36	I/O	<b>Oscillator Inputs:</b> These inputs are designed for a quartz crystal to control the frequency of the oscillator. If X <sub>0</sub> /OP <sub>2</sub> is directly pulled up to V <sub>CC</sub> or if X <sub>0</sub> /OP <sub>2</sub> is connected to a 1K $\Omega$ resistor pulled to +12V then X <sub>1</sub> /CLK becomes a TTL input for an external clock.
X <sub>1</sub> /CLK	37	I/O	
16K/64K	35	I	<b>Mode Select:</b> This input selects 16K mode (2117, 2118) or 64K mode (2164). Pins 23-26 change function based on the mode of operation.
V <sub>CC</sub>	40		<b>Power Supply:</b> +5V.
GND	20		<b>Ground.</b>

### Functional Description

The 8203 provides a complete dynamic RAM controller for microprocessor systems as well as expansion memory boards. All of the necessary control signals are provided for 2164, 2118, 2117, and 2104A dynamic RAM's.

The 8203 has three modes, one for 4K dynamic RAM's, one for 16K's and one for 64K's, controlled by pin 35 and pin 18.

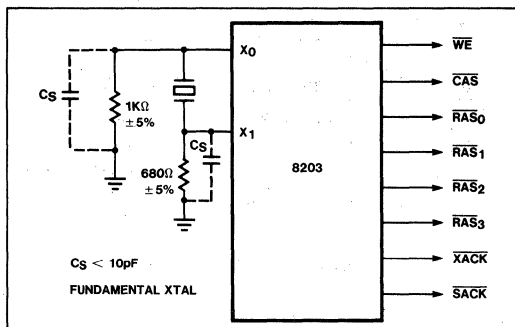


Figure 3. Crystal Operation

All 8203 timing is generated from a single reference clock. This clock is provided via an external oscillator or an on-chip crystal oscillator. All output signal transitions are synchronous with respect to this clock reference, except for the CPU handshake signals  $\overline{SACK}$  and  $\overline{XACK}$ .

CPU memory requests normally use the  $\overline{RD}$  and  $\overline{WR}$  inputs. The Advanced-Read mode allows ALE and S1 to be used in place of the  $\overline{RD}$  input.

Failsafe refresh is provided via an internal timer which generates refresh requests. Refresh requests can also be generated via the REFREQ input.

An on-chip synchronizer / arbiter prevents memory and refresh requests from affecting a cycle in progress. The READ, WRITE, and external REFRESH requests may be asynchronous to the 8203 clock; on-chip logic will synchronize the requests, and the arbiter will decide if the requests should be delayed, pending completion of a cycle in progress.

### 16K/64K Option Selection

Pin 35 is a strap input that controls the two 8203 modes. Figure 4 shows the four pins that are multiplexed. In 16K mode (pin 35 tied to  $V_{CC}$  or left open), the 8203 has two Bank Select inputs to select one of four  $\overline{RAS}$  outputs. In this mode, the 8203 is exactly compatible with the Intel 8202A Dynamic RAM Controller. In 64K mode (pin 35 tied to GND), there is only one Bank Select input (pin 26) to select the two  $\overline{RAS}$  outputs. More than two banks of 64K dynamic RAM's can be used with external logic.

### Other Option Selections

The 8203 has three strapping options. When OP<sub>1</sub> is selected (16K mode only), pin 32 changes from a  $\overline{RD}$  input to an S1 input, and pin 34 changes from a REFREQ input to an ALE input. See "Refresh Cycles" and "Read Cycles" for more detail. OP<sub>1</sub> is selected by tying pin 25 to +12V through a 5.1K ohm resistor.

When OP<sub>2</sub> is selected, by connecting pin 36 to  $V_{CC}$ , pin 37 changes from a crystal input (X<sub>1</sub>) to the CLK input for an external TTL clock.

OP<sub>3</sub> is selected by connecting pin 18 to +12V through a 5.1K ohm resistor, the 8203 will change its internal refresh timer from 128-row refresh (2164, 2118, 2117) to 64-row refresh (2104A).

### Refresh Timer

The refresh timer is used to monitor the time since the last refresh cycle occurred. When the appropriate amount of time has elapsed, the refresh timer will request a refresh cycle. External refresh requests will reset the refresh timer.

### Refresh Counter

The refresh counter is used to sequentially refresh all of the memory's rows. The 8-bit counter is incremented after every refresh cycle.

Pin #	16K Function	64K Function
23	$\overline{RAS}_2$	Address Output ( $\overline{OUT}_7$ )
24	Bank Select (B <sub>0</sub> )	Address Input (AL <sub>7</sub> )
25	Bank Select (B <sub>1</sub> )	Address Input (AH <sub>7</sub> )
26	$\overline{RAS}_3$	Bank Select (B <sub>0</sub> )

Figure 4. 16K/64K Mode Selection

	Inputs		Outputs			
	B <sub>0</sub>	B <sub>1</sub>	$\overline{RAS}_0$	$\overline{RAS}_1$	$\overline{RAS}_2$	$\overline{RAS}_3$
16K Mode	0	0	0	1	1	1
	0	1	1	0	1	1
	1	0	1	1	0	1
	1	1	1	1	1	0
64K Mode	0	—	0	1	—	—
	1	—	1	0	—	—

Figure 5. Bank Selection

Description	Pin #	Normal Function	Option Function
B1/OP <sub>1</sub> (16K only)/AH <sub>7</sub>	25	Bank (RAS) Select	Advanced-Read Mode
X <sub>0</sub> /OP <sub>2</sub>	36	Internal (Crystal) Oscillator	External Oscillator
AL <sub>6</sub> /OP <sub>3</sub>	18	Address Input	64-Row Refresh

Figure 6. 8203 Option Selection

## Address Multiplexer

The address multiplexer takes the address inputs and the refresh counter outputs, and gates them onto the address outputs at the appropriate time. The address outputs, in conjunction with the  $\overline{\text{RAS}}$  and  $\overline{\text{CAS}}$  outputs, determine the address used by the dynamic RAMs for read, write, and refresh cycles. During the first part of a read or write cycle,  $\text{AL}_0\text{--}\text{AL}_7$  are gated to  $\overline{\text{OUT}}_0\text{--}\overline{\text{OUT}}_7$ , then  $\text{AH}_0\text{--}\text{AH}_7$  are gated to the address outputs.

During a refresh cycle, the refresh counter is gated onto the address outputs. All refresh cycles are RAS-only refresh ( $\overline{\text{CAS}}$  inactive,  $\overline{\text{RAS}}$  active).

To minimize buffer delay, the information on the address outputs is inverted from that on the address inputs.

$\overline{\text{OUT}}_0\text{--}\overline{\text{OUT}}_7$  do not need inverters or buffers unless additional drive is required.

## Synchronizer / Arbiter

The 8203 has three inputs,  $\text{REFRQ}/\text{ALE}$  (pin 34),  $\overline{\text{RD}}$  (pin 32) and  $\overline{\text{WR}}$  (pin 31). The  $\overline{\text{RD}}$  and  $\overline{\text{WR}}$  inputs allow an external CPU to request a memory read or write cycle, respectively. The  $\text{REFRQ}/\text{ALE}$  allows refresh requests to be requested external to the 8203.

All three of these inputs may be asynchronous with respect to the 8203's clock. The arbiter will resolve conflicts between refresh and memory requests, for both pending cycles and cycles in progress. Read and write requests will be given priority over refresh requests.

## System Operation

The 8203 is always in one of the following states:

- a) IDLE
- b) TEST Cycle
- c) REFRESH Cycle
- d) READ Cycle
- e) WRITE Cycle

The 8203 is normally in the IDLE state. Whenever one of the other cycles is requested, the 8203 will leave the IDLE state to perform the desired cycle. If no other cycles are pending, the 8203 will return to the IDLE state.

## Test Cycle

The TEST Cycle is used to check operation of several 8203 internal functions. TEST cycles are requested by activating the PCS,  $\overline{\text{RD}}$  and  $\overline{\text{WR}}$  inputs. The TEST Cycle will reset the refresh address counter and perform a WRITE Cycle. The TEST Cycle should not be used in normal system operation, since it would affect the dynamic RAM refresh.

## Refresh Cycles

The 8203 has two ways of providing dynamic RAM refresh:

- 1) Internal (failsafe) refresh
- 2) External (hidden) refresh

Both types of 8203 refresh cycles activate all of the  $\overline{\text{RAS}}$  outputs, while  $\overline{\text{CAS}}$ ,  $\overline{\text{WE}}$ ,  $\overline{\text{SACK}}$ , and  $\overline{\text{XACK}}$  remain inactive.

Internal refresh is generated by the on-chip refresh timer. The timer uses the 8203 clock to ensure that refresh of all rows of the dynamic RAM occurs every 2 milliseconds (128 cycles) or every 4 milliseconds (256 cycles). If  $\text{REFRQ}$  is inactive, the refresh timer will request a refresh cycle every 10-16 microseconds.

External refresh is requested via the  $\text{REFRQ}$  input (pin 34). External refresh control is not available when the Advanced-Read mode is selected. External refresh requests are latched, then synchronized to the 8203 clock.

The arbiter will allow the refresh request to start a refresh cycle only if the 8203 is not in the middle of a cycle.

Simultaneous memory request and external refresh request will result in the memory request being honored first. This 8203 characteristic can be used to "hide" refresh cycles during system operation. A circuit similar to Figure 7 can be used to decode the CPU's instruction fetch status to generate an external refresh request. The refresh request is latched while the 8203 performs the instruction fetch; the refresh cycle will start immediately after the memory cycle is completed, even if the  $\overline{\text{RD}}$  input has not gone inactive. If the CPU's instruction decode time is long enough, the 8203 can complete the refresh cycle before the next memory request is generated.

Certain system configurations require complete external refresh requests. If external refresh is requested faster than the minimum internal refresh timer ( $t_{\text{REF}}$ ), then, in effect, all refresh cycles will be caused by the external refresh request, and the internal refresh timer will never generate a refresh request.

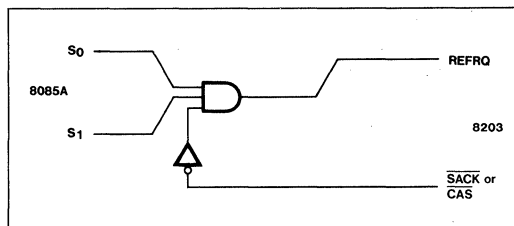


Figure 7. Hidden Refresh

**Read Cycles**

The 8203 can accept two different types of memory Read requests:

- 1) Normal Read, via the  $\overline{RD}$  input
- 2) Advanced Read, using the S1 and ALE inputs (16K mode only)

The user can select the desired Read request configuration via the B1/OP1 hardware strapping option on pin 25.

	Normal Read	Advanced Read
Pin 25	B1 input	+12 Volt Option
Pin 32	RD input	S1 input
Pin 34	REFRQ input	ALE input
# RAM banks	4 ( $\overline{RAS}$ 0-3)	2 ( $\overline{RAS}$ 2-3)
Ext. Refresh	Yes	No

**Figure 8. 8203 Read Options**

Normal Reads are requested by activating the  $\overline{RD}$  input, and keeping it active until the 8203 responds with an  $\overline{XACK}$  pulse. The  $\overline{RD}$  input can go inactive as soon as the command hold time ( $t_{CHS}$ ) is met.

Advanced Read cycles are requested by pulsing ALE while S1 is active; if S1 is inactive (low) ALE is ignored. Advanced Read timing is similar to Normal Read timing, except the falling edge of ALE is used as the cycle start reference.

If a Read cycle is requested while a refresh cycle is in progress, then the 8203 will set the internal delayed-SACK latch. When the Read cycle is eventually started, the 8203 will delay the active SACK transition until  $\overline{XACK}$  goes active, as shown in the AC timing diagrams. This delay was designed to compensate for the CPU's READY setup and hold times. The delayed-SACK latch is cleared after every READ cycle.

Based on system requirements, either  $\overline{SACK}$  or  $\overline{XACK}$  can be used to generate the CPU READY signal.  $\overline{XACK}$  will normally be used; if the CPU can tolerate an advanced READY, then  $\overline{SACK}$  can be used, but only if the CPU can tolerate the amount of advance provided by  $\overline{SACK}$ . If  $\overline{SACK}$  arrives too early to provide the appropriate number of WAIT states, then either  $\overline{XACK}$  or a delayed form of SACK should be used.

**Write Cycles**

Write cycles are similar to Normal Read cycles, except for the  $\overline{WE}$  output.  $\overline{WE}$  is held inactive for Read cycles, but goes active for Write cycles. All 8203 Write cycles are "early-write" cycles;  $\overline{WE}$  goes active before  $\overline{CAS}$  goes active by an amount of time sufficient to keep the dynamic RAM output buffers turned off.

**General System Considerations**

All memory requests (Normal Reads, Advanced Reads, Writes) are qualified by the PCS input. PCS should be stable, either active or inactive, prior to the leading edge of  $\overline{RD}$ ,  $\overline{WR}$ , or ALE. Systems which use battery backup should pullup  $\overline{PCS}$  to prevent erroneous memory requests.

In order to minimize propagation delay, the 8203 uses an inverting address multiplexer without latches. The system must provide adequate address setup and hold times to guarantee  $\overline{RAS}$  and  $\overline{CAS}$  setup and hold times for the RAM. The  $t_{AD}$  AC parameter should be used for this system calculation.

The  $B_0$ - $B_1$  inputs are similar to the address inputs in that they are not latched.  $B_0$  and  $B_1$  should not be changed during a memory cycle, since they directly control which  $\overline{RAS}$  output is activated.

The 8203 uses a two-stage synchronizer for the memory request inputs ( $\overline{RD}$ ,  $\overline{WR}$ , ALE), and a separate two stage synchronizer for the external refresh input (REFRQ). As with any synchronizer, there is always a finite probability of metastable states inducing system errors. The 8203 synchronizer was designed to have a system error rate less than 1 memory cycle every three years based on the full operating range of the 8203.

A microprocessor system is concerned when the data is valid after  $\overline{RD}$  goes low. See Figure 9. In order to calculate memory read access times, the dynamic RAM's A.C. specifications must be examined, especially the RAS-access time ( $t_{RAC}$ ) and the CAS-access time ( $t_{CAC}$ ). Most configurations will be CAS-access limited; i.e., the data from the RAM will be stable  $t_{CC,max}(8203) + t_{CAC}(RAM)$  after a memory read cycle is started. Be sure to add any delays (due to buffers, data latches, etc.) to calculate the overall read access time.

Since the 8203 normally performs "early-write" cycles, the data must be stable at the RAM data inputs by the time  $\overline{CAS}$  goes active, including the RAM's data setup time. If the system does not normally guarantee sufficient write data setup, you must either delay the  $\overline{WR}$  input signal or delay the 8203  $\overline{WE}$  output.

Delaying the  $\overline{WR}$  input will delay all 8203 timing, including the READY handshake signals, SACK and  $\overline{XACK}$ , which may increase the number of WAIT states generated by the CPU.

If the  $\overline{WE}$  output is externally delayed beyond the  $\overline{CAS}$  active transition, then the RAM will use the falling edge of  $\overline{WE}$  to strobe the write data into the RAM. This  $\overline{WE}$  transition should not occur too late during the CAS active transition, or else the  $\overline{WE}$  to  $\overline{CAS}$  requirements of the RAM will not be met.

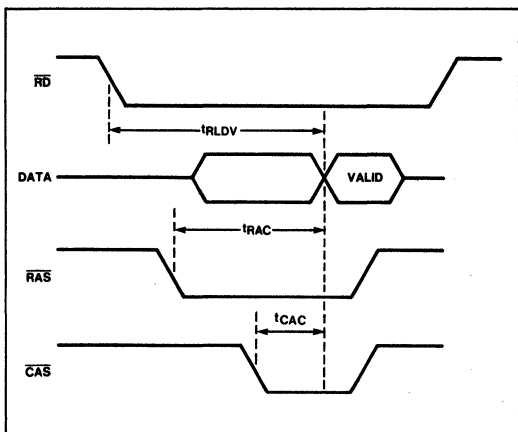


Figure 9. Read Access Time

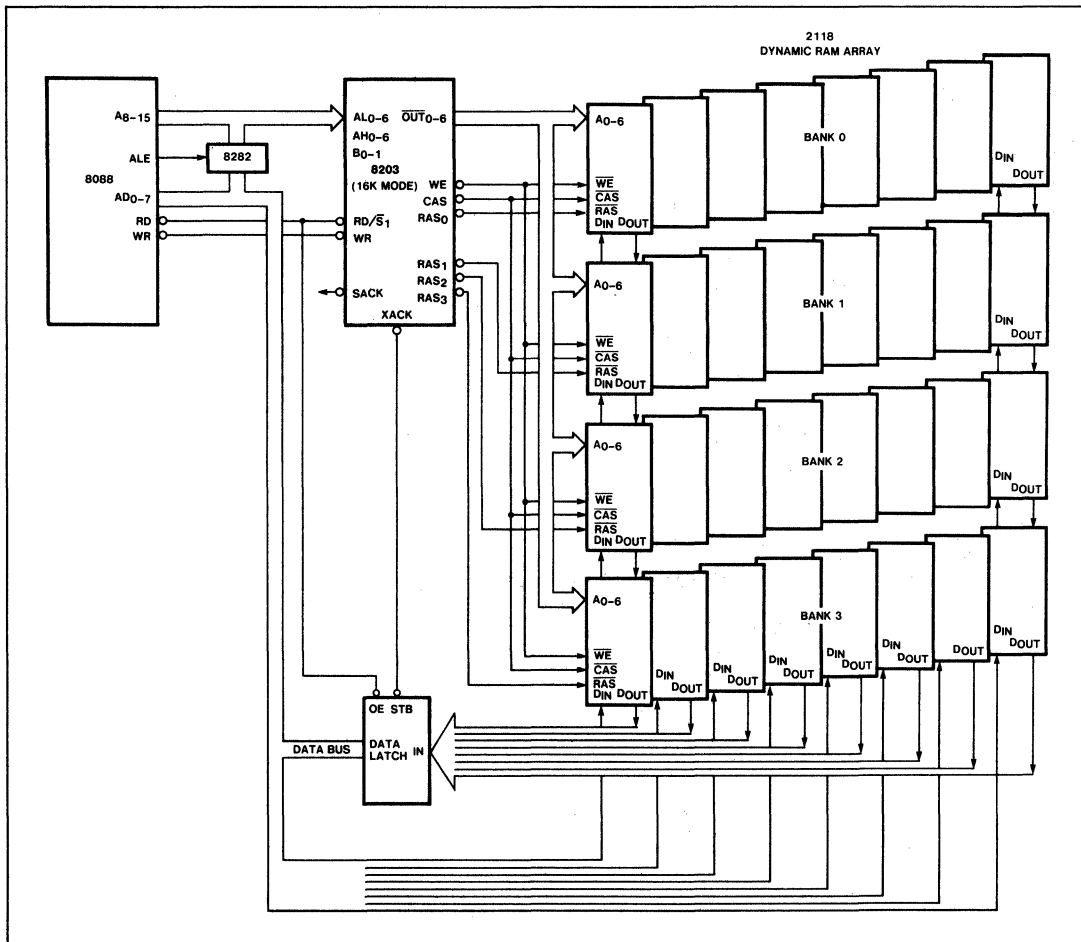


Figure 10. Typical 8088 System



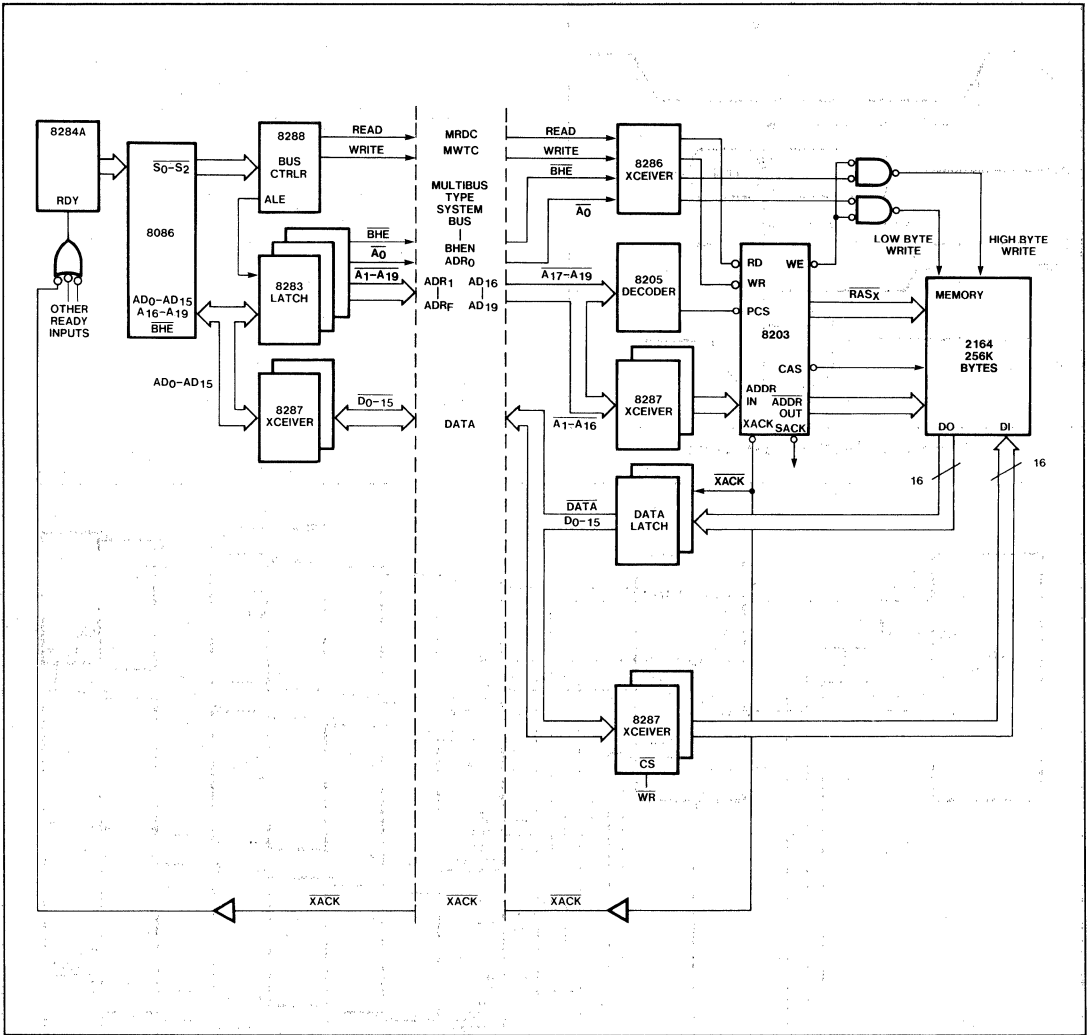


Figure 11. 8086 /256K Byte System

**ABSOLUTE MAXIMUM RATINGS\***

Ambient Temperature Under Bias ..... 0°C to 70°C  
 Storage Temperature ..... -65°C to +150°C  
 Voltage On any Pin  
     With Respect to Ground ..... -0.5V to +7V<sup>4</sup>  
 Power Dissipation ..... 1.5 Watts

*\*NOTE: Stresses above those listed under "Absolute Maximum Ratings" may cause permanent damage to the device. This is a stress rating only and functional operation of the device at these or any other conditions above those indicated in the operational sections of this specification is not implied. Exposure to absolute maximum rating conditions for extended periods may affect device reliability.*

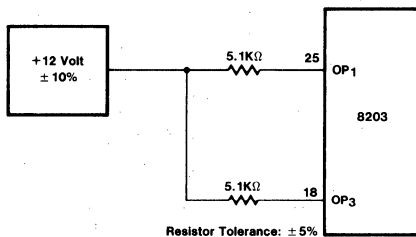
**D.C. CHARACTERISTICS**

$T_A = 0^\circ\text{C to } 70^\circ\text{C}; V_{CC} = 5.0\text{V} \pm 10\%$  (5.0V  $\pm$  5% for 8203-3); GND = 0V

Symbol	Parameter	Min	Max	Units	Test Conditions
V <sub>C</sub>	Input Clamp Voltage		-1.0	V	I <sub>C</sub> = -5 mA
I <sub>CC</sub>	Power Supply Current		270	mA	
I <sub>F</sub>	Forward Input Current CLK All Other Inputs <sup>3</sup>		-2.0 -320	mA $\mu$ A	V <sub>F</sub> = 0.45V V <sub>F</sub> = 0.45V
I <sub>R</sub>	Reverse Input Current <sup>3</sup>		40	$\mu$ A	V <sub>R</sub> = V <sub>CC</sub> (Note 1)
V <sub>OL</sub>	Output Low Voltage SACK, XACK All Other Outputs		0.45 0.45	V V	I <sub>OL</sub> = 5 mA I <sub>OL</sub> = 3 mA
V <sub>OH</sub>	Output High Voltage SACK, XACK All Other Outputs	2.4 2.6		V V	V <sub>IL</sub> = 0.65 V I <sub>OH</sub> = -1 mA I <sub>OH</sub> = -1 mA
V <sub>IL</sub>	Input Low Voltage		0.8	V	V <sub>CC</sub> = 5.0V (Note 2)
V <sub>IH1</sub>	Input High Voltage	2.0	V <sub>CC</sub>	V	V <sub>CC</sub> = 5.0V
V <sub>IH2</sub>	Option Voltage		V <sub>CC</sub>	V	(Note 3)
C <sub>IN</sub>	Input Capacitance		30	pF	F = 1 MHz V <sub>BIAS</sub> = 2.5V, V <sub>CC</sub> = 5V T <sub>A</sub> = 25°C

**NOTES:**

1. I<sub>R</sub> = 200 mA for pin 37 (CLK) for external clock mode.
2. For test mode RD & WR must be held at GND.
3. Except for pin 36 in XTAL mode.
- 4.



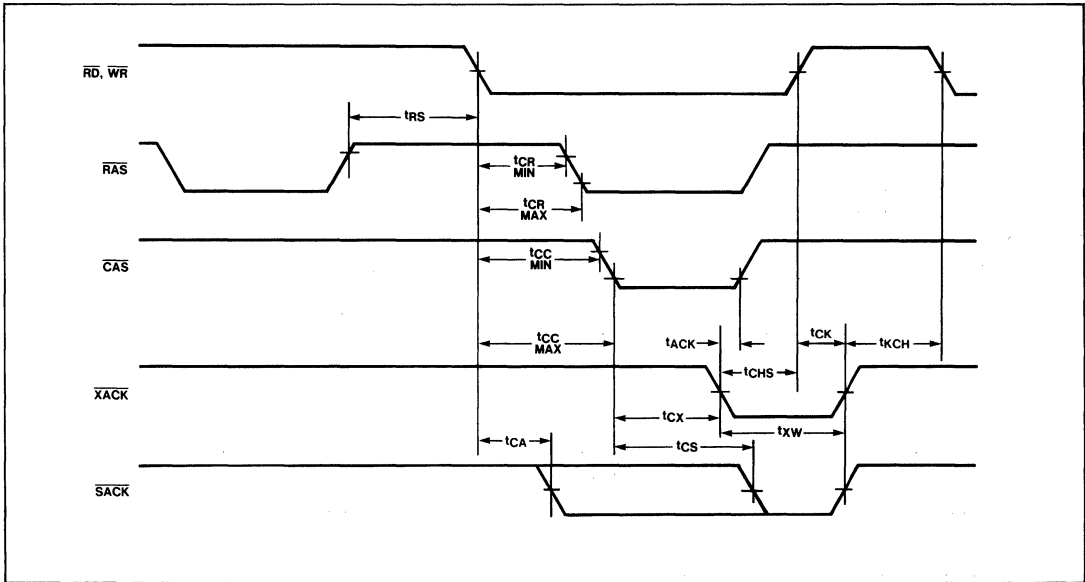
**A.C. CHARACTERISTICS**
 $T_A = 0^\circ\text{C to } 70^\circ\text{C}; V_{CC} = 5\text{V} \pm 10\% (5.0\text{V} \pm 5\% \text{ for } 8203\text{-}3); \text{GND} = 0\text{V}$ 

 Measurements made with respect to  $\overline{\text{RAS}}_0\text{-}\overline{\text{RAS}}_3$ ,  $\overline{\text{CAS}}$ ,  $\overline{\text{WE}}$ ,  $\overline{\text{OUT}}_0\text{-}\overline{\text{OUT}}_6$  are at 2.4V and 0.8V. All other pins are measured at 1.5V. All times are in nsec.

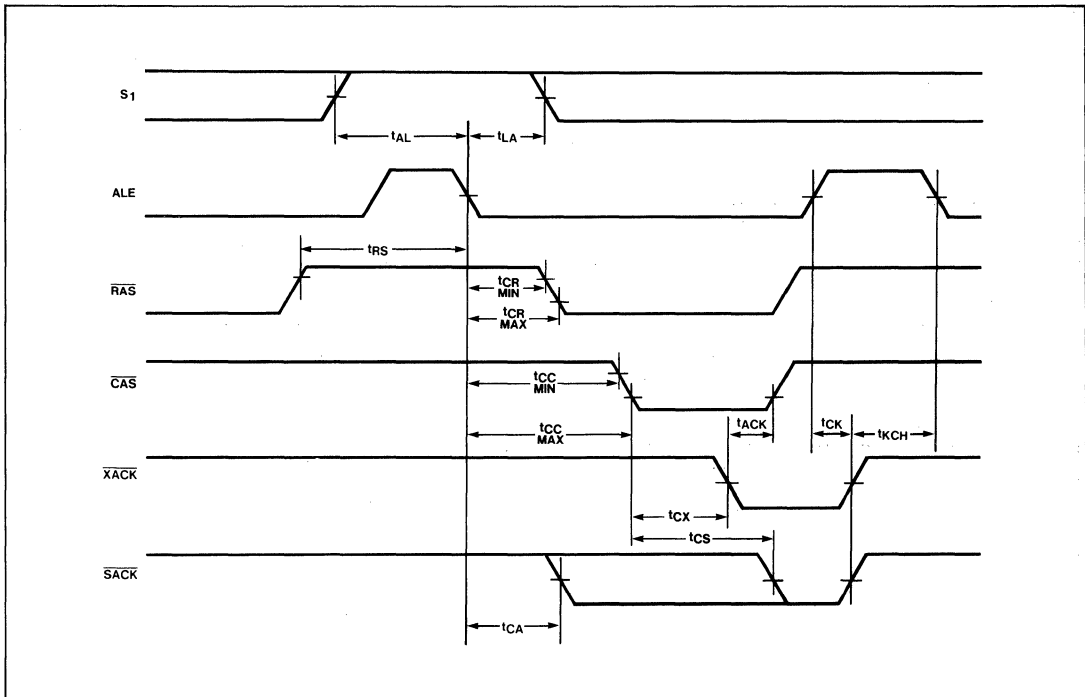
Symbol	Parameter	Min	Max	Notes
$t_p$	Clock Period	40	54	
$t_{PH}$	External Clock High Time	20		
$t_{PL}$	External Clock Low Time—above ( $>$ ) 20 MHz	17		
$t_{PL}$	External Clock Low Time—below ( $\leq$ ) 20 MHz	20		
$t_{RC}$	Memory Cycle Time	10tp - 30	12tp	4, 5
$t_{REF}$	Refresh Time (64 cycles—4K mode)	548tp	576tp	
$t_{REF}$	Refresh Time (128 cycles)	264tp	288tp	
$t_{RP}$	$\overline{\text{RAS}}$ Precharge Time	4tp - 30		
$t_{RSH}$	$\overline{\text{RAS}}$ Hold After $\overline{\text{CAS}}$	5tp - 30		3
$t_{ASR}$	Address Setup to $\overline{\text{RAS}}$	tp - 30		3
$t_{RAH}$	Address Hold From $\overline{\text{RAS}}$	tp - 10		3
$t_{ASC}$	Address Setup to $\overline{\text{CAS}}$	tp - 30		3
$t_{CAH}$	Address Hold from $\overline{\text{CAS}}$	5tp - 20		3
$t_{CAS}$	$\overline{\text{CAS}}$ Pulse Width	5tp - 10		
$t_{WCS}$	$\overline{\text{WE}}$ Setup to $\overline{\text{CAS}}$	tp - 40		
$t_{WCH}$	$\overline{\text{WE}}$ Hold After $\overline{\text{CAS}}$	5tp - 35		8
$t_{RS}$	$\overline{\text{RD}}$ , $\overline{\text{WR}}$ , ALE, REFRQ delay from $\overline{\text{RAS}}$	5tp		2, 6
$t_{MRP}$	$\overline{\text{RD}}$ , $\overline{\text{WR}}$ setup to $\overline{\text{RAS}}$	0		5
$t_{RMS}$	REFRQ setup to $\overline{\text{RD}}$ , $\overline{\text{WR}}$	2tp		6
$t_{RMP}$	REFRQ setup to $\overline{\text{RAS}}$	2tp		5
$t_{PCS}$	$\overline{\text{PCS}}$ Setup to $\overline{\text{RD}}$ , $\overline{\text{WR}}$ , ALE	20		
$t_{AL}$	S1 Setup to ALE	15		
$t_{LA}$	S1 Hold from ALE	30		
$t_{CR}$	$\overline{\text{RD}}$ , $\overline{\text{WR}}$ , ALE to $\overline{\text{RAS}}$ Delay	tp + 30	2tp + 70	2
$t_{CC}$	$\overline{\text{RD}}$ , $\overline{\text{WR}}$ , ALE to $\overline{\text{CAS}}$ Delay	3tp + 25	4tp + 85	2
$t_{SC}$	CMD Setup to Clock	15		1
$t_{MRS}$	$\overline{\text{RD}}$ , $\overline{\text{WR}}$ setup to REFRQ	5		2
$t_{CA}$	$\overline{\text{RD}}$ , $\overline{\text{WR}}$ , ALE to $\overline{\text{SACK}}$ Delay		2tp + 47	2
$t_{CX}$	$\overline{\text{CAS}}$ to $\overline{\text{XACK}}$ Delay	5tp - 25	5tp + 20	
$t_{CS}$	$\overline{\text{CAS}}$ to $\overline{\text{SACK}}$ Delay	5tp - 25	5tp + 40	2
$t_{ACK}$	$\overline{\text{XACK}}$ to $\overline{\text{CAS}}$ Setup	10		
$t_{XW}$	$\overline{\text{XACK}}$ Pulse Width	tp - 25		
$t_{CK}$	$\overline{\text{SACK}}$ , $\overline{\text{XACK}}$ turn-off Delay		35	
$t_{KCH}$	CMD Inactive Hold after $\overline{\text{SACK}}$ , $\overline{\text{XACK}}$	10		
$t_{LL}$	REFRQ Pulse Width	20		
$t_{CHS}$	CMD Hold Time	30		
$t_{RFR}$	REFRQ to $\overline{\text{RAS}}$ Delay		4tp + 100	6
$t_{WW}$	$\overline{\text{WR}}$ to $\overline{\text{WE}}$ Delay	0	50	8
$t_{AD}$	CPU Address Delay	0	40	3

WAVEFORMS

Normal Read or Write Cycle

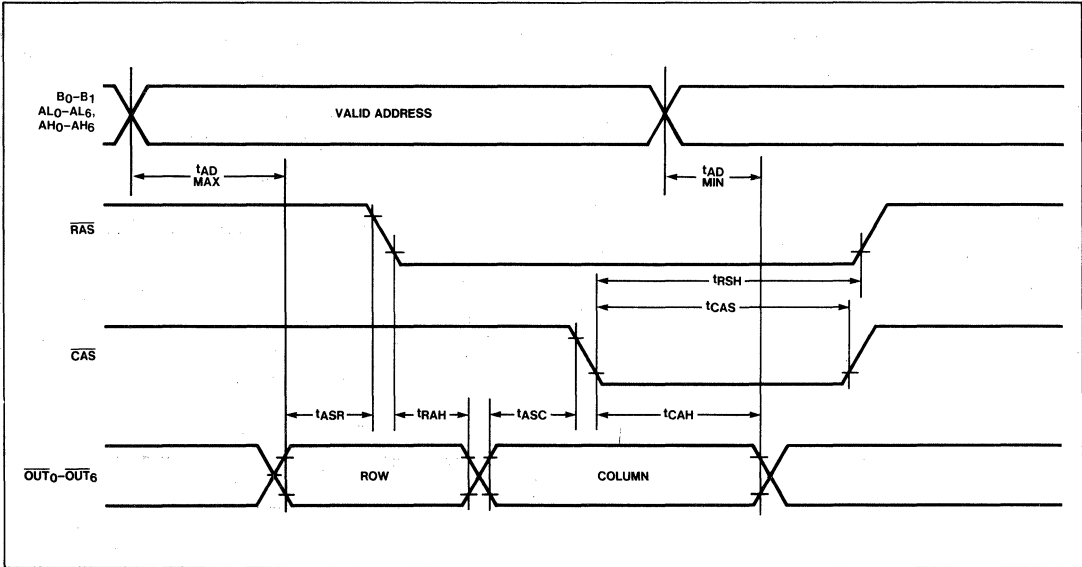


Advanced Read Mode

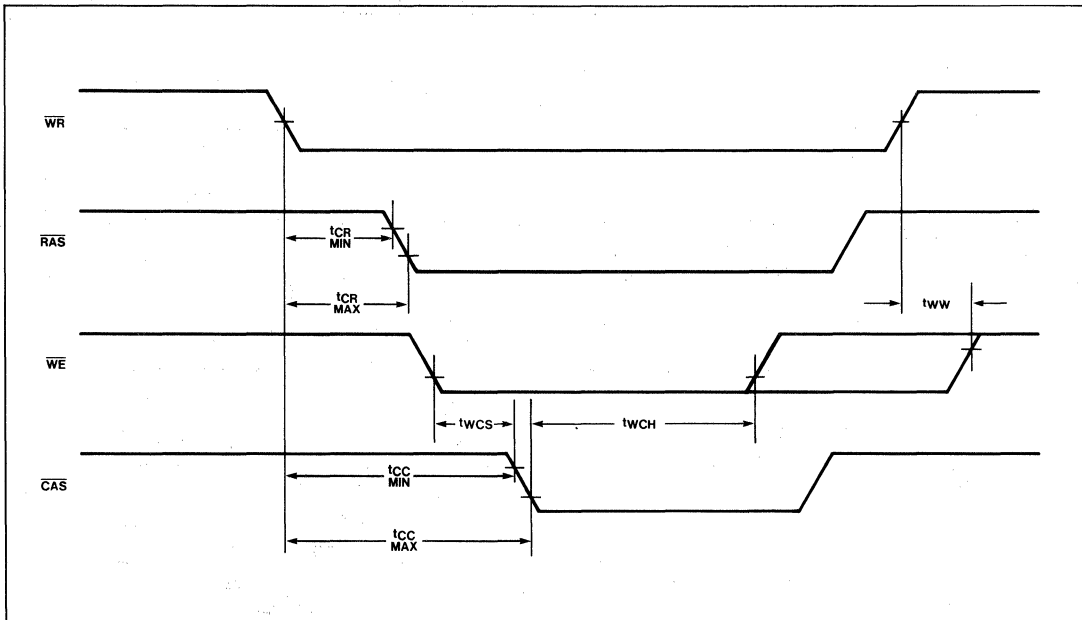


WAVEFORMS (cont'd)

Memory Compatibility Timing

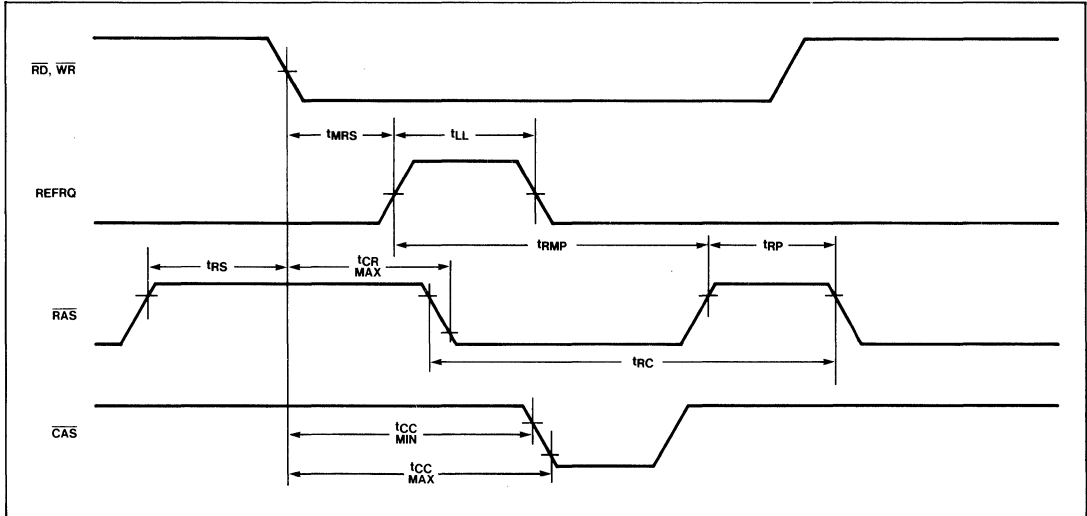


Write Cycle Timing

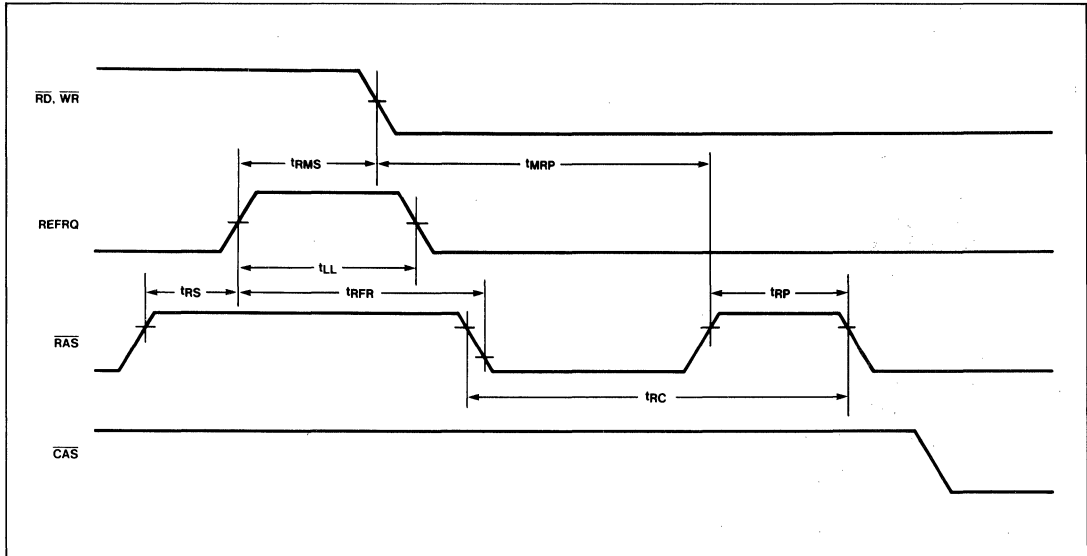


WAVEFORMS (cont'd)

Read or Write Followed By External Refresh

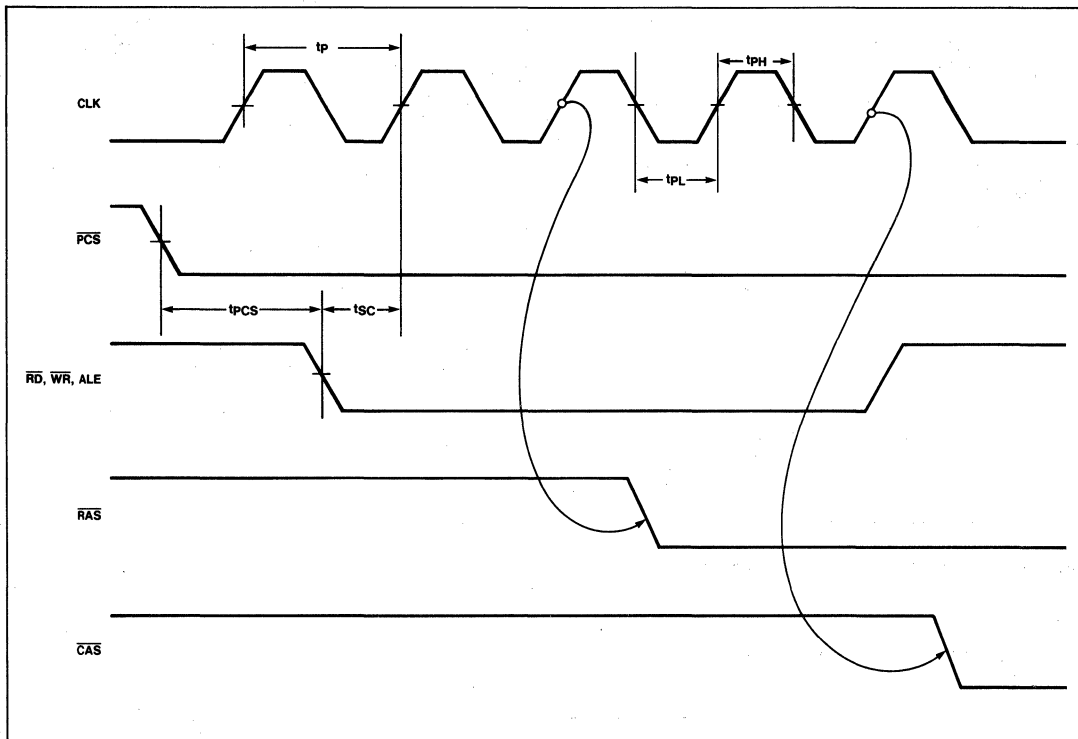


External Refresh Followed By Read or Write



WAVEFORMS (cont'd)

Clock And System Timing



**Table 2 8203 Output Loading.**  
**All specifications are for the Test Load unless otherwise noted.**

Pin	Test Load
SACK, XACK	$C_L = 30 \text{ pF}$
OUT <sub>0</sub> -OUT <sub>6</sub>	$C_L = 160 \text{ pF}$
RAS <sub>0</sub> -RAS <sub>3</sub>	$C_L = 60 \text{ pF}$
WE	$C_L = 224 \text{ pF}$
CAS	$C_L = 320 \text{ pF}$

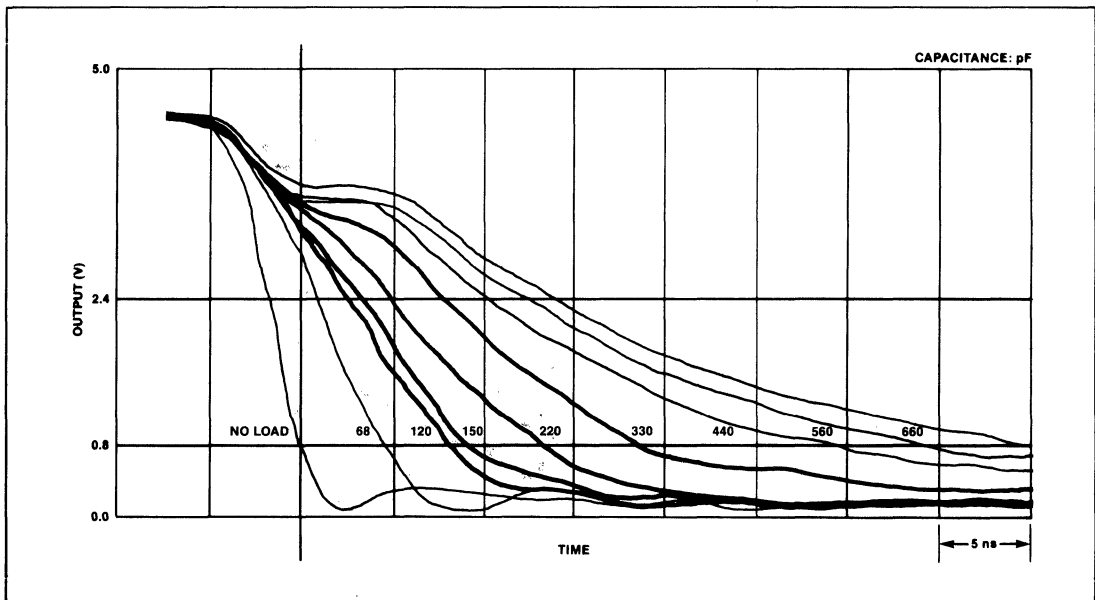
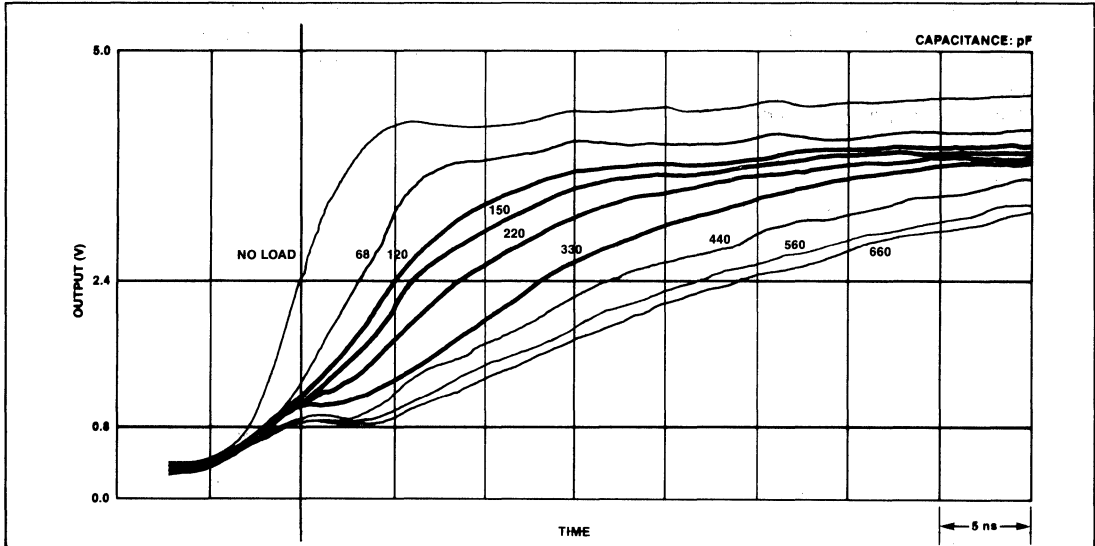
**NOTES:**

- $t_{SC}$  is a reference point only. ALE,  $\overline{RD}$ ,  $\overline{WR}$ , and REFREQ inputs do not have to be externally synchronized to 8203 clock.
- If  $t_{RS}$  min and  $t_{MS}$  min are met then  $t_{CA}$ ,  $t_{CR}$ , and  $t_{CC}$  are valid, otherwise  $t_{CS}$  is valid.
- $t_{ASR}$ ,  $t_{RAH}$ ,  $t_{ASC}$ ,  $t_{CAH}$ , and  $t_{RSH}$  depend upon B0-B1 and CPU address remaining stable throughout the memory cycle. The address inputs are not latched by the 8203.
- For back-to-back refresh cycles,  $t_{RC}$  max = 13tp
- $t_{RC}$  max is valid only if  $t_{RMP}$  min is met (READ, WRITE followed by REFRESH) or  $t_{MPP}$  min is met (REFRESH followed by READ, WRITE).
- $t_{RRF}$  is valid only if  $t_{RS}$  min and  $t_{MS}$  min are met.
- $t_{XW}$  min applies when  $\overline{RD}$ ,  $\overline{WR}$  has already gone high. Otherwise XACK follows  $\overline{RD}$ ,  $\overline{WR}$ .
- $\overline{WE}$  goes high according to  $t_{WCH}$  or  $t_{WW}$ , whichever occurs first.

The typical rising and falling characteristic curves can be used to determine the effects of capacitive loading on the A.C. Timing Parameters. Using this

design tool in conjunction with the timing waveforms, the designer can determine typical timing shifts based on system capacitive load.

**A.C. CHARACTERISTICS FOR DIFFERENT CAPACITIVE LOADS**



**NOTE:**  
Use the Test Load as the base capacitance for estimating timing shifts for system critical timing parameters.

**TYPICAL CONDITIONS:**  
 $T_A = 25^\circ\text{C}$  Pins not under test are loaded with Test Load capacitance.  
 $V_{CC} = +5\text{V}$   
 $t_p = 50\text{ ns}$



Example: Find the effect on  $t_{CR}$  and  $t_{CC}$  using 32 2164 Dynamic RAMs configured in 2 banks.

1. Determine the typical RAS and CAS capacitance:  
From the data sheet RAS = 5 pF and CAS = 5 pF.  
 $\therefore$  RAS load = 80 pF + board capacitance.  
CAS load = 160 pF + board capacitance.  
Assume 2 pF/in (trace length) for board capacitance.
2. From the waveform diagrams, we determine that the falling edge timing is needed for  $t_{CR}$  and  $t_{CC}$ .

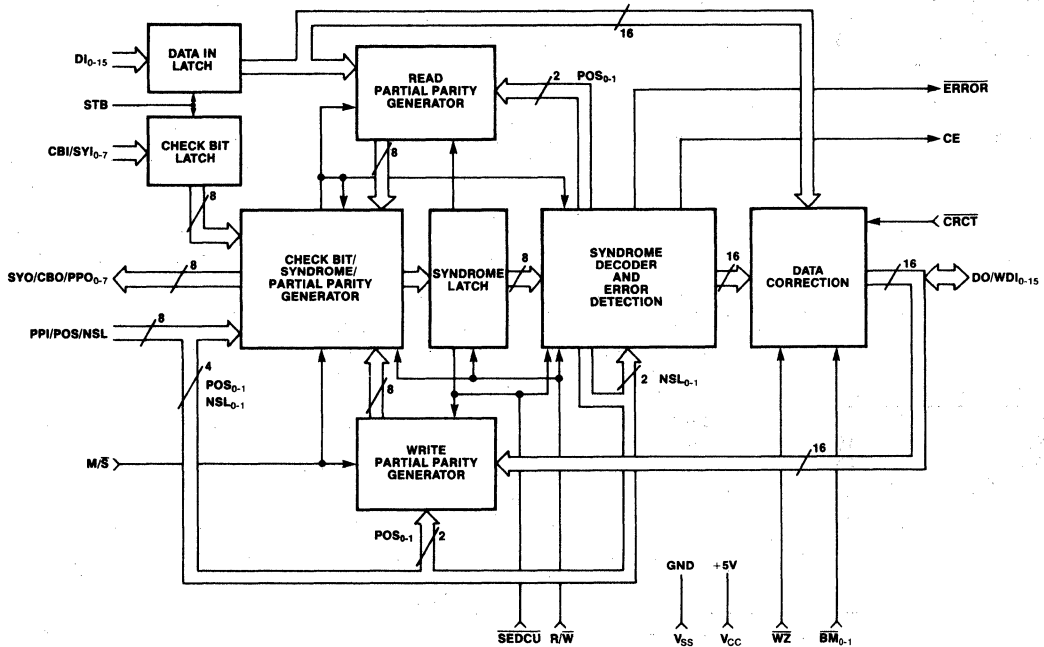
Next find the curve that *best* approximates the test load; i.e., 68 pF for RAS and 330 pF for CAS.

3. If we use 88 pF for RAS loading, then  $t_{CR}$  (min.) spec should be increased by about 1 ns, and  $t_{CR}$  (max.) spec should be increased by *about* 2 ns. Similarly if we use 176 pF for CAS, then  $t_{CC}$  (min.) should decrease by 3 ns and  $t_{CC}$  (max.) should decrease about 7 ns.

## 8206 ERROR DETECTION AND CORRECTION UNIT

- Detects and Corrects All Single Bit Errors
- Detects All Double Bit and Most Multiple Bit Errors
- 52 ns Maximum for Detection; 67 ns Maximum for Correction (16 Bit System)
- Expandable to Handle 80 Bit Memories
- Syndrome Outputs for Error Logging
- Separate Input and Output Busses—No Timing Strobes Required
- Supports Reads With and Without Correction, Writes, Partial (Byte) Writes, and Read-Modify-Writes
- HMOS Technology for Low Power
- 68 Pin Leadless JEDEC Package
- Single +5V Supply

The HMOS 8206 Error Detection and Correction Unit is a high-speed device that provides error detection and correction for memory systems (static and dynamic) requiring high reliability and performance. Each 8206 handles 8 or 16 data bits and up to 8 check bits. 8206's can be cascaded to provide correction and detection for up to 80 bits of data. Other 8206 features include the ability to handle byte writes, memory initialization, and error logging.



**Figure 1. 8206 Block Diagram**

Table 1. Pin Description

Symbol	Pin No.	Type	Name and Function
DI <sub>0-15</sub>	1, 68-61, 59-53	I	<b>Data In:</b> These inputs accept a 16 bit data word from RAM for error detection and/or correction.
CBI/SYI <sub>0</sub> CBI/SYI <sub>1</sub> CBI/SYI <sub>2</sub> CBI/SYI <sub>3</sub> CBI/SYI <sub>4</sub> CBI/SYI <sub>5</sub> CBI/SYI <sub>6</sub> CBI/SYI <sub>7</sub>	5 6 7 8 9 10 11 12	I I I I I I I I	<b>Check Bits In/Syndrome In:</b> In a single 8206 system, or in the master in a multi-8206 system, these inputs accept the check bits (5 to 8) from the RAM. In a single 8206 16 bit system, CBI <sub>0-5</sub> are used. In slave 8206's these inputs accept the syndrome from the master, with the syndrome latched by R/W going low.
DO/WDI <sub>0</sub> DO/WDI <sub>1</sub> DO/WDI <sub>2</sub> DO/WDI <sub>3</sub> DO/WDI <sub>4</sub> DO/WDI <sub>5</sub> DO/WDI <sub>6</sub> DO/WDI <sub>7</sub> DO/WDI <sub>8</sub> DO/WDI <sub>9</sub> DO/WDI <sub>10</sub> DO/WDI <sub>11</sub> DO/WDI <sub>12</sub> DO/WDI <sub>13</sub> DO/WDI <sub>14</sub> DO/WDI <sub>15</sub>	51 50 49 48 47 46 45 44 42 41 40 39 38 37 36 35	I/O I/O I/O I/O I/O I/O I/O I/O I/O I/O I/O I/O I/O I/O I/O I/O	<b>Data Out/Write Data In:</b> In a read cycle, data accepted by DI <sub>0-15</sub> appears at these outputs corrected if CRCT is low, or uncorrected if CRCT is high. The BM inputs must be high to enable the output buffers during the read cycle. In a write cycle, data to be written into the RAM is accepted by these inputs for computing the write check bits. In a partial-write cycle, the byte not to be modified appears at either DO <sub>0-7</sub> if BM <sub>0</sub> is high, or DO <sub>8-15</sub> if BM <sub>1</sub> is high, for writing to the RAM. When WZ is active, it causes the 8206 to output all zeros at DO <sub>0-15</sub> , with the proper write check bits on CBO.
SYO/CBO/PPO <sub>0</sub> SYO/CBO/PPO <sub>1</sub> SYO/CBO/PPO <sub>2</sub> SYO/CBO/PPO <sub>3</sub> SYO/CBO/PPO <sub>4</sub> SYO/CBO/PPO <sub>5</sub> SYO/CBO/PPO <sub>6</sub> SYO/CBO/PPO <sub>7</sub>	23 24 25 27 28 29 30 31	O O O O O O O O	<b>Syndrome Out/Check Bits Out/Partial Parity Out:</b> In a single 8206 system, or in the master in a multi-8206 system, the syndrome appears at these outputs during a read. During a write, the write check bits appear. In slave 8206's the partial parity bits used by the master appear at these outputs. The syndrome is latched (during read-modify-writes) by R/W going low.
PPI <sub>0</sub> /POS <sub>0</sub> PPI <sub>1</sub> /POS <sub>1</sub>	13 14	I I	<b>Partial Parity In/Position:</b> In the master in a multi-8206 system, these inputs accept partial parity bits 0 and 1 from the slaves. In a slave 8206 these inputs inform it of its position within the system (1 to 4). Not used in a single 8206 system.
PPI <sub>2</sub> /NSL <sub>0</sub> PPI <sub>3</sub> /NSL <sub>1</sub>	15 16	I I	<b>Partial Parity In/Number of Slaves:</b> In the master in a multi-8206 system, these inputs accept partial parity bits 2 and 3 from the slaves. In a multi-8206 system these inputs are used in slave number 1 to tell it the total number of slaves in the system (1 to 4). Not used in other slaves or in a single 8206 system.
PPI <sub>4</sub> /CE	17	I/O	<b>Partial Parity In/Correctable Error:</b> In the master in a multi-8206 system this pin accepts partial parity bit 4. In slave number 1 only, or in a single 8206 system, this pin outputs the correctable error flag. CE is latched by R/W going low. Not used in other slaves.
PPI <sub>5</sub> PPI <sub>6</sub> PPI <sub>7</sub>	18 19 20	I I I	<b>Partial Parity In:</b> In the master in a multi-8206 system these pins accept partial parity bits 5 to 7. The number of partial parity bits equals the number of check bits. Not used in single 8206 systems or in slaves.
ERROR	22	O	<b>Error:</b> This pin outputs the error flag in a single 8206 system or in the master of a multi-8206 system. It is latched by R/W going low. Not used in slaves.
CRCT	52	I	<b>Correct:</b> When low this pin causes data correction during a read or read-modify-write cycle. When high, it causes error correction to be disabled, although error checking is still enabled.
STB	2	I	<b>Strobe:</b> STB is an input control used to strobe data at the DI inputs and check-bits at the CBI/SYI inputs. The signal is active high to admit the inputs. The signals are latched by the high-to-low transition of STB.

**Table 1. Pin Description (Continued)**

Symbol	Pin No.	Type	Name and Function
$\overline{BM}_0$	33	I	<b>Byte Marks:</b> When high, the Data Out pins are enabled for a read cycle. When low, the Data Out buffers are tristated for a write cycle. $\overline{BM}_0$ controls $DO_{0-7}$ , while $\overline{BM}_1$ controls $DO_{8-15}$ . In partial (byte) writes, the byte mark input is low for the new byte to be written.
$\overline{BM}_1$	32	I	
$R/\overline{W}$	21	I	<b>Read/Write:</b> When high this pin causes the 8206 to perform detection and correction (if CRCT is low). When low, it causes the 8206 to generate check bits. On the high-to-low transition the syndrome is latched internally for read-modify-write cycles.
$\overline{WZ}$	34	I	<b>Write Zero:</b> When low this input overrides the $\overline{BM}_{0-1}$ and $R/\overline{W}$ inputs to cause the 8206 to output all zeros at $DO_{0-15}$ with the corresponding check bits at $CBO_{0-7}$ . Used for memory initialization.
$M/\overline{S}$	4	I	<b>Master/Slave:</b> Input tells the 8206 whether it is a master (high) or a slave (low).
$\overline{SEDCU}$	3	I	<b>Single EDC Unit:</b> Input tells the master whether it is operating as a single 8206 (low) or as the master in a multi-8206 system (high). Not used in slaves.
$V_{CC}$	60	I	<b>Power Supply:</b> +5V
$V_{SS}$	26, 43	I	<b>Ground</b>

**FUNCTIONAL DESCRIPTION**

The 8206 Error Detection and Correction Unit provides greater memory system reliability through its ability to detect and correct memory errors. It is a single chip device that can detect and correct all single bit errors and detect all double bit and some higher multiple bit errors. Some other odd multiple bit errors (e.g., 5 bits in error) are interpreted as single bit errors, and the CE flag is raised. While some even multiple bit errors (e.g., 4 bits in error) are interpreted as no error, most are detected as double bit errors. This error handling is a function of the number of check bits used by the 8206 (see Figure 2) and the specific Hamming code used. Errors in check bits are not distinguished from errors in a word.

For more information on error correction codes, see Intel Application Notes AP-46 and AP-73.

A single 8206 handles 8 or 16 bits of data, and up to 5 8206's can be cascaded in order to handle data paths of 80 bits. For a single 8206 8 bit system, the  $DI_{8-15}$ ,  $DO/WDI_{8-15}$  and  $\overline{BM}_1$  inputs are grounded. See the Multi-Chip systems section for information on 24-80 bit systems.

The 8206 has a "flow through" architecture. It supports two kinds of error correction architecture: 1) Flow-through, or correct-always; and 2) Parallel, or check-only. There are two separate 16-pin busses,

DATA WORD BITS	CHECK BITS
8	5
16	6
24	6
32	7
40	7
48	8
56	8
64	8
72	8
80	8

**Figure 2. Number of Check Bits Used by 8206**

one to accept data from the RAM (DI) and the other to deliver corrected data to the system bus (DO/WDI). The logic is entirely combinatorial during a read cycle. This is in contrast to an architecture with only one bus, with bidirectional bus drivers that must first read the data and then be turned around to output the corrected data. The latter architecture typically requires additional hardware (latches and/or transceivers) and may be slower in a system due to timing skews of control signals.

## READ CYCLE

With the  $R/\overline{W}$  pin high, data is received from the RAM outputs into the DI pins where it is optionally latched by the STB signal. Check bits are generated from the data bits and compared to the check bits read from the RAM into the CBI pins. If an error is detected the ERROR flag is activated and the correctable error flag (CE) is used to inform the system whether the error was correctable or not. With the  $\overline{BM}$  inputs high, the word appears corrected at the DO pins if the error was correctable, or unmodified if the error was uncorrectable.

If more than one 8206 is being used, then the check bits are read by the master. The slaves generate a partial parity output (PPO) and pass it to the master. The master 8206 then generates and returns the syndrome to the slaves (SYO) for correction of the data.

The 8206 may alternatively be used in a "check-only" mode with the CRCT pin left high. With the correction facility turned off, the propagation delay from memory outputs to 8206 outputs is significantly shortened. In this mode the 8206 issues an ERROR flag to the CPU, which can then perform one of several options: lengthen the current cycle for correction, restart the instruction, perform a diagnostic routine, etc.

A syndrome word, five to eight bits in length and containing all necessary information about the existence and location of an error, is made available to the system at the SYO<sub>0-7</sub> pins. Error logging may be accomplished by latching the syndrome and the memory address of the word in error.

## WRITE CYCLE

For a full write, in which an entire word is written to memory, the data is written directly to the RAM, bypassing the 8206. The same data enters the 8206 through the WDI pins where check bits are generated. The Byte Mark inputs must be low to tristate the DO drivers. The check bits, 5 to 8 in number, are then written to the RAM through the CBO pins for storage along with the data word. In a multi-chip system, the master writes the check bits using partial parity information from the slaves.

In a partial write, part of the data word is overwritten, and part is retained in memory. This is accomplished by performing a read-modify-write cycle. The complete old word is read into the 8206 and corrected,

with the syndrome internally latched by  $R/\overline{W}$  going low. Only that part of the word not to be modified is output onto the DO pins, as controlled by the Byte Mark inputs. That portion of the word to be overwritten is supplied by the system bus. The 8206 then calculates check bits for the new word, using the byte from the previous read and the new byte from the system bus, and writes them to the memory.

## READ-MODIFY-WRITE CYCLES

Upon detection of an error the 8206 may be used to correct the bit in error in memory. This reduces the probability of getting multiple-bit errors in subsequent read cycles. This correction is handled by executing read-modify-write cycles.

The read-modify-write cycle is controlled by the  $R/\overline{W}$  input. After (during) the read cycle, the system dynamic RAM controller or CPU examines the 8206 ERROR and CE outputs to determine if a correctable error occurred. If it did, the dynamic RAM controller or CPU forces  $R/\overline{W}$  low, telling the 8206 to latch the generated syndrome and drive the corrected check bits onto the CBO outputs. The corrected data is available on the DO pins. The DRAM controller then writes the corrected data and corresponding check bits into memory.

The 8206 may be used to perform read-modify-writes in one or two RAM cycles. If it is done in two cycles, the 8206 latches are used to hold the data and check bits from the read cycle to be used in the following write cycle. The Intel 8207 Advanced Dynamic RAM controller allows read-modify-write cycles in one memory cycle. See the System Environment section.

## INITIALIZATION

A memory system operating with ECC requires some form of initialization at system power-up in order to set valid data and check bit information in memory. The 8206 supports memory initialization by the write zero function. By activating the  $\overline{WZ}$  pin, the 8206 will write a data pattern of zeros and the associated check bits in the current write cycle. By thus writing to all memory at power-up, a controller can set memory to valid data and check bits. Massive memory failure, as signified by both data and check bits all ones or zeros, will be detected as an uncorrectable error.

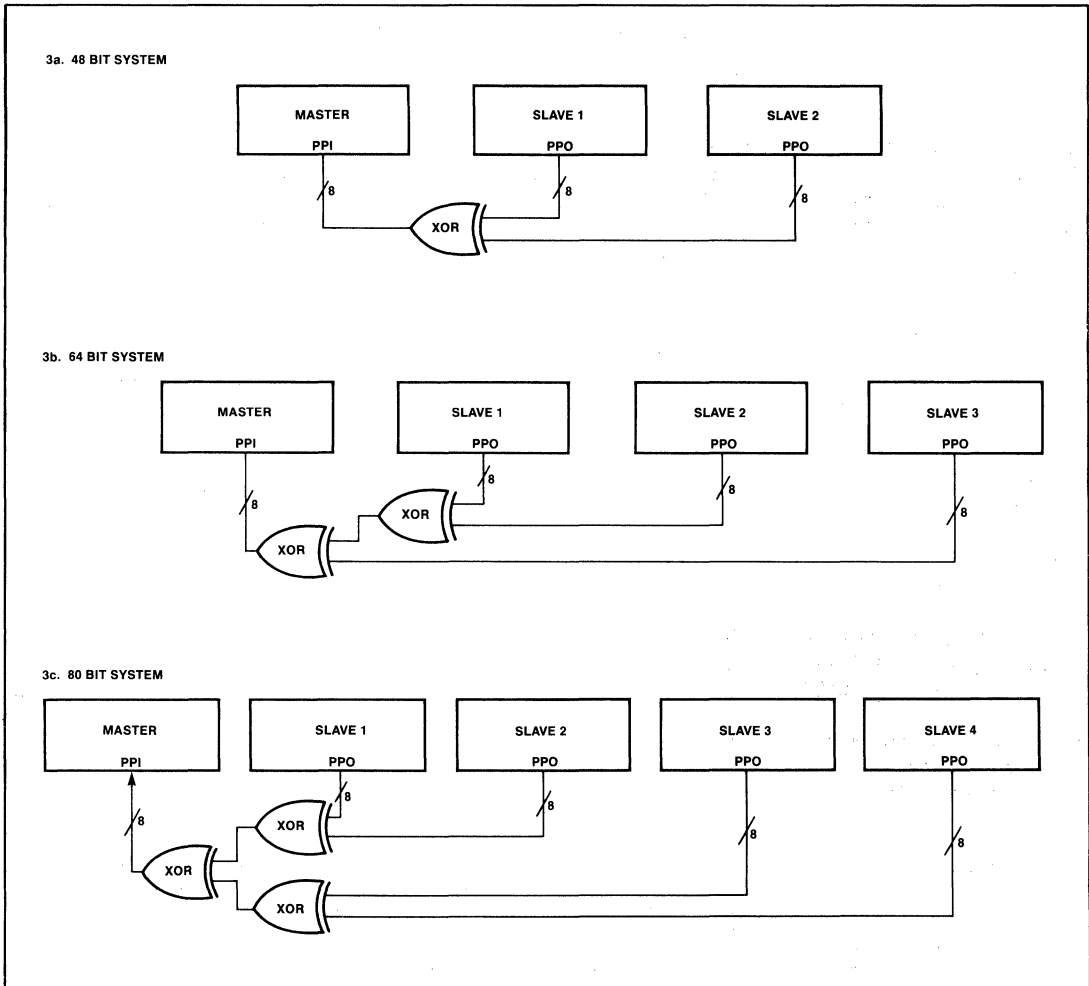
**MULTI-CHIP SYSTEMS**

A single 8206 handles 8 or 16 bits of data and 5 or 6 check bits, respectively. Up to 5 8206's can be cascaded for 80 bit memories with 8 check bits.

When cascaded, one 8206 operates as a master, and all others as slaves. As an example, during a read cycle in a 32 bit system with one master and one slave, the slave calculates parity on its portion of the word—"partial parity"—and presents it to the master through the PPO pins. The master combines the partial parity from the slave with the parity it calculated from its own portion of the word to generate

the syndrome. The syndrome is then returned by the master to the slave for error correction. In systems with more than one slave the above description continues to apply, except that the partial parity outputs of the slaves must be XOR'd externally. Figure 3 shows the necessary external logic for multi-chip systems. Write and read-modify-write cycles are carried out analogously. See the System Operation section for multi-chip wiring diagrams.

There are several pins used to define whether the 8206 will operate as a master or a slave. Tables 2 and 3 illustrate how these pins are tied.



**Figure 3. External Logic For Multi-Chip Systems**

**Table 2. Master/Slave Pin Assignments**

Pin No.	Pin Name	Master	Slave 1	Slave 2	Slave 3	Slave 4
4	M/ $\bar{S}$	+5V	Gnd	Gnd	Gnd	Gnd
3	SEDCU	+5V	+5V	+5V	+5V	+5V
13	PPI <sub>0</sub> /POS <sub>0</sub>	PPI	Gnd	+5V	Gnd	+5V
14	PPI <sub>1</sub> /POS <sub>1</sub>	PPI	Gnd	Gnd	+5V	+5V
15	PPI <sub>2</sub> /NSL <sub>0</sub>	PPI	*	+5V	+5V	+5V
16	PPI <sub>3</sub> /NSL <sub>1</sub>	PPI	*	+5V	+5V	+5V

\*See Table 3.

**Table 3. NSL Pin Assignments for Slave 1**

Pin	Number of Slaves			
	1	2	3	4
PPI <sub>2</sub> /NSL <sub>0</sub>	Gnd	+5V	Gnd	+5V
PPI <sub>3</sub> /NSL <sub>1</sub>	Gnd	Gnd	+5V	+5V

The timing specifications for multi-chip systems must be calculated to take account of the external XOR gating in 3, 4, and 5-chip systems. Let tXOR be the delay for a single external TTL XOR gate. Then the following equations show how to calculate the relevant timing parameters for 2-chip (n=0), 3-chip (n=1), 4-chip (n=2), and 5-chip (n=2) systems:

$$\text{Data-in to corrected data-out (read cycle)} = \text{TDVSV} + \text{TPVSV} + \text{TSVQV} + \text{ntXOR}$$

$$\text{Data-in to error flag (read cycle)} = \text{TDVSV} + \text{TPVEV} + \text{ntXOR}$$

$$\text{Data-in to correctable error flag (read cycle)} = \text{TDVSV} + \text{TPVSV} + \text{TSVQV} + \text{ntXOR}$$

$$\text{Write data to check-bits valid (full write cycle)} = \text{TQVQV} + \text{TPVSV} + \text{ntXOR}$$

$$\text{Data-in to check-bits valid (read-mod-write cycle)} = \text{TDVSV} + \text{TPVSV} + \text{TSVQV} + \text{TQVQV} + \text{TPVSV} + 2\text{ntXOR}$$

$$\text{Data-in to check-bits valid (non-correcting read-modify-write cycle)} = \text{TDVQU} + \text{TQVQV} + \text{TPVSV} + \text{ntXOR}$$

### HAMMING CODE

The 8206 uses a modified Hamming code which was optimized for multi-chip EDCU systems. The code is such that partial parity is computed by all 8206's in

parallel. No 8206 requires more time for propagation through logic levels than any other one, and hence no one device becomes a bottleneck in the parity operation. However, one or two levels of external TTL XOR gates is required in systems with three to five chips. The code appears in Table 4. The check bits are derived from the table by XORing or XNORing together the bits indicated by 'X's in each row corresponding to a check bit. For example, check bit 0 in the MASTER for data word 1000110101101011 will be "0." It should be noted that the 8206 will detect the gross-error condition of all lows or all highs.

Error correction is accomplished by identifying the bad bit and inverting it. Table 4 can also be used as an error syndrome table by replacing the 'X's with '1's. Each column then represents a different syndrome word, and by locating the column corresponding to a particular syndrome the bit to be corrected may be identified. If the syndrome cannot be located then the error cannot be corrected. For example, if the syndrome word is 00110111, the bit to be corrected is bit 5 in the slave one data word (bit 21).

The syndrome decoding is also summarized in Table 5, which can be used for error logging. By finding the appropriate syndrome word (starting with bit zero, the least significant bit), the result is either: 1) no error; 2) an identified (correctable) single bit error; 3) a double bit error; or 4) a multi-bit uncorrectable error.

Table 4. Modified Hamming Code Check Bit Generation

Check bits are generated by XOR'ing (except for the CB0 and CB1 data bits, which are XNOR'ed in the Master) the data bits in the rows corresponding to the check bits. Note there are 6 check bits in a 16-bit system, 7 in a 32-bit system, and 8 in 48-or-more-bit systems.

BYTE NUMBER	0							1							OPERATION	2							3							4							5																					
BIT NUMBER	0	1	2	3	4	5	6	7	0	1	2	3	4	5		6	7	0	1	2	3	4	5	6	7	0	1	2	3	4	5	6	7	0	1	2	3	4	5	6	7	0	1	2	3	4	5	6	7									
CHECK BITS	CB0 =	x	x	-	x	-	x	x	-	x	-	-	x	-	x	-	-	XNOR	-	x	x	x	-	x	x	-	-	x	x	-	-	x	-	-	x	x	-	x	-	x	x	-	x	-	-	x	-	x	-	-	x	-	-	x	-	x	-	-
	CB1 =	x	-	x	-	-	x	-	x	-	x	-	x	x	-	x	-	XNOR	x	x	x	-	-	x	-	x	x	x	-	-	-	-	x	-	x	-	x	-	-	x	-	x	x	-	x	-	-	x	-	x	x	-	x	-	-	x	-	x
	CB2 =	-	x	x	-	x	-	x	x	-	-	x	-	x	-	-	x	XOR	-	x	x	x	-	x	x	x	-	-	x	x	-	-	-	-	-	x	x	-	x	-	x	x	-	-	x	-	x	-	x	x	-	-	x	-	x	-	x	x
	CB3 =	x	x	x	x	x	-	-	-	x	x	x	-	-	-	-	-	XOR	x	x	x	-	x	x	x	x	x	-	-	x	-	x	-	-	x	x	x	x	x	-	-	-	x	x	x	x	x	-	-	-	x	x	x	x	x	-	-	-
	CB4 =	-	-	-	x	x	x	x	x	-	-	-	-	-	-	-	-	XOR	x	x	-	-	x	x	x	x	-	-	-	-	-	x	-	-	-	-	x	x	x	x	-	-	-	-	x	x	x	x	-	-	-	-	x	x	x	x	-	-
	CB5 =	-	-	-	-	-	-	-	-	x	x	x	x	x	x	x	x	XOR	-	-	-	x	x	x	x	x	-	-	-	-	-	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	-	-	-	-	-	-	-	-
	CB6 =	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	XOR	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	x	x	x	x	x	x	x	x	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
	CB7 =	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	XOR	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	x	x	x	x	x	x	x	x	-	-	-	-	-	-	-	-
DATA BITS	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1		1	1	1	1	2	2	2	2	2	2	2	2	2	2	3	3	3	3	3	3	3	3	3	3	4	4	4	4	4	4	4	4	0	1	2	3	4	5	6	7	

16 BIT OR MASTER

SLAVE #1

SLAVE #2

BYTE NUMBER	6							7							8							9							OPERATION				
BIT NUMBER	0	1	2	3	4	5	6	7	0	1	2	3	4	5	6	7	0	1	2	3	4	5	6	7	0	1	2	3		4	5	6	7
CHECK BITS	CB0 =	x	-	x	-	x	x	-	x	-	x	x	-	-	x	-	-	x	x	x	-	x	x	-	-	x	x	-	-	x	-	-	XOR
	CB1 =	-	x	x	-	-	x	x	x	x	x	-	-	x	-	x	-	x	x	x	-	x	x	x	-	-	x	x	-	-	-	-	XOR
	CB2 =	-	x	x	x	-	x	x	-	x	x	-	x	-	-	x	-	x	-	-	x	x	-	-	-	x	x	-	-	-	-	-	XOR
	CB3 =	x	-	x	-	-	x	x	x	x	-	-	x	x	-	-	-	x	x	x	x	-	-	x	x	x	-	-	x	-	-	-	XOR
	CB4 =	-	-	-	-	x	x	x	x	x	x	-	-	x	x	x	-	-	-	-	x	x	x	-	-	x	x	-	-	-	-	-	XOR
	CB5 =	-	-	-	-	-	-	-	-	x	x	x	x	x	x	x	x	x	x	x	x	x	-	x	-	-	-	-	-	-	-	-	XOR
	CB6 =	x	x	x	x	x	x	x	-	-	-	-	-	-	-	-	-	x	x	-	-	x	x	x	x	-	-	-	-	-	-	-	XOR
	CB7 =	-	-	-	-	-	-	-	x	x	x	x	x	x	x	-	-	-	-	-	-	-	-	-	x	x	x	x	x	x	x	XOR	
DATA BITS	4	4	5	5	5	5	5	5	5	5	6	6	6	6	6	6	6	6	6	7	7	7	7	7	7	7	7	7	7	7			

SLAVE #3

SLAVE #4



Table 5. Syndrome Decoding

Syndrome Bits	0	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1			
7	6	5	4	N	CB0	CB1	D	CB2	D	D	18	CB3	D	D	0	D	1	2	D	
0 0 0 0	1	0	0	1	1	0	0	1	1	0	0	1	1	0	0	1	1	0	1	1
0 0 0 1	2	0	0	0	0	1	1	1	1	0	0	0	0	1	1	1	1	1	1	1
0 0 1 0	3	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	1
0 0 1 1	CB4	D	D	5	D	6	7	D	D	3	16	D	4	D	D	17				
0 1 0 0	CB5	D	D	11	D	19	12	D	D	8	9	D	10	D	D	67				
0 1 0 1	D	13	14	D	15	D	D	21	20	D	D	66	D	22	23	D				
0 1 1 0	CB6	D	D	25	D	26	49	D	D	48	24	D	27	D	D	50				
0 1 1 1	D	52	55	D	51	D	D	70	28	D	D	65	D	53	54	D				
1 0 0 0	D	29	31	D	64	D	D	69	68	D	D	32	D	33	34	D				
1 0 0 1	30	D	D	37	D	38	39	D	D	35	71	D	36	D	D	U				
1 0 1 0	CB7	D	D	43	D	77	44	D	D	40	41	D	42	D	D	U				
1 0 1 1	D	45	46	D	47	D	D	74	72	D	D	U	D	73	U	D				
1 1 0 0	D	59	75	D	79	D	D	58	60	D	D	56	D	U	57	D				
1 1 0 1	63	D	D	62	D	U	U	D	D	U	U	D	61	D	D	U				
1 1 1 0	D	U	U	D	U	D	D	U	76	D	D	U	D	U	D	U				
1 1 1 1	78	D	D	U	D	U	U	D	D	U	U	D	U	D	D	U				
1 1 1 1	U	D	D	U	D	U	U	D	D	U	U	D	U	D	D	U				
1 1 1 1	D	U	U	D	U	D	D	U	D	D	U	D	U	D	U	D				

N = No Error  
 CBX = Error in Check Bit X  
 X = Error in Data Bit X  
 D = Double Bit Error  
 U = Uncorrectable Multi-Bit Error

SYSTEM ENVIRONMENT

The 8206 interface to a typical 32 bit memory system is illustrated in Figure 4. For larger systems, the partial parity bits from slaves two to four must be

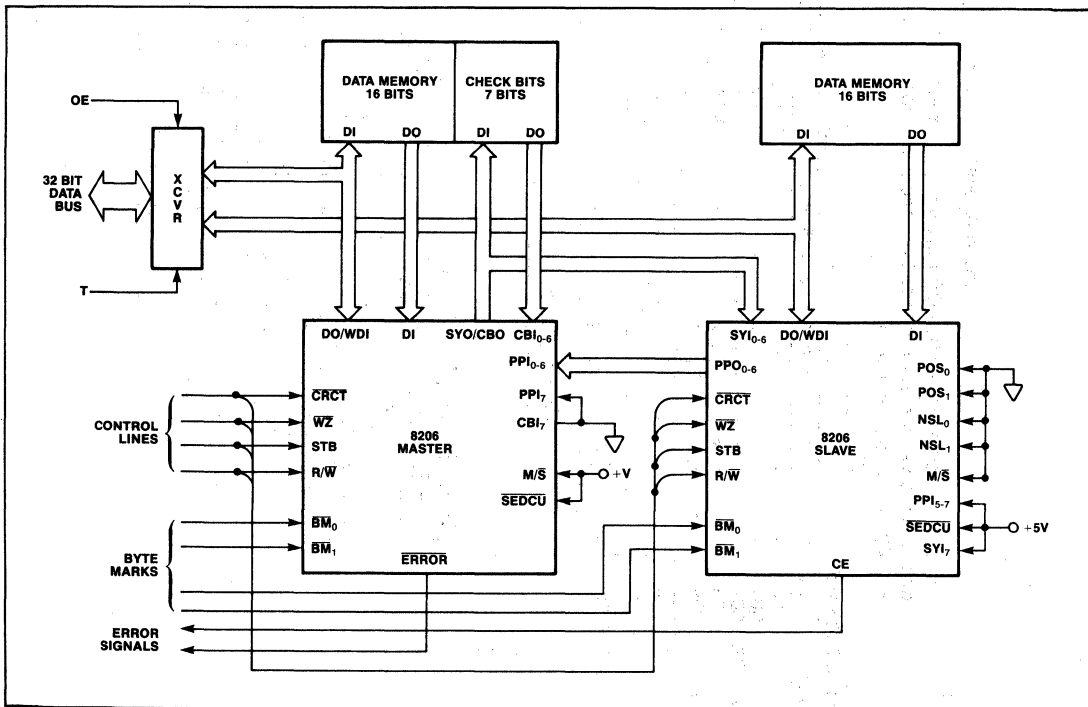


Figure 4. 32-Bit 8206 System Interface

XOR'ed externally, which calls for one level of XOR gating for three 8206's and two levels for four or five 8206's.

The 8206 is designed for direct connection to the Intel 8207 Advanced Dynamic RAM Controller, due to be sampled in the first quarter of 1982. The 8207 has the ability to perform dual port memory control,

and Figure 5 illustrates a highly integrated dual port RAM implementation using the 8206 and 8207. The 8206/8207 combination permits such features as automatic scrubbing (correcting errors in memory during refresh), extending RAS and CAS timings for Read-Modify-Writes in single memory cycles, and automatic memory initialization upon reset. Together these two chips provide a complete dual-port, error-corrected dynamic RAM subsystem.

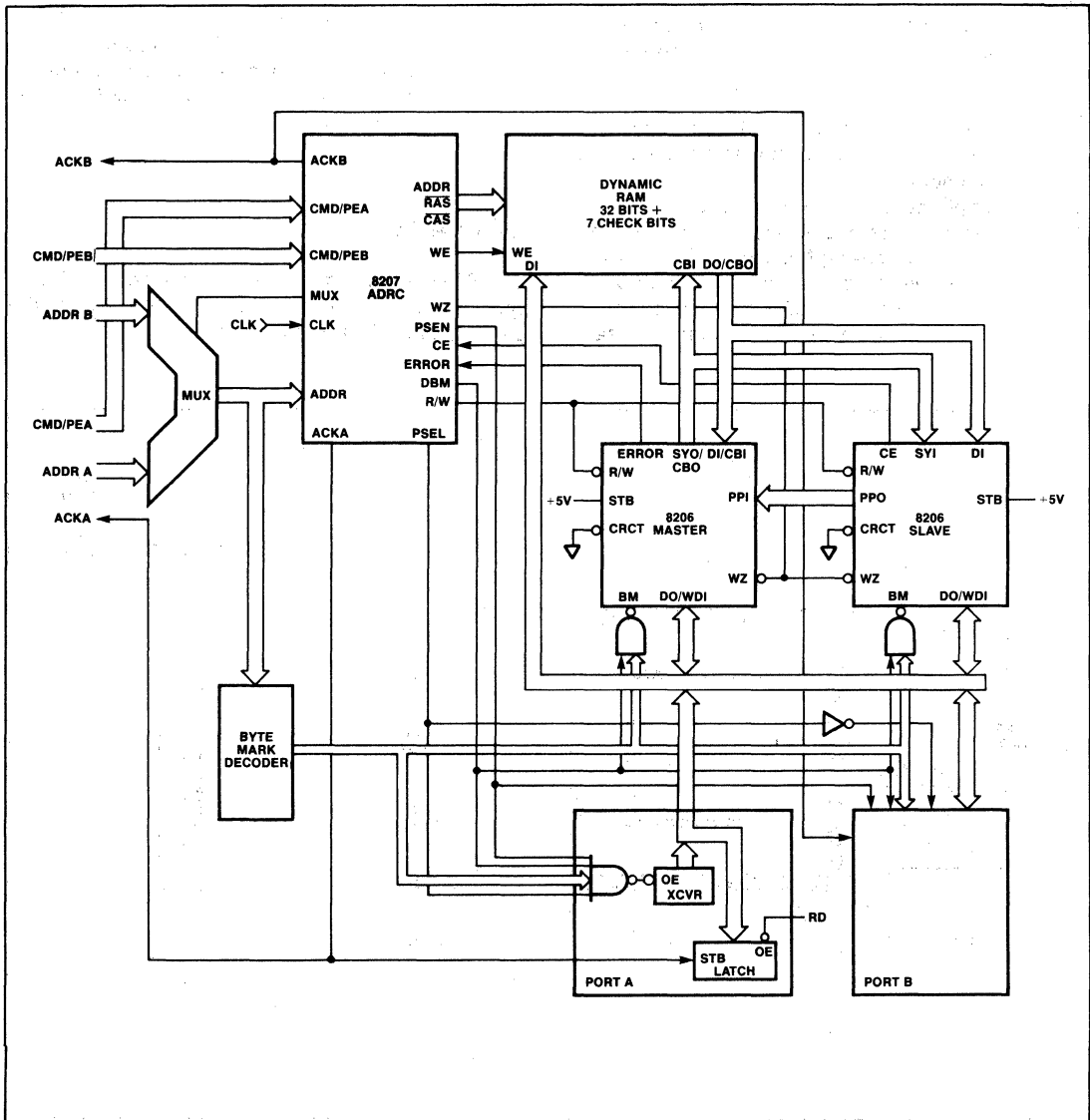


Figure 5. Dual Port RAM Subsystem with 8206/8207 (32-bit bus)

### MEMORY BOARD TESTING

The 8206 lends itself to straightforward memory board testing with a minimum of hardware overhead. The following is a description of four common test modes and their implementation.

**Mode 0—Read and write with error correction.**  
 Implementation: This mode is the normal 8206 operating mode.

**Mode 1—Read and write data with error correction disabled to allow test of data memory.**  
 Implementation: This mode is performed with  $\overline{\text{CRCT}}$  deactivated.

**Mode 2—Read and write check bits with error correction disabled to allow test of check bits memory.**  
 Implementation: Any pattern may be written into the check bits memory by judiciously choosing the proper data word to generate the desired check bits, through the use of the 8206 Hamming code. To read out the check bits it is first necessary

to fill the data memory with all zeros, which may be done by activating  $\overline{\text{WZ}}$  and incrementing memory addresses with  $\overline{\text{WE}}$  to the check bits memory held inactive, and then performing ordinary reads. The check bits will then appear directly at the SYO outputs, with bits CB0 and CB1 inverted.

**Mode 3—Write data, without altering or writing check bits, to allow the storage of bit combinations to cause error correction and detection.**  
 Implementation: This mode is implemented by writing the desired word to memory with  $\overline{\text{WE}}$  to the check bits array held inactive.

### PACKAGE

The 8206 is packaged in a 68-pin, leadless JEDEC type A hermetic chip carrier. Figure 6 illustrates the package, and Figure 7 is the pinout.

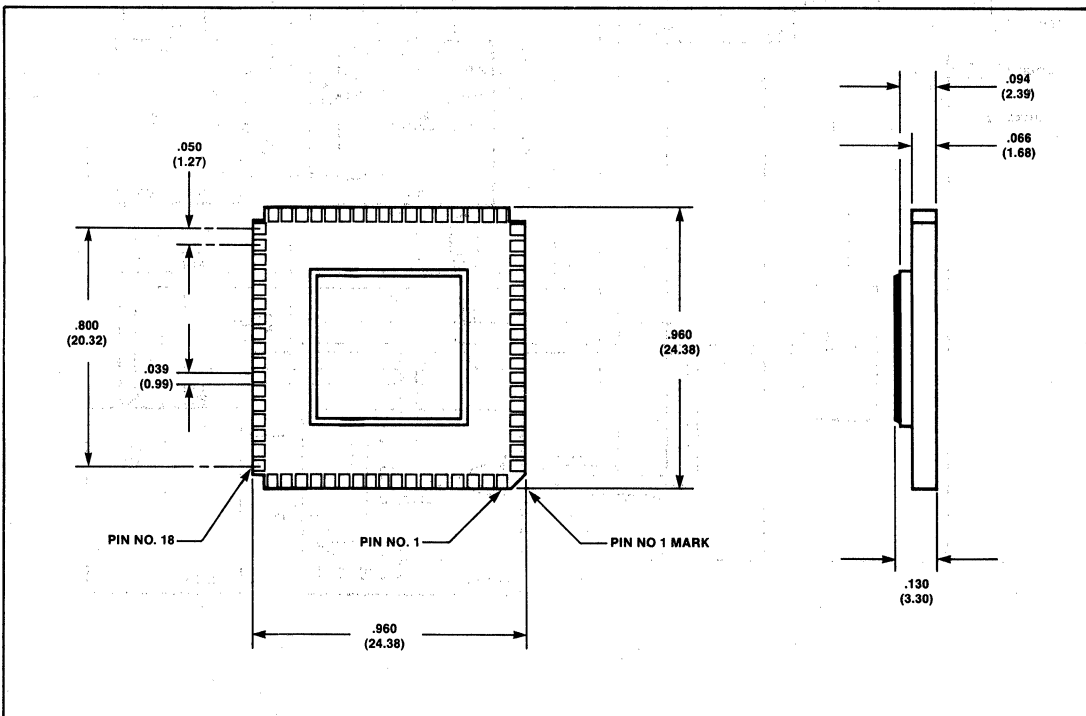


Figure 6. 8206 JEDEC Type A Package

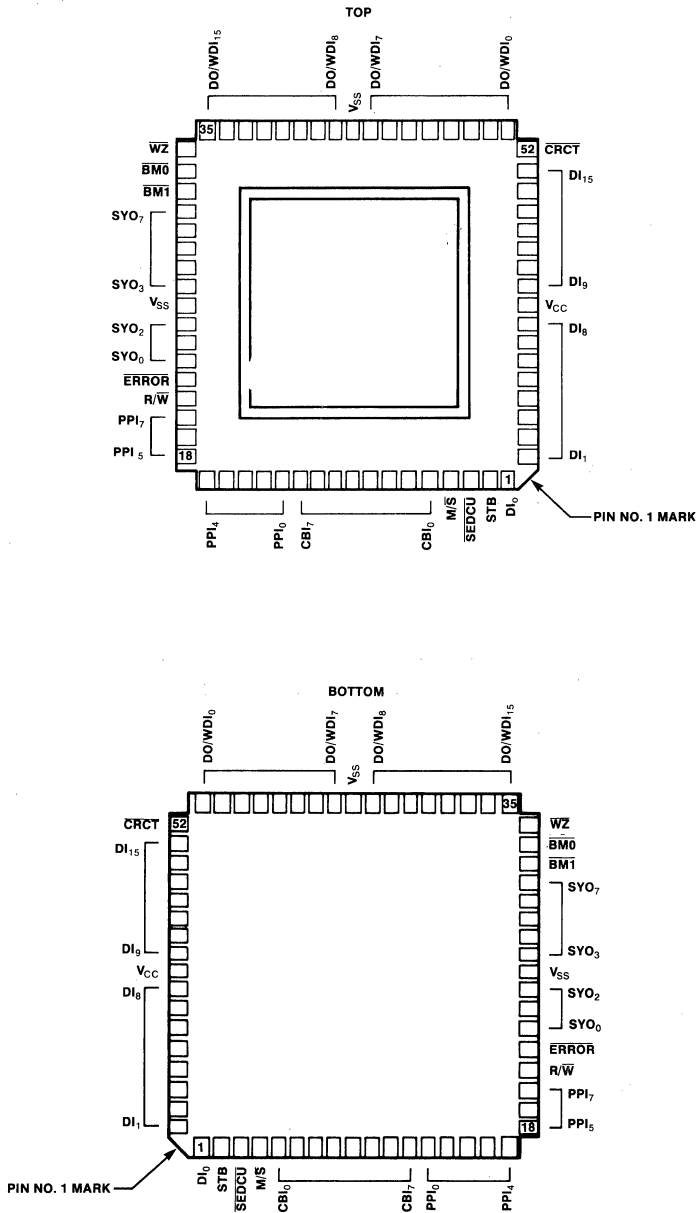


Figure 7. 8206 Pinout Diagram

**ABSOLUTE MAXIMUM RATINGS\***

Ambient Temperature Under Bias ..... 0°C to 70°C  
 Storage Temperature ..... -65°C to +150°C  
 Voltage On Any Pin  
     With Respect to Ground ..... -0.5V to +7V  
 Power Dissipation ..... 2.5 Watts

*\*NOTE: Stresses above those listed under "Absolute Maximum Ratings" may cause permanent damage to the device. This is a stress rating only and functional operation of the device at these or any other conditions above those indicated in the operational sections of this specification is not implied. Exposure to absolute maximum rating conditions for extended periods may affect device reliability.*

**D.C. CHARACTERISTICS** ( $T_A = 0^\circ\text{C}$  to  $70^\circ\text{C}$ ,  $V_{CC} = 5.0\text{V} \pm 10\%$ ,  $V_{SS} = \text{GND}$ )

Symbol	Parameter	Min.	Max.	Unit	Test Conditions
$I_{CC}$	Power Supply Current —Single 8206 or Slave #1 —Master in Multi-Chip or Slaves #2, 3, 4		230 190	mA mA	
$V_{IL}^1$	Input Low Voltage	-0.5	0.8	V	
$V_{IH}^1$	Input High Voltage	2.0	$V_{CC} + 0.5\text{V}$	V	
$V_{OL}$	Output Low Voltage —DO —All Others		0.4 0.4	V V	$I_{OL} = 8\text{mA}$ $I_{OL} = 2.0\text{mA}$
$V_{OH}$	Output High Voltage —DO, CBO —All Other Outputs	2.6 2.4		V V	$I_{OH} = -2\text{mA}$ $I_{OH} = -0.4\text{mA}$
$I_{OL}$	Output Leakage Current		$\pm 10$	$\mu\text{A}$	$0.45\text{V} \leq V_{OUT} \leq V_{CC}$
$I_{LI}$	Input Leakage Current —WDI, PPI, CB16-7, SEDCU —All Other Inputs		20 10	$\mu\text{A}$ $\mu\text{A}$	$0\text{V} \leq V_{IN} \leq V_{CC}$

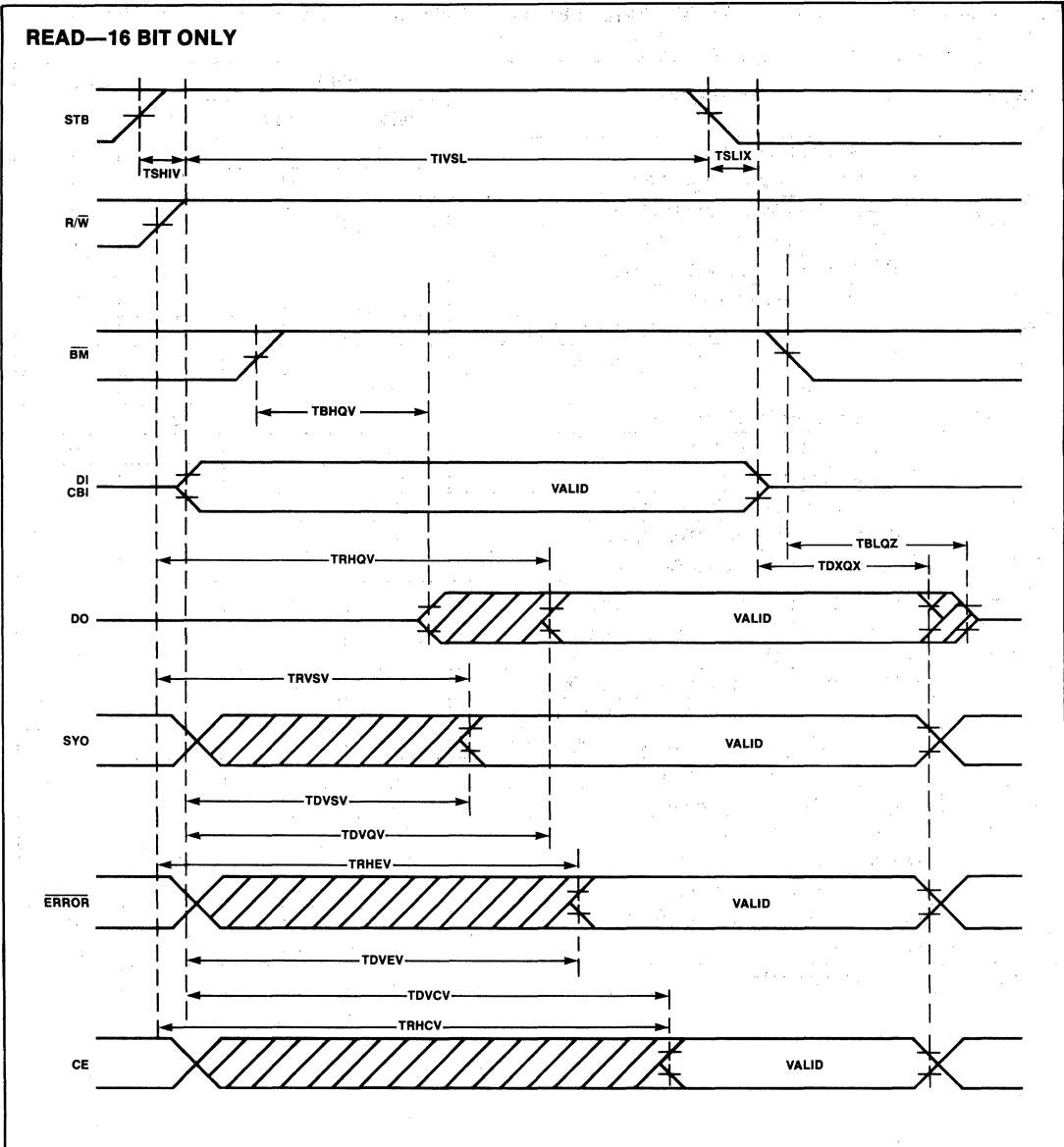
**NOTES:**

1. SEDCU (pin 3) and  $M\bar{S}$  (pin 4) are device strapping options and should be tied to  $V_{CC}$  or GND.  $V_{IH} \text{ min} = V_{CC} - 0.5\text{V}$  and  $V_{IL} \text{ max} = 0.5\text{V}$ .

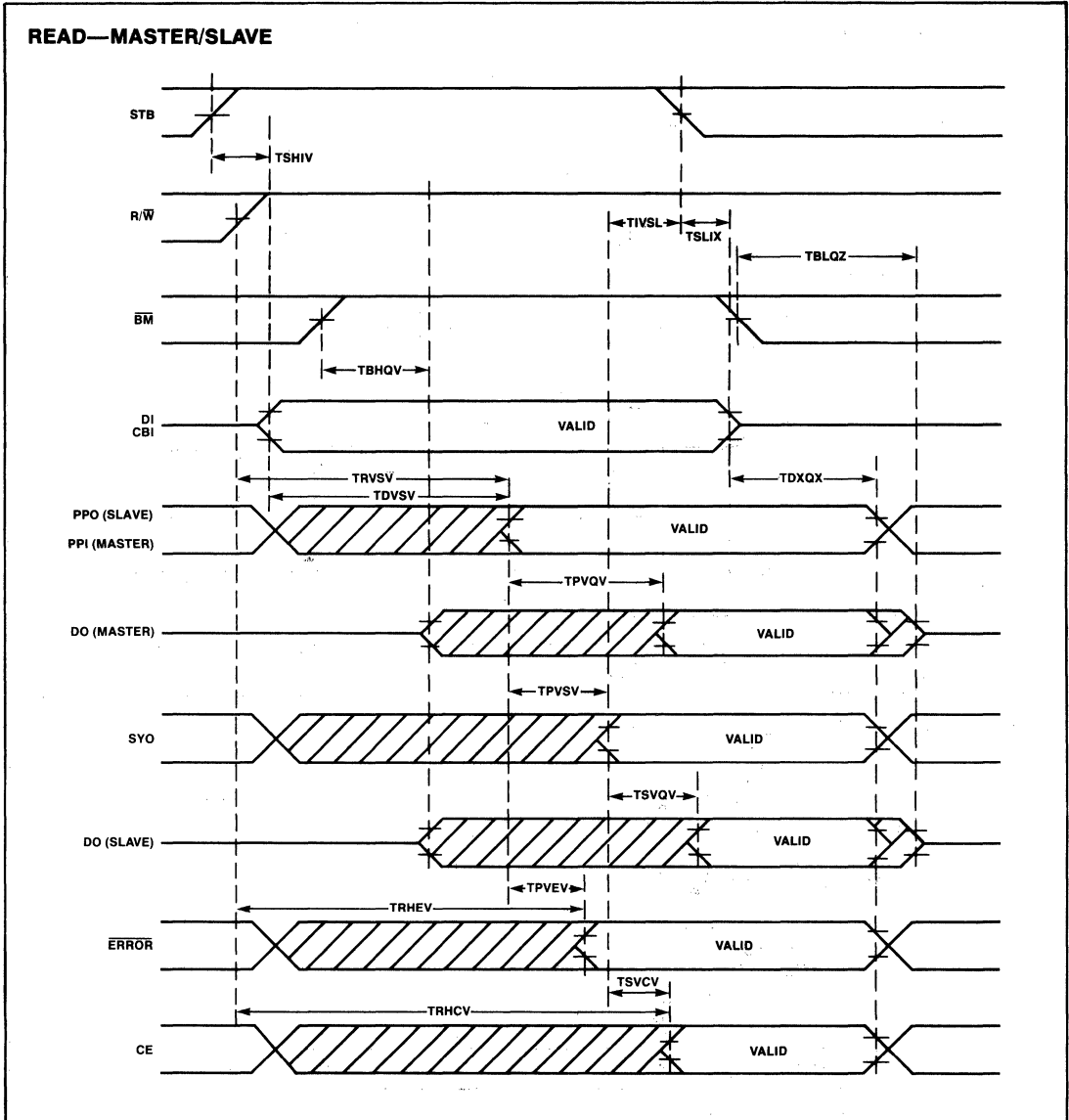
**A.C. CHARACTERISTICS** ( $T_A = 0^\circ\text{C}$  to  $70^\circ\text{C}$ ,  $V_{CC} = +5\text{V} \pm 10\%$ ,  $V_{SS} = 0\text{V}$ ,  $C_L = 100\text{pF}$ ; all times are in nsec.)  
 Measurements made with respect to STB,  $R/\overline{W}$ ,  $\overline{BM}_{0-1}$  are at 1.5V. All other pins are measured at 2.4V and 0.8V. All times are in nsec.

Symbol	Parameter	8206		8206-8	
		Min.	Max.	Min.	Max.
TRHEV	$\overline{\text{ERROR}}$ Valid from $R/\overline{W}\uparrow$		25		34
TRHCV	CE Valid from $R/\overline{W}\uparrow$ (Single 8206)		44		59
TRHQV	Corrected Data Valid from $R/\overline{W}\uparrow$		49		66
TRVSV	SYO/CBO/PPO Valid From $R/\overline{W}$		42		56
TDVEV	$\overline{\text{ERROR}}$ Valid from Data/Check Bits In		52		70
TDVCV	CE Valid from Data/Check Bits In		70		94
TDVQV	Corrected Data Valid from Data/Check Bits In		67		90
TDVSV	SYO/PPO Valid from Data/Check Bits In		55		74
TBHQV	Corrected Data Access Time		32		43
TDXQX	Hold Time from Data/Check Bits In	0		0	
TBLQZ	Corrected Data Float Delay	5	28	5	38
TSHIV	STB High to Data Valid	30		40	
TIVSL	Data/Check Bits In to $\text{STB}\downarrow$ Set-up	0		0	
TSLIX	Data/Check Bits In from $\text{STB}\downarrow$ Hold	20		30	
TPVEV	$\overline{\text{ERROR}}$ Valid from Partial Parity In		30		40
TPVQV	Corrected Data (Master) from Partial Parity In		56		76
TPVSV	Syndrome/Check Bits Out from Partial Parity In		38		51
TSVQV	Corrected Data (Slave) Valid from Syndrome		51		69
TSVCV	CE Valid from Syndrome (Slave number 1)		48		65
TQVQV	Check Bits/Partial Parity Out from Write Data In		59		80
TRHSX	Check Bits/Partial Parity Out from $R/\overline{W}$ , $\overline{WZ}$ Hold	0		0	
TRLSX	Syndrome Out from $R/\overline{W}$ Hold	0		0	
TQXQX	Hold Time from Write Data In	0		0	
TQVRL	Syndrome Out to $R/\overline{W}\downarrow$ Set-up	17		22	
TDVRL	Data/Check Bits In to $R/\overline{W}$ Set-up	34		46	
TDVQU	Uncorrected Data Out from Data In		32		43
TTVQV	Corrected Data Out from $\overline{\text{CORRECT}}\downarrow$		30		40
TWLQL	$\overline{WZ}\downarrow$ to Zero Out		30		40
TWHQX	Zero Out from $\overline{WZ}\uparrow$ Hold	0		0	

WAVEFORMS

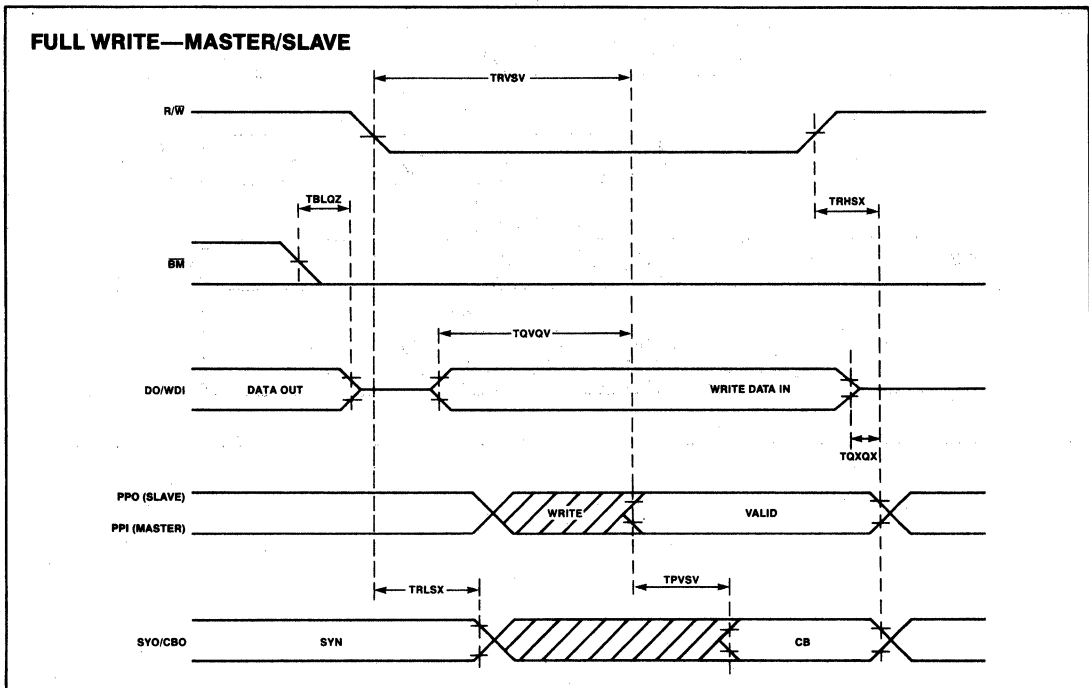
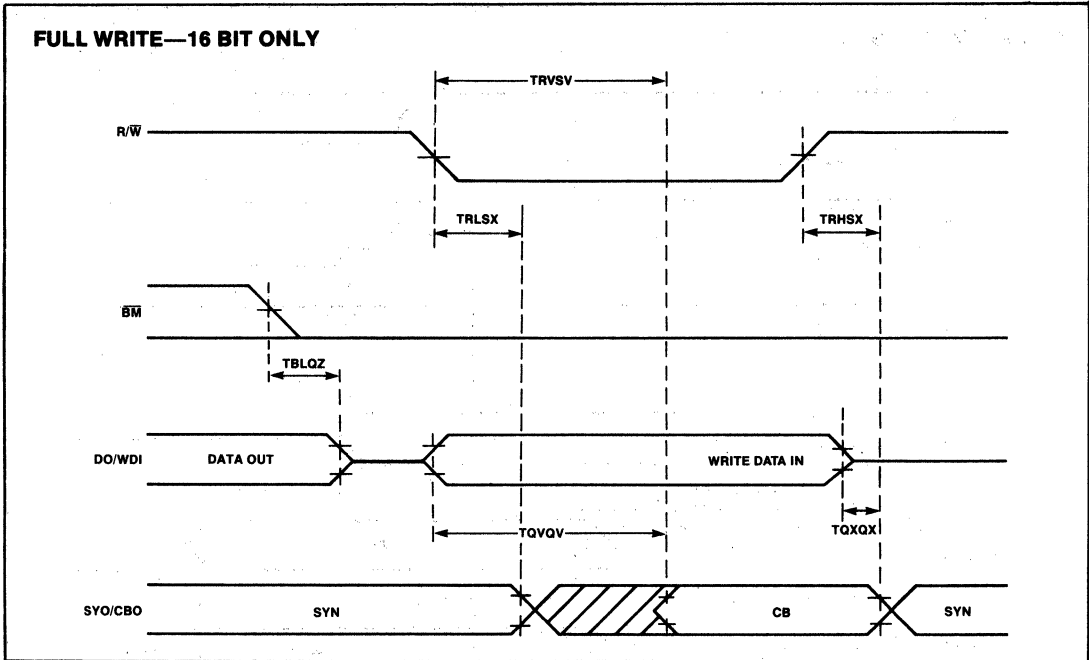


WAVEFORMS (Continued)

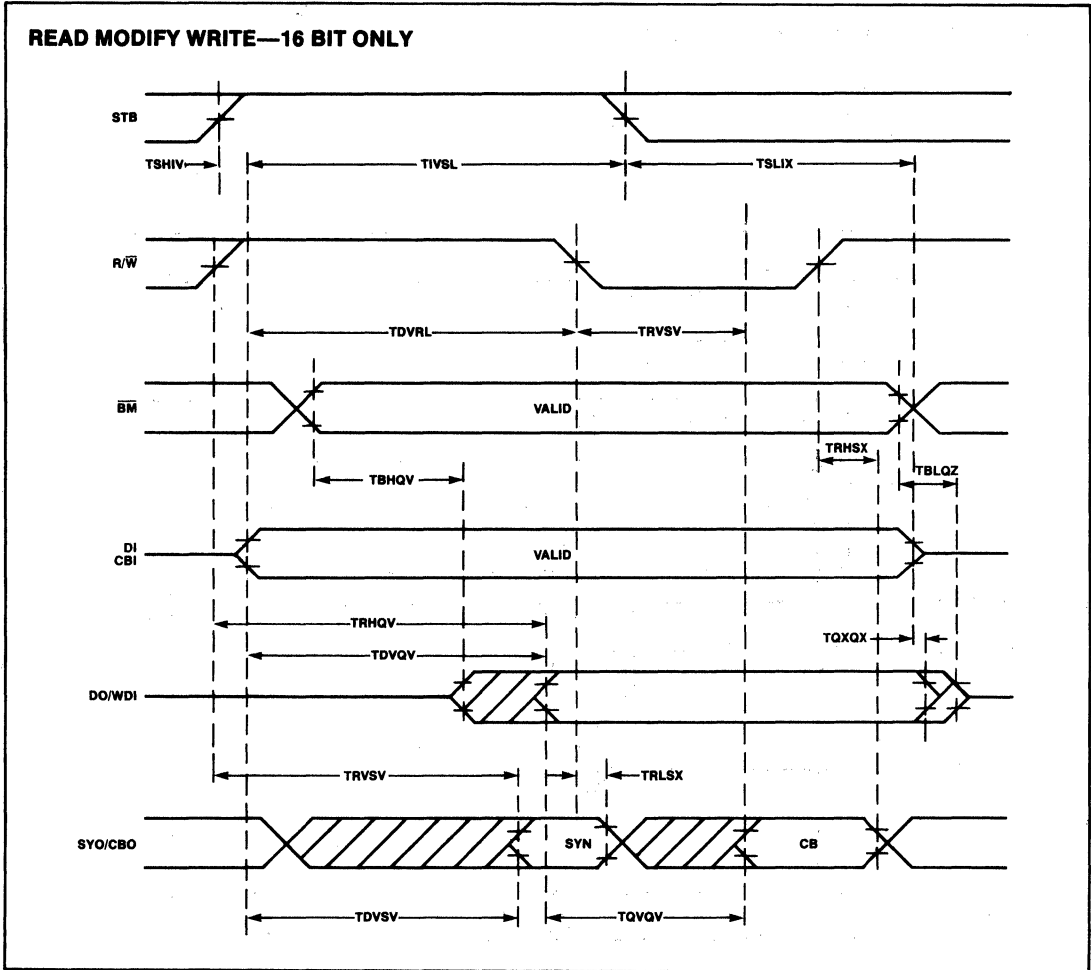




WAVEFORMS (Continued)

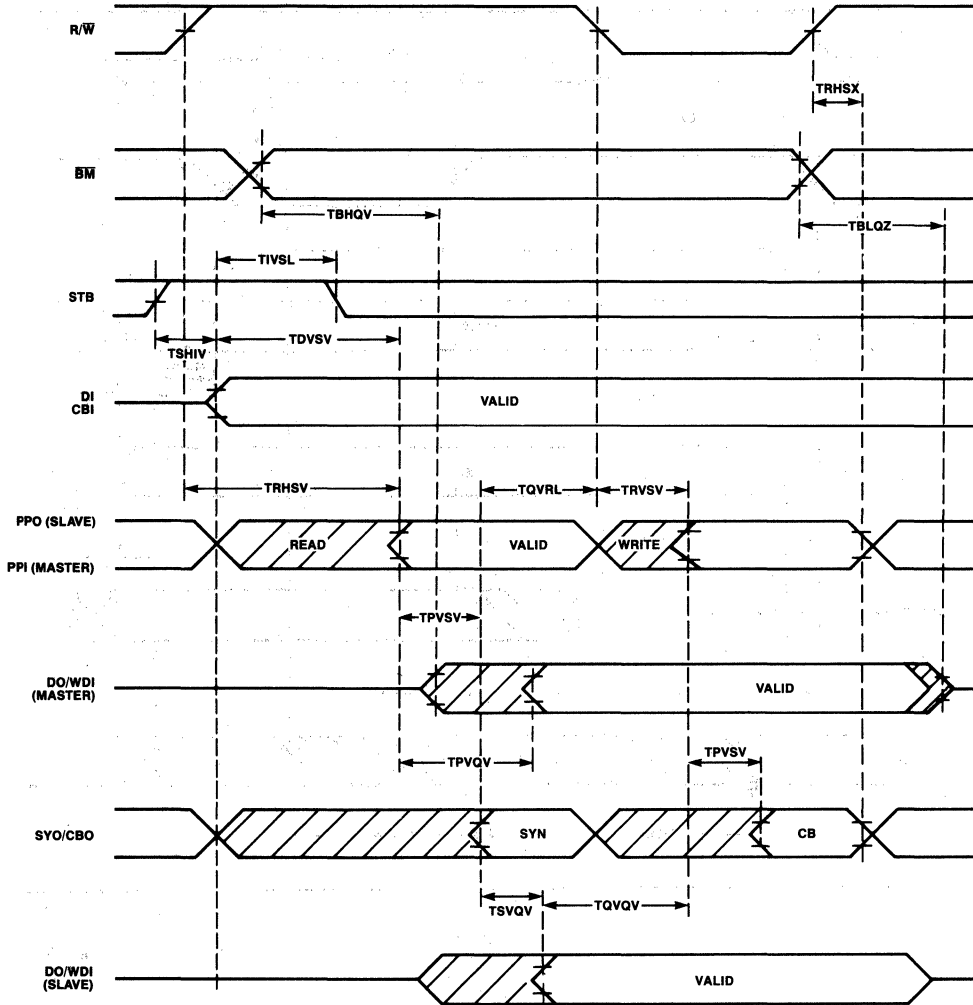


WAVEFORMS (Continued)

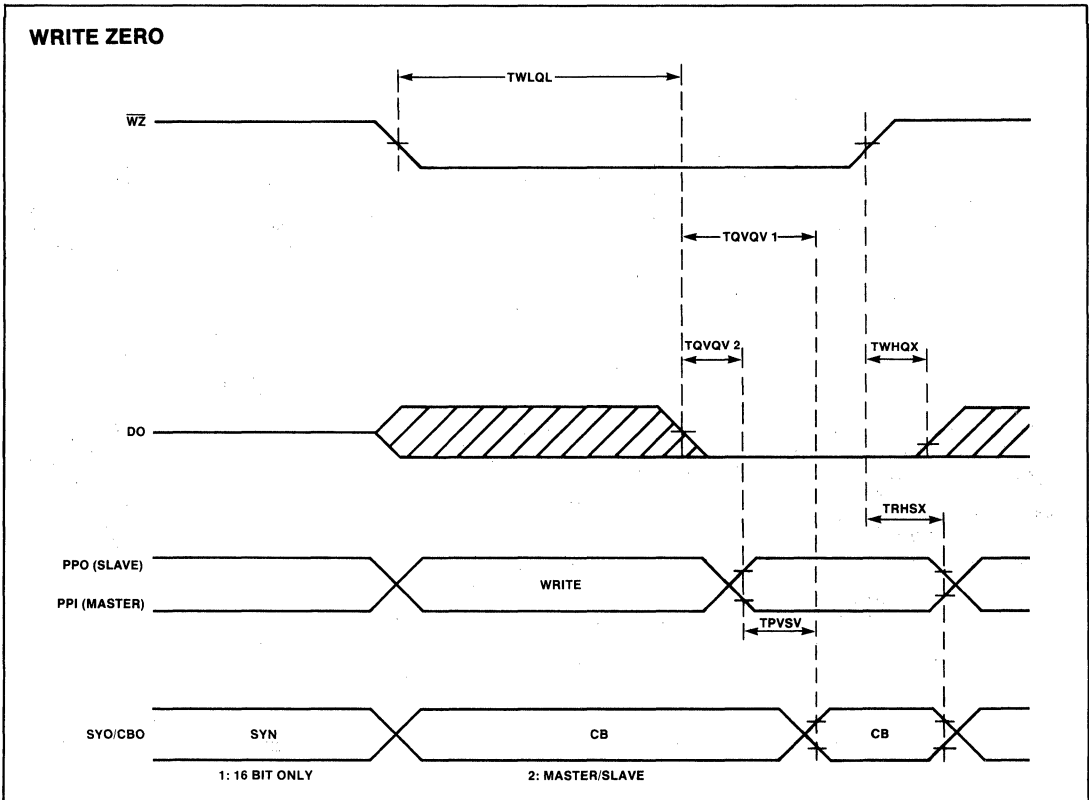
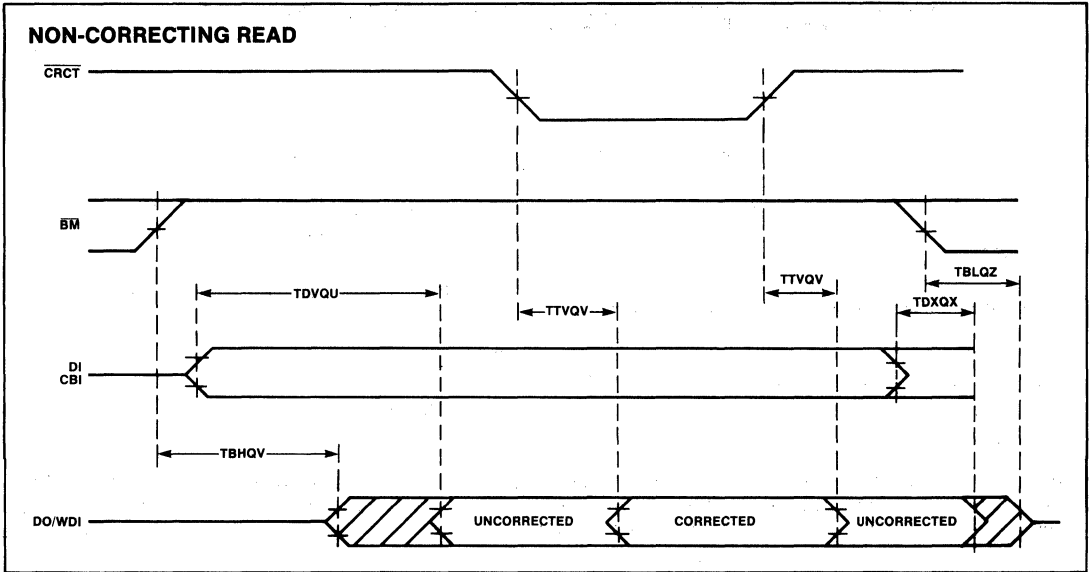


WAVEFORMS (Continued)

READ MODIFY WRITE—MASTER/SLAVE



WAVEFORMS (Continued)





# 8271/8271-6 PROGRAMMABLE FLOPPY DISK CONTROLLER

- IBM 3740 Soft Sector Format Compatible
- Programmable Record Lengths
- Multi-Sector Capability
- Maintain Dual Drives with Minimum Software Overhead Expandable to 4 Drives
- Automatic Read/Write Head Positioning and Verification
- Internal CRC Generation and Checking
- Programmable Step Rate, Settle-Time, Head Load Time, Head Unload Index Count
- Fully MCS-80™ and MCS-85™ Compatible
- Single +5V Supply
- 40-Pin Package

The Intel® 8271 Programmable Floppy Disk Controller (FDC) is an LSI component designed to interface one to 4 floppy disk drives to an 8-bit microcomputer system. Its powerful control functions minimize both hardware and software overhead normally associated with floppy disk controllers.

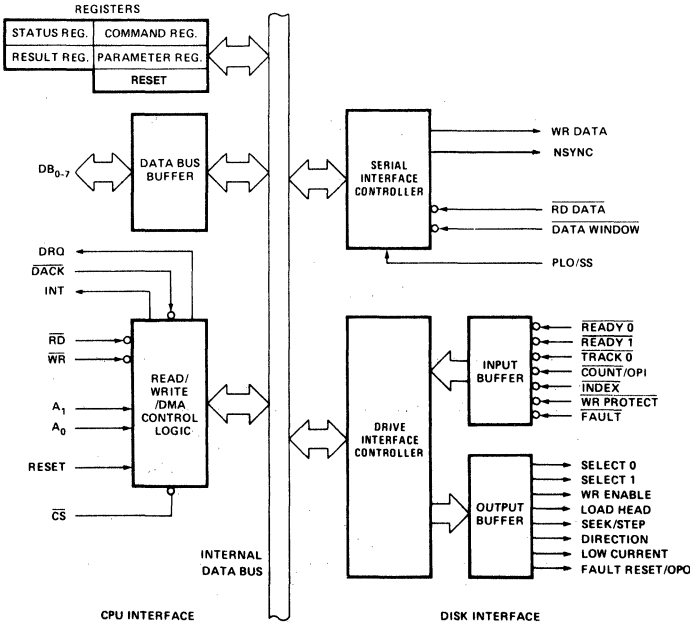


Figure 1. Block Diagram

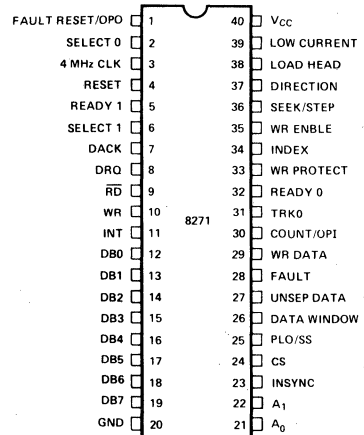


Figure 2. Pin Configuration

**Table 1. Pin Description**

Symbol	Pin No.	Type	Name and Function
V <sub>CC</sub>	40		<b>+5V Supply.</b>
GND	20		<b>Ground.</b>
Clock	3	I	<b>Clock:</b> A square wave clock.
Reset	4	I	<b>Reset:</b> A high signal on the reset input forces the 8271 to an idle state. The 8271 remains idle until a command is issued by the CPU. The output signals of the drive interface are forced inactive (LOW). Reset must be active for 10 or more clock cycles.
$\overline{CS}$	24	I	<b>Chip Select:</b> The I/O Read and I/O Write inputs are enabled by the chip select signal.
DB <sub>7</sub> -DB <sub>0</sub>	19-12	I/O	<b>Data Bus:</b> The Data Bus lines are bidirectional, three-state lines (8080 data bus compatible).
$\overline{WR}$	10	I	<b>Write:</b> The Write signal is used to signal the control logic that a transfer of data from the data bus to the 8271 is required.
$\overline{RD}$	9	I	<b>Read:</b> The Read signal is used to signal the control logic that a transfer of data from the 8271 to the data bus is required.
INT	11	O	<b>Interrupt:</b> The interrupt signal indicates that the 8271 requires service.
A <sub>1</sub> -A <sub>0</sub>	22-21	I	<b>Address Line:</b> These two lines are CPU Interface Register select lines.
DRQ	8	O	<b>Data Request:</b> The DMA request signal is used to request a transfer of data between the 8271 and memory.
$\overline{DACK}$	7	I	<b>Data Acknowledge:</b> The DMA acknowledge signal notifies the 8271 that a DMA cycle has been granted. For non-DMA transfers, this signal should be driven in the manner of a "Chip Select."
Select 1- Select 0	6 2	O	<b>Selected Drive:</b> These lines are used to specify the selected drive. These lines are set by the command byte.

Symbol	Pin No.	Type	Name and Function
Fault Reset/ OPO	1	O	<b>Fault Reset:</b> The optional fault reset output line is used to reset an error condition which is latched by the drive. If this line is not used for a fault reset it can be used as an optional output line. This line is set with the write special register command.
Write Enable	35	O	<b>Write Enable:</b> This signal enables the drive write logic.
Seek/Step	36	O	<b>Seek/Step:</b> This multi-function line is used during drive seeks.
Direction	37	O	<b>Direction:</b> The direction line specifies the seek direction. A high level on this pin steps the R/W head toward the spindle (step-in), a low level steps the head away from the spindle (step-out).
Load Head	38	O	<b>Load Head:</b> The load head line causes the drive to load the Read/Write head against the diskette.
Low Current	39	O	<b>Low Current:</b> This line notifies the drive that track 43 or greater is selected.
Ready 1, Ready 0	5 32	I	<b>Ready 1:</b> These two lines indicate that the specified drive is ready.
$\overline{Fault}$	28	I	<b>Fault:</b> This line is used by the drive to specify a file unsafe condition.
$\overline{Count/OPI}$	30	I	<b>Count/OPI:</b> If the optional seek/direction/count seek mode is selected, the count pin receives pulses to step the R/W head to the desired track. Otherwise, this line can be used as an optional input.
$\overline{Write Protect}$	33	I	<b>Write Protect:</b> This signal specifies that the diskette inserted is write protected.
$\overline{TRK0}$	31	I	<b>Track Zero:</b> This signal indicates when the R/W head is positioned over track zero.
$\overline{Index}$	34	I	<b>Index:</b> The index signal gives an indication of the relative position of the diskette.

Table 1. Pin Description (Continued)

Symbol	Pin No.	Type	Name and Function
PLO/SS	25	I	<b>Phase-Locked Oscillator/Single Shot:</b> This pin is used to specify the type of data separator used.
Write Data	29	O	<b>Write Data:</b> Composite write data.
Unseparated Data	27	I	<b>Unseparated Data:</b> This input is the unseparated data and clocks.
Data Window	26	I	<b>Data Window:</b> This is a data window established by a single-shot or phase-locked oscillator data separator.
INSYNC	23	O	<b>Input Synchronization:</b> This line is high when 8271 has attained input data synchronization, by detecting 2 bytes of zeros followed by an expected Address Mark. It will stay high until the end of the ID or data field.

**FUNCTIONAL DESCRIPTION**

**General**

The 8271 Floppy Disk Controller (FDC) interfaces either two single or one dual floppy drive to an eight bit microprocessor and is fully compatible with Intel's new high performance MCS-85 microcomputer system. With minimum external circuitry, this innovative controller supports most standard, commonly-available flexible disk drives including the mini-floppy.

The 8271 FDC supports a comprehensive soft sectored format which is IBM 3740 compatible and includes provision for the designating and handling of bad tracks. It is a high level controller that relieves the CPU (and user) of many of the control tasks associated with implementing a floppy disk interface. The FDC supports a variety of high level instructions which allow the user to store and retrieve data on a floppy disk without dealing with the low level details of disk operation.

In addition to the standard read/write commands, a scan command is supported. The scan command allows the user program to specify a data pattern and instructs the FDC to search for that pattern on a track. Any application that is required to search the disk for information (such as point of sale price lookup, disk directory search, etc.), may use the scan command to reduce the CPU overhead. Once the scan operation is initiated, no CPU intervention is required.

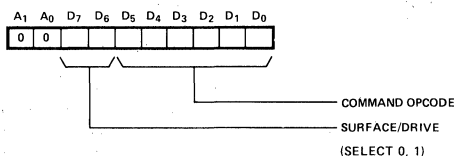
**CPU Interface Description**

This interface minimizes CPU involvement by supporting a set of high level commands and both DMA and non-DMA type data transfers and by providing hierarchical status information regarding the result of command execution.

The CPU utilizes the control interface (see the Block diagram) to specify the FDC commands and to determine the result of an executed command. This interface is supported by five Registers which are addressed by the CPU via the A<sub>1</sub>, A<sub>0</sub>, RD and WR signals. If an 8080 based system is used, the RD and WR signals can be driven by the 8228's I/OR and I/OW signals. The registers are defined as follows:

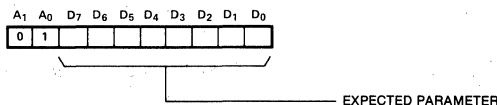
**Command Register**

The CPU loads an appropriate command into the Command Register which has the following format:



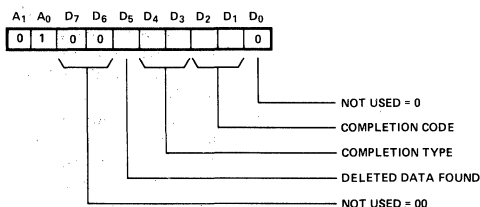
**Parameter Register**

Accepts parameters of commands that require further description; up to five parameters may be required, example:



**Result Register**

The Result Register is used to supply the outcome of FDC command execution (such as a good/bad completion) to the CPU. The standard Result byte format is:



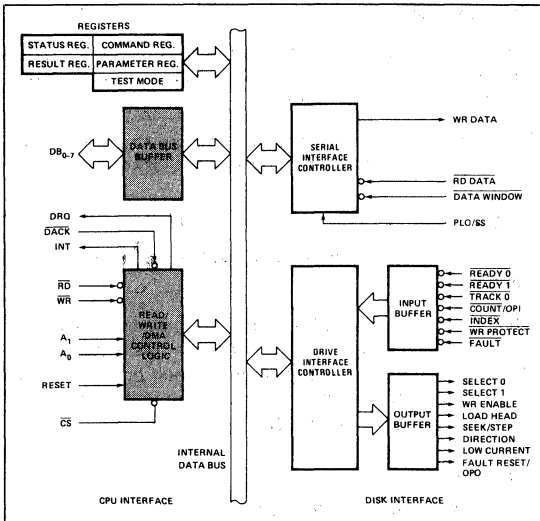
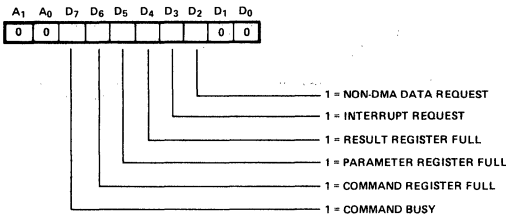


Figure 3. 8271 Block Diagram Showing CPU Interface Functions

**Status Register**

Reflects the state of the FDC.



**Reset Register**

Allows the 8271 to be reset by the program. Reset must be active for 11 or more chip clocks.

**INT (Interrupt Line)**

Another element of the control interface is the Interrupt line (INT). This line is used to signal the CPU that an FDC operation has been completed. It remains active until the result register is read.

**DMA Operation**

The 8271 can transfer data in either DMA or non DMA mode. The data transfer rate of a floppy disk drive is high enough (one byte every 32 usec) to justify DMA transfer. In DMA mode the elements of the DMA interface are:

**DRQ: DMA Request:**

The DMA request signal is used to request a transfer of data between the 8271 and memory.

**DACK: DMA Acknowledge:**

The DMA acknowledge signal notifies the 8271 that a DMA cycle has been granted.

**RD, WR: Read, Write**

The read and write signals are used to specify the direction of the data transfer.

DMA transfers require the use of a DMA controller such as the Intel<sup>®</sup> 8257. The function of the DMA controller is to provide sequential addresses and timing for the transfer at a starting address determined by the CPU. Counting of data block lengths is performed by the FDC.

To request a DMA transfer, the FDC raises DRQ. DACK and RD enable DMA data onto the bus (independently of CHIP SELECT). DACK and WR transfer DMA data to the FDC. If a data transfer request (read or write) is not serviced within 31 μsec, the command is cancelled, a late DMA status is set, and an interrupt is generated. In DMA mode, an interrupt is generated at the completion of the data block transfer.

When configured to transfer data in non-DMA mode, the CPU must pass data to the FDC in response to the non-DMA data requests indicated by the status word. The data is passed to and from the chip by asserting the DACK and the RD or WR signals. Chip select should be inactive (HIGH).

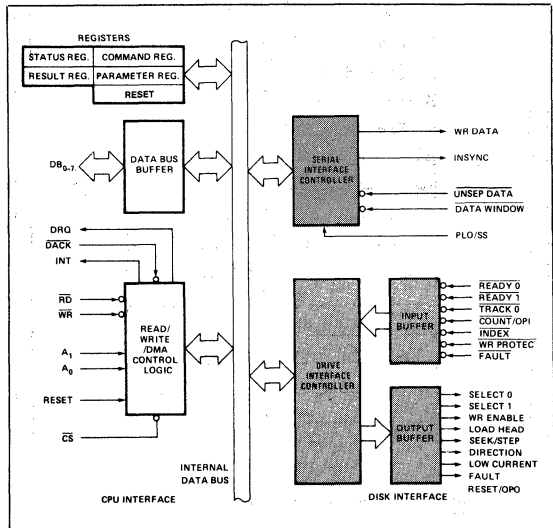


Figure 4. 8271 Block Diagram Showing Disk Interface Functions



**Disk Drive Interface**

The 8271 disk drive interface supports the high level command structure described in the Command Description section. The 8271 maintains the location of bad tracks and the current track location for two drives. However, with minor software support, this interface can support four drives by expanding the two drive select lines (select 0, select 1) with the addition of minimal support hardware.

The FDC Disk Drive Interface has the following major functions.

**READ FUNCTIONS**

Utilize the user supplied data window to obtain the clock and data patterns from the unseparated read data.

Establish byte synchronization.

Compute and verify the ID and data field CRCs.

**WRITE FUNCTIONS**

Encode composite write data.

Compute the ID and data field CRCs and append them to their respective fields.

**CONTROL FUNCTIONS**

Generate the programmed step rate, head load time, head settling time, head unload delay, and monitor drive functions.

**Data Separation**

The 8271 needs only a data window to separate the data from the composite read data as well as to detect missing clocks in the Address Marks.

The window generation logic may be implemented using either a single-shot separator or a phase-locked oscillator.

**Single-Shot Separator**

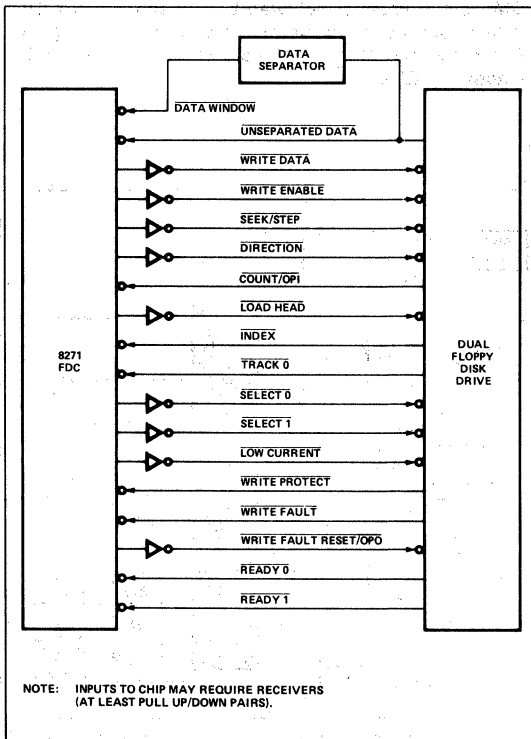
The single-shot separator approach is the lowest cost solution.

The FDC samples the value of Data Window on the leading edge of Unseparated Data and determines whether the delay from the previous pulse was a half or full bit-cell (high input = full bit-cell, low input = half bit-cell).

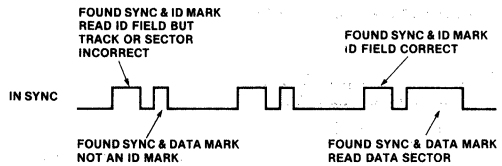
PLO/SS should be tied to Ground.

**Insync Pin**

This pin gives an indication of whether the 8271 is synchronized with the serial data stream during read operations. This pin can be used with a phase-locked oscillator for soft and hard locking.



**Figure 5. 8271 Disk Drive Interface**



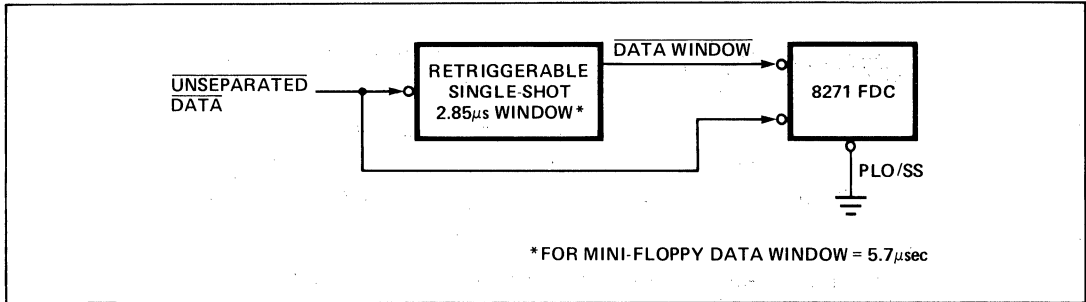


Figure 6. Single-Shot Data Separator Block Diagram

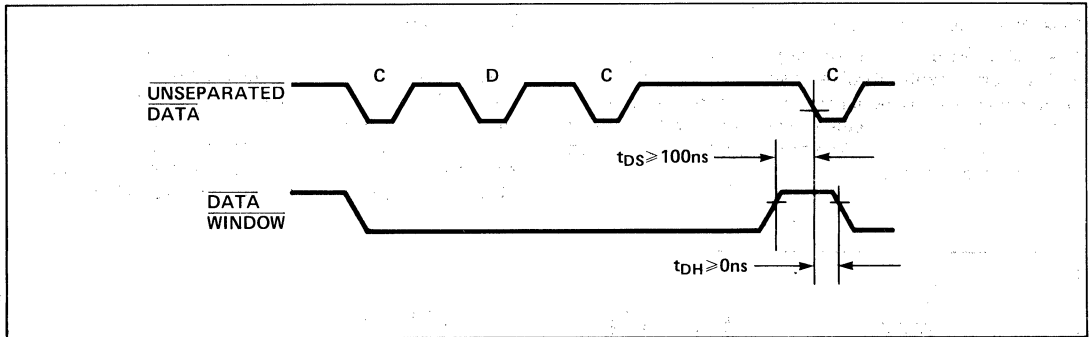


Figure 7. Single-Shot Data Window Timing

**Phase-Locked Oscillator Separator**

The FDC samples the value of Data Window on the leading edge of Unseparated Data and determines whether the pulse represents a Clock or Data Pulse.

Insync may be used to provide soft and hard locking control for the phase-locked oscillator.

PLO/SS should be tied to Vcc (+5V).

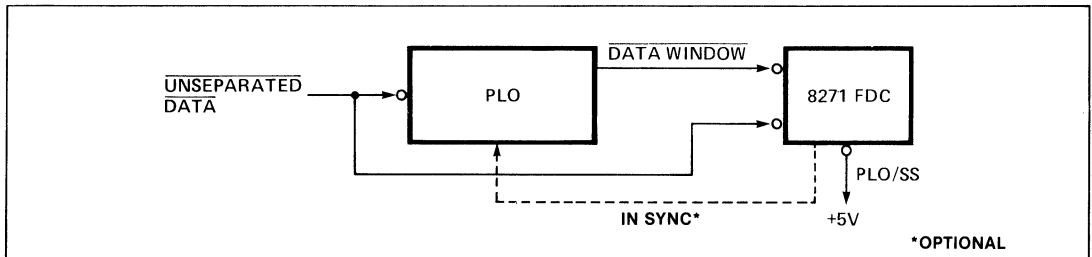


Figure 8. PLO Data Separator Block Diagram

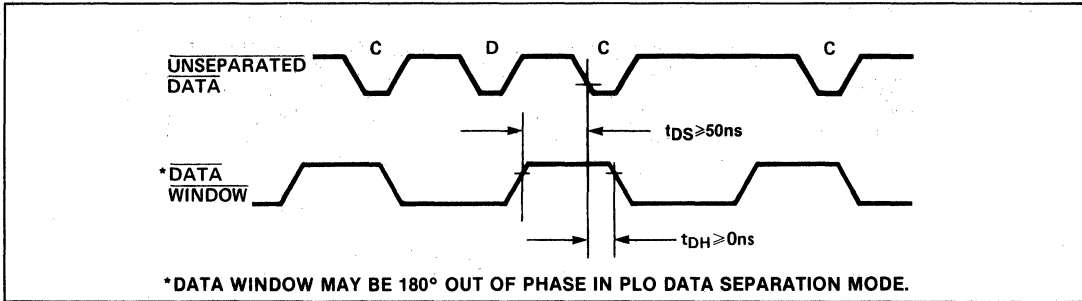


Figure 9. PLO Data Window Timing

**Disk Drive Control Interface**

The disk drive control interface performs the high level and programmable flexible disk drive operations. It custom tailors many varied drive performance parameters such as the step rate, settling time, head load time, and head unload index count. The following is the description of the control interface.

**Write Enable**

The Write Enable controls the read and write functions of a flexible disk drive. When Write Enable is a logical one, it enables the drive write electronics to pass current through the Read/Write head. When Write Enable is a logical zero, the drive Write circuitry is disabled and the Read/Write head detects the magnetic flux transitions recorded on a diskette. The write current turn-on is as follows.

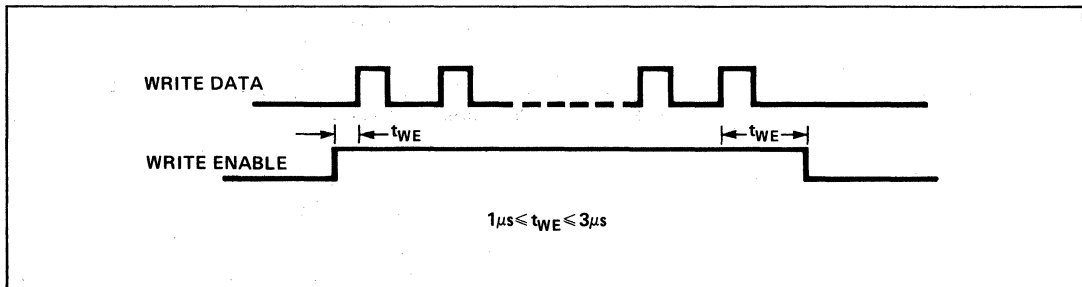


Figure 10. Write Enable Timing

**Seek Control**

Seek Control is accomplished by Seek/Step, Direction, and Count pins and can be implemented two ways to provide maximum flexibility in the subsystem design. One instance is when the programmed step rate is not equal to zero. In this case, the 8271 uses the Seek/Step and Direction pins (the Seek/Step pin becomes a Step pin). Programmable Step timing parameters are shown.

Another instance is when the programmable step rate is equal to zero, in which case the 8271 holds the seek line high until the appropriate number of user-supplied step pulses have been counted on the count input pin.

The Direction pin is a control level indicating the direction in which the R/W head is stepped. A logic high level on this line moves the head toward the spindle (step-in). A logic low level moves the head away from the spindle (step-out).

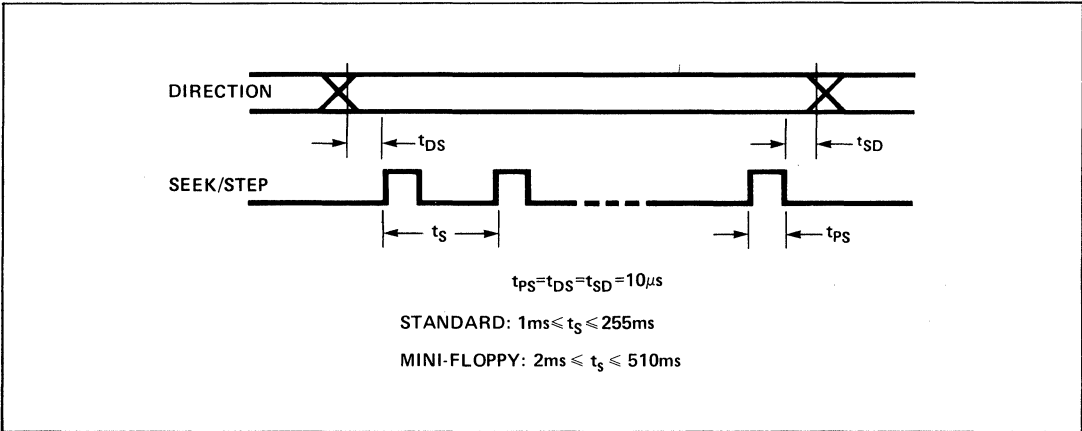


Figure 11. Seek Timing

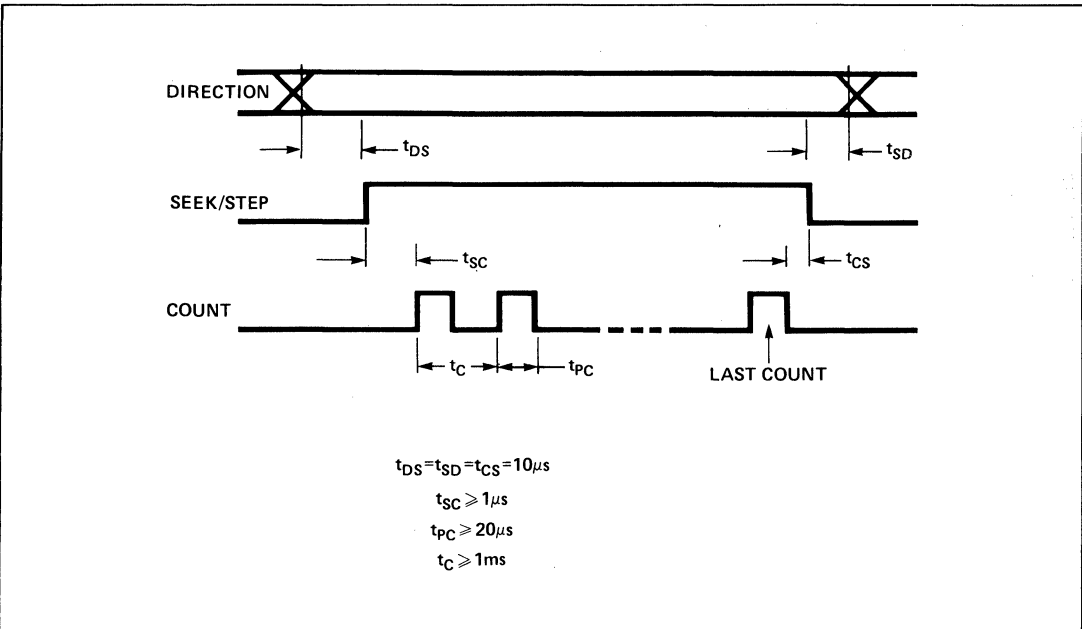
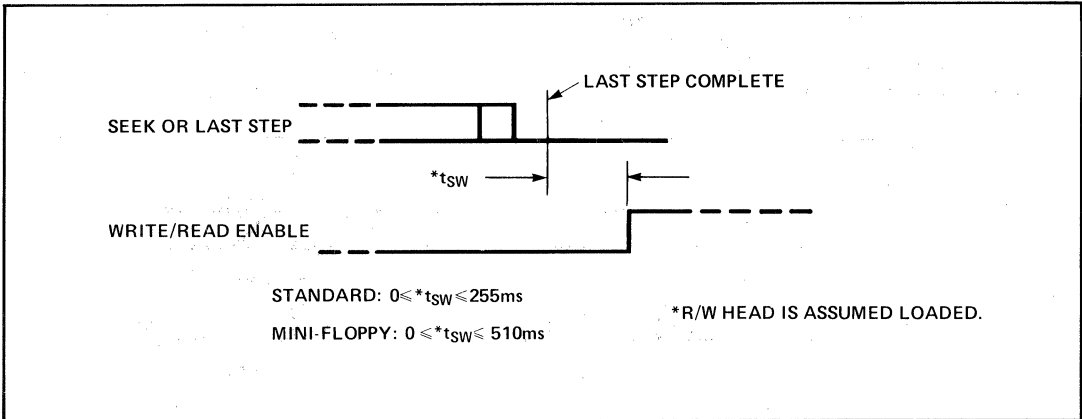


Figure 12. Seek/Step/Count Timing

**Head Seek Settling Time**

The 8271 allows the head settling time to be programmed from 0 to 255ms, in increments of 1ms.

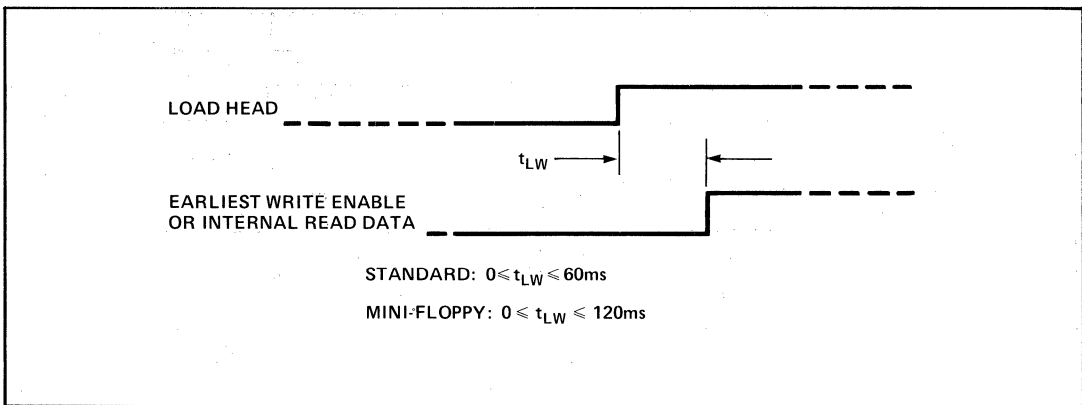
The head settling time is defined as the interval of time from completion of the last step to the time when reading or writing on the diskette is possible (R/W Enable). The R/W head is assumed loaded.



**Figure 13. Head Load Settling Timing**

**Load Head**

When active, load head output pin causes the drive's read/write head to be loaded on the diskette. When the head is initially loaded, there is a programmed delay (0 to 60ms in 4ms increments) prior to any read or write operation. Provision is also made to unload the head following an operation within a programmed number of diskette revolutions.



**Figure 14. Head Load to Read/Write Timing**

## Index

The Index input is used to determine "Sector not found" status and to initiate format track/read ID commands and head unload Index and Count operations.

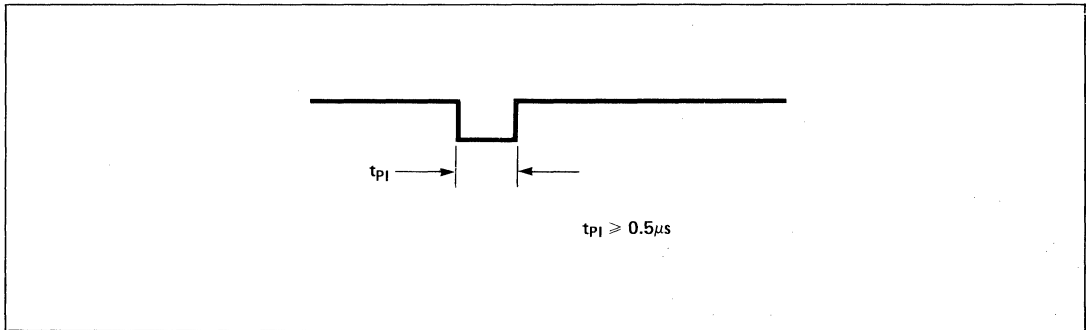


Figure 15. Index Timing

### Track 0

This input pin indicates that the diskette is at track 0. During any seek operation, the stepping out of the actuator ceases when the track 0 pin becomes active.

### Select 1, 0

Only one drive may be selected at a time. The Input/Output pins that must be externally qualified with Select 0 and Select 1 are:

- Unseparated Data
- Data Window
- Write Enable
- Seek/Step
- Count/Optional Input
- Load Head
- Track 0
- Low Current
- Write Protect
- Write Fault
- Fault Reset/Optional Output
- Index

When a new set of select bits is specified by a new command or the FDC finishes the index count before head unload, the following pins will be set to the 0 state:

- Write Enable (35)
- Seek/Step (36)
- Direction (37)
- Load Head (38)
- Low Head Current (39)

The select pins will be set to the state specified by the command or both are set to zero following the index count before head unload.

### Low Current

This output pin is active whenever the physical track location of the selected drive is greater than 43. Generally

this signal is used to enable compensation for the lower velocities encountered while recording on the inner tracks.

### Write Protect

The 8271 will not write to a disk when this input pin is active and will interrupt the CPU if a Write attempt is made. Operations which check Write Protect are aborted if the Write Protect line is active.

This signal normally originates from a sensor which detects the presence or absence of the Write Protect hole in the diskette jacket.

### Write Fault and Write Fault Reset

The Write Fault input is normally latched by the drive and indicates any condition which could endanger data integrity. The 8271 interrupts the CPU anytime Write Fault is detected during an operation and immediately resets the Write Enable, Seek/Step, Direction, and Low Current signals. The write fault condition can be cleared by using the write fault reset pin. If the drive being used does not support write fault, then this pin should be connected to  $V_{CC}$  through a pull-up resistor.

### Ready 1, 0

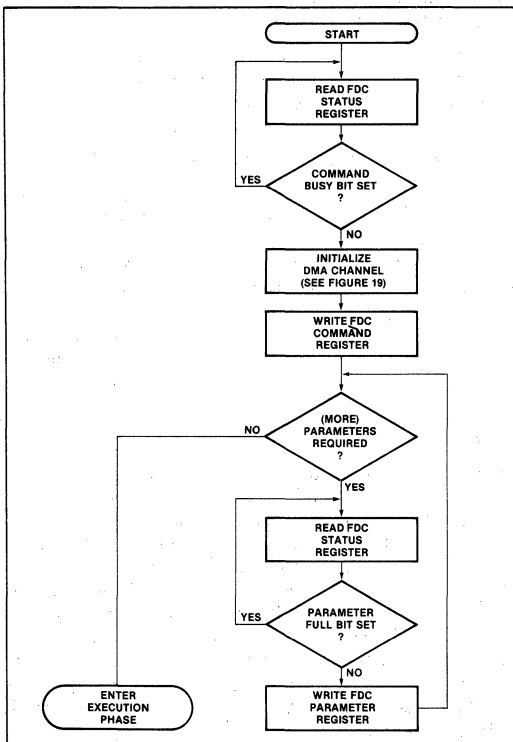
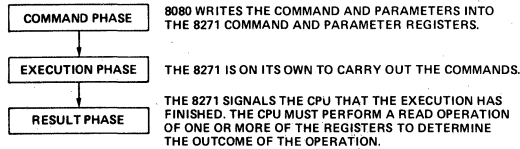
These two pins indicate the functional status of the disk drives. Whenever an operation is attempted on a drive which is not ready, an interrupt is generated. The interface continually monitors this input during an operation and if a Not Ready condition occurs, immediately terminates the operation. Note that the 8271 latches the Not Ready condition and it can only be reset by the execution of a Read Drive Status command. For drives that do not support a ready signal, either one can be derived with a one shot and the index pulse, or the ready inputs can be grounded and Ready determined through some software means.

**PRINCIPLES OF OPERATION**

As an 8080 peripheral device, the 8271 accepts commands from the CPU, executes them and provides a RESULT back to the 8080 CPU at the end of command execution. The communication with the CPU is established by the activation of  $\overline{CS}$  and  $\overline{RD}$  or  $\overline{WR}$ . The  $A_1, A_0$  inputs select the appropriate registers on the chip:

DACK	$\overline{CS}$	$A_1$	$A_0$	$\overline{RD}$	$\overline{WR}$	Operation
1	0	0	0	0	1	Read Status
1	0	0	0	1	0	Write Command
1	0	0	1	0	1	Read Result
1	0	0	1	1	0	Write Parameter
1	0	1	0	1	0	Write Reset Reg.
0	1	X	X	1	0	Write Data
0	1	X	X	0	1	Read Data
0	0	X	X	X	X	Not Allowed

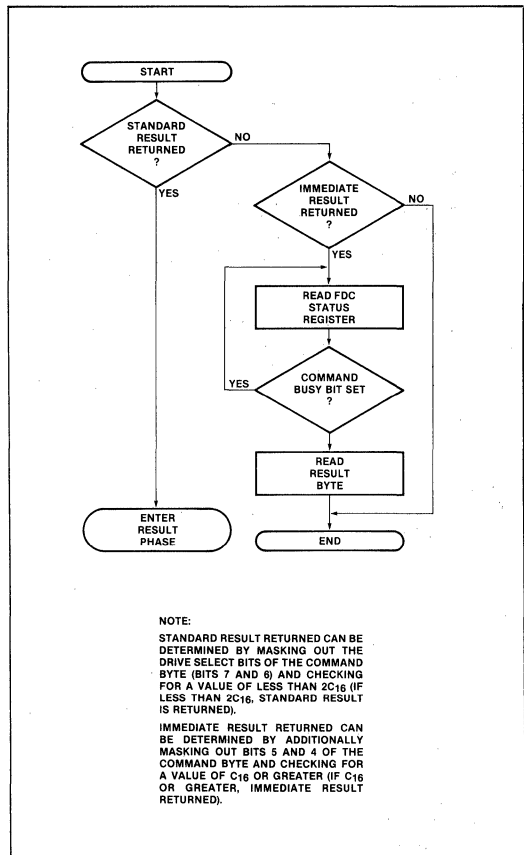
The FDC operation is composed of the following sequence of events.



**Figure 16. Passing the Command and Parameters to the 8271**

**The Command Phase**

The software writes a command to the command register. As a function of the command issued, from zero to five parameters are written to the parameter register. Refer to diagram showing a flow chart of the command phase. Note that the flow chart shows that a command may not be issued if the FDC status register indicates that the device is busy. Issuing a command while another command is in progress is illegal. The flow chart also shows a parameter buffer full check. The FDC status indicates the state of the parameter buffer. If a parameter is issued while the parameter buffer is full, the previous parameter is overwritten and lost.



**Figure 17. Checking for Result Type Following 8271 Command and Parameters**

**The Execution Phase**

During the execution phase the operation specified during the command phase is performed. During this phase, there is no CPU involvement if the system utilizes DMA for the data transfers. The execution phase of each command is discussed within the detailed command descriptions. The following table summarizes many of the basic execution phase characteristics.

**EXECUTION PHASE BASIC CHARACTERISTICS**

The following table summarizes the various commands with corresponding execution phase characteristics.

**Table 2. Execution Phase Basic Characteristics**

COMMANDS	1 Deleted Data	2 Head	3 Ready	4 Write/ Protect	5 Seek	6 Seek Check	7 Result	8 Completion Interrupt
SCAN DATA	SKIP	LOAD	✓	x	YES	YES	YES	YES
SCAN DATA AND DEL DATA	XFER	LOAD	✓	x	YES	YES	YES	YES
WRITE DATA	x	LOAD	✓	✓	YES	YES	YES	YES
WRITE DEL DATA	x	LOAD	✓	✓	YES	YES	YES	YES
READ DATA	SKIP	LOAD	✓	x	YES	YES	YES	YES
READ DATA AND DEL DATA	XFER	LOAD	✓	x	YES	YES	YES	YES
READ ID	x	LOAD	✓	x	YES	NO	YES	YES
VERIFY DATA AND DEL DATA	XFER	LOAD	✓	x	YES	YES	YES	YES
FORMAT TRACK	x	LOAD	✓	✓	YES	NO	YES	YES
SEEK	x	LOAD	y	x	YES	NO	YES	YES
READ DRIVE STATUS	x	—	x	x	NO	NO	NOTE 5	NO
SPECIFY	x	—	x	x	NO	NO	NO	NO
RESET	x	UNLOAD	x	x	NO	NO	NO	NO
R SP REGISTERS	x	—	x	x	NO	NO	NOTE 6	NO
W SP REGISTERS	x	—	x	x	NO	NO	NO	NO

Note: 1. "x" → DON'T CARE 2. "✓" → check 3. "—" → No change 4. "y" → Check at end of operation 5. See "READ DRIVE STATUS" command.  
6. See "READ SPECIAL REGISTER" command.

Explanation of the execution phase characteristics table.

**1. Deleted Data Processing**

If deleted data is encountered during an operation that is marked skip in the table, the deleted data record is not transferred into memory, but the record is counted. For example, if the command and parameters specify a read of five records and one of the records was written with a deleted data mark, four records are transferred to memory. The deleted data flag is set in the result byte. However, if the operation is marked transfer, all data is transferred to memory regardless of the type of data mark.

**2. Head**

The Head column in the table specifies whether the Read/Write head will be loaded or not. If the table specifies load, the head is loaded after it is positioned over the track. The head loaded by a command remains loaded until the user specified number of index pulses have occurred.

**3. Ready**

The Ready column indicates if the ready line (Ready 1, Ready 0) associated with the selected drive is checked. A not ready state is latched by the 8271 until the user executes a read status command.

**4. Write Protect**

The operations that are marked check Write Protect are immediately aborted if Write Protect line is active at the beginning of an operation.

**5. Seek**

Many of the 8271 commands cause a seek to the desired track. A current track register is maintained for each drive or surface.

**6. Seek Check**

Operations that perform Seek Check verify that selected data in the ID field is correct before the 8271 accesses the data field.



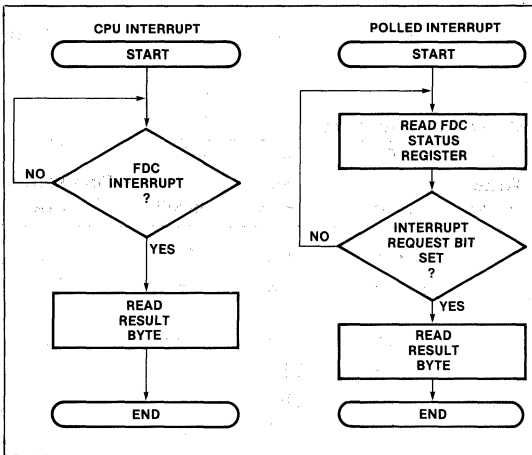


Figure 18. Getting the Result

### The Result Phase

During the Result Phase, the FDC notifies the CPU of the outcome of the command execution. This phase may be initiated by:

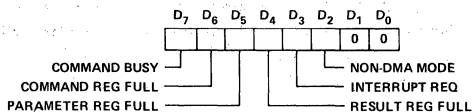
1. The successful completion of an operation.
2. An error detected during an operation.

## PROGRAMMING

A <sub>1</sub>	A <sub>0</sub>	CS RD	CS WR
0	0	Status Reg	Command Reg
0	1	Result Reg	Parameter Reg
1	0	—	Reset Reg
1	1	—	—

### STATUS REGISTER

#### FDC Status



#### Bit 7: Command Busy

The command busy bit is set on writing to the command register. Whenever the FDC is busy processing a command, the command busy bit is set to a one. This bit is set to zero after the command is completed.

#### Bit 6: Command Full

The command full bit is set on writing to the command buffer and cleared when the FDC begins processing the command.

#### Bit 5: Parameter Full

This bit indicates the state of the parameter buffer. This bit is set when a parameter is written to the FDC and reset after the FDC has accepted the parameter.

#### Bit 4: Result Full

This bit indicates the state of the result buffer. It is valid only after Command Busy bit is low. This bit is set when the FDC finishes a command and is reset after the result byte is read by the CPU. The data in the result buffer is valid only after the FDC has completed a command. Reading the result buffer while a command is in progress yields no useful information.

#### Bit 3: Interrupt Request

This bit reflects the state of the FDC INT pin. It is set when FDC requests attention as a result of the completion of an operation or failure to complete an intended operation. This bit is cleared by reading the result register.

#### Bit 2: Non-DMA Data Request

When the FDC is utilized without a DMA controller, this bit is used to indicate FDC data requests. Note that in the non-DMA mode, an interrupt is generated (interrupt request bit is set) with each data byte written to or read from the diskette.

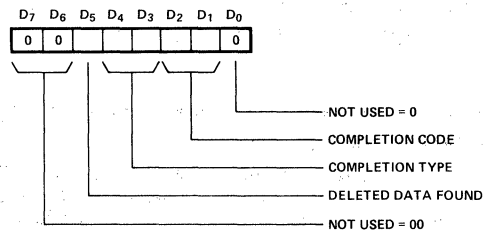
#### Bits 1 and 0:

Not used (zero returned).

After reading the Status Register, the CPU then reads the Result Register for more information.

### THE RESULT REGISTER

This byte format facilitates the use of an address table to look up error routines and messages. The standard result byte format is:



#### Bits 7 and 6:

Not used (zero returned).

#### Bit 5:

Deleted Data Found: This bit is set when deleted data is encountered during a transaction.

#### Bits 4 and 3: Completion Type

The completion type field provides general information regarding the outcome of an operation.

The completion type field provides general information regarding the outcome of an operation.

#### Completion Type

Completion Type	Event
00	Good Completion — No Error
01	System Error — recoverable errors; operator intervention probably required for recovery.
10	Command/Drive Error — either a program error or drive hardware failure.
11	Command/Drive Error — either a program error or drive hardware failure.

**Bits 2 and 1: Completion Code**

The completion code field provides more detailed information about the completion type (See Table).

Completion Type	Completion Code	Event
00	00	Good Completion/ Scan Not Met
00	01	Scan Met Equal
00	10	Scan Met Not Equal
00	11	---
01	00	Clock Error
01	01	Late DMA
01	10	ID CRC Error
01	11	Data CRC Error
10	00	Drive Not Ready
10	01	Write Protect
10	10	Track 0 Not Found
10	11	Write Fault
11	00	Sector Not Found
11	01	---
11	10	---
11	11	---

It is important to note the hierarchical structure of the result byte. In very simple systems where only a GO-NO GO result is required, the user may simply branch on a zero result (a zero result is a good completion). The next level of complexity is at the completion type interface. The completion type supplies enough information so that the software may distinguish between fatal and non-fatal errors. If a completion type 01 occurs, ten retries should be performed before the error is considered unrecoverable.

The Completion Type/Completion Code interface supplies the greatest detail about each type of completion. This interface is used when detailed information about the transaction completion is required.

**Bit 0:**  
Not used (zero returned).

**Table 3. Completion Code Interpretation**

Definition	Interpretation
Successful Completion/ Scan Not Met	The diskette operation specified was completed without error. If scan operation was specified, the pattern scanned was not found on the track addressed.
Scan Met Equal	The data pattern specified with the scan command was found on the track addressed with the specified comparison, and the equality was met.
Scan Met Not Equal	The data pattern specified with the scan command was found with the specified comparison on the track addressed, but the equality was not met.
Clock Error	During a diskette read operation, a clock bit was missing (dropped). Note that this function is disabled when reading any of the ID address marks (which contain missing clock pulses). If this error occurs, the operation is terminated immediately and an interrupt is generated.
Late DMA	During either a diskette read or write operation, the data channel did not respond within the allotted time interval to prevent data from being overwritten or lost. This error immediately terminates the operation and generates an interrupt.
ID Field CRC Error	The CRC word (two bytes) derived from the data read in an ID field did not match the CRC word written in the ID field when the track was formatted. If this error occurs, the associated diskette operation is prevented and no data is transferred.
Data Field CRC Error	During a diskette read operation, the CRC word derived from the data field read did not match the data field CRC word previously written. If this error occurs, the data read from the sector should be considered invalid.
Drive Not Ready	The drive addressed was not ready. This indication is caused by any of the following conditions: <ol style="list-style-type: none"> <li>1. Drive not powered up</li> <li>2. Diskette not loaded</li> <li>3. Non-existent drive addressed</li> <li>4. Drive went not ready during an operation</li> </ol> Note that this completion code is cleared only through an FDC read drive status command.
Write Protect	A diskette write operation was specified on a write protected diskette. The intended write operation is prevented and no data is written on the diskette.
Track 00 Not Found	During a seek to track 00 operation, the drive failed to provide a track 00 indication after being stepped 255 times.
Write Fault	This error is dependent on the drive supported and indicates that the fault input to the FDC has been activated by the drive.
Sector Not Found	Either the sector addressed could not be found within one complete revolution of the diskette (two index marks encountered) or the track address specified did not match the track address contained in the ID field. Note that when the track address specified and the track address read do not match, the FDC automatically increments its track address register (stepping the drive to the next track) and again compares the track addresses. If the track addresses still do not match, the track address register is incremented a second time and another comparison is made before the sector not found completion code is set.

**INITIALIZATION**

**Reset Command**

	A <sub>1</sub>	A <sub>0</sub>	D <sub>7</sub>	D <sub>6</sub>	D <sub>5</sub>	D <sub>4</sub>	D <sub>3</sub>	D <sub>2</sub>	D <sub>1</sub>	D <sub>0</sub>
PAR:	1	0	0	0	0	0	0	0	0	1
PAR:	1	0	0	0	0	0	0	0	0	0

Function: The Reset command emulates the action of the reset pin. It is issued by outputting a one followed by a zero to the Reset register.

1. The drive control signals are forced low.
2. An in-progress command is aborted.
3. The FDC status register flags are cleared.
4. The FDC enters an idle state until the next command is issued.

Reset must be active for 10 or more clock cycles.

**SPECIFY COMMAND**

Many of the interface characteristics of the FDC are specified by the systems software. Prior to initiating any drive operation command, the software must execute the three specify commands. There are two types of specify commands selectable by the first parameter issued.

**First Parameter Specify Type**

0DH	Initialization
10H	Load bad Tracks Surface '0'
18H	Load bad Tracks Surface '1'

The Specify command is used prior to performing any diskette operation (including formatting of a diskette) to define the drive's inherent operating characteristics and also is used following a formatting operation or installation of another diskette to define the locations of bad tracks. Since the Specify command only loads internal registers within the 8271 and does not involve an actual diskette operation, command processing is limited to only Command Phase. Note that once the operating characteristics and bad tracks have been specified for a given drive and diskette, redefining these values need only be done if a diskette with unique bad tracks is to be used or if the system is powered down.

**Initialization:**

	A <sub>1</sub>	A <sub>0</sub>	D <sub>7</sub>	D <sub>6</sub>	D <sub>5</sub>	D <sub>4</sub>	D <sub>3</sub>	D <sub>2</sub>	D <sub>1</sub>	D <sub>0</sub>
CMD:	0	0	0	0	1	1	0	1	0	1
PAR:	0	1	0	0	0	0	1	1	0	1
PAR:	0	1	STEP RATE*							
PAR:	0	1	HEAD SETTling TIME*							
PAR:	0	1	INDEX CNT BEFORE HEAD UNLOAD*				HEAD LOAD TIME*			

\*Note: Mini-floppy parameters are doubled.

- Parameter 0 — 0DH = Select Specify Initialization.
- Parameter 1 — D<sub>7</sub>-D<sub>0</sub> = Step Rate (0-255ms in 1ms steps).
- Parameter 2 — D<sub>7</sub>-D<sub>0</sub> = Head Settling Time (0-255ms in 1ms steps). {0-510ms in 2ms steps} () = standard, {} = mini
- Parameter 3 — D<sub>7</sub>-D<sub>4</sub> = Index Count — Specifies the number of Revolutions (0-14) which are to occur before the FDC automatically unloads the R/W head. If 15 is specified, the head remains loaded.
- D<sub>3</sub>-D<sub>0</sub> = Head Load Time (0-60ms in steps of 4ms). {0-120ms in 8ms steps} () = standard, {} = mini

**Load Bad Tracks**

	A <sub>1</sub>	A <sub>0</sub>	D <sub>7</sub>	D <sub>6</sub>	D <sub>5</sub>	D <sub>4</sub>	D <sub>3</sub>	D <sub>2</sub>	D <sub>1</sub>	D <sub>0</sub>
CMD:	0	0	0	0	1	1	0	1	0	1
PAR:	0	1	0	0	0	1	1/0	0	0	0
PAR:	0	1	BAD TRACK NO. 1							
PAR:	0	1	BAD TRACK NO. 2							
PAR:	0	1	CURRENT TRACK							

Parameter 0: 10H = Load Surface zero bad tracks  
18H = Load Surface one bad track

Parameter 1:  
Bad track address number 1 (Physical Address).

It is recommended to program both bad tracks and current track to FF<sub>H</sub> during initialization.

**SEEK COMMAND**

The seek command moves the head to the specified track without loading the head or verifying the track.

The seek operation uses the specified bad tracks to compute the physical track address. This feature insures that the seek operation positions the head over the correct track.

When a seek to track zero is specified, the FDC steps the head until the track 00 signal is detected.

If the track 00 signal is not detected within (FF)<sub>H</sub> steps, a track 0 not found error status is returned.

A seek to track zero is used to position the read/write head when the current head position is unknown (such as after a power up).

	A <sub>1</sub>	A <sub>0</sub>	D <sub>7</sub>	D <sub>6</sub>	D <sub>5</sub>	D <sub>4</sub>	D <sub>3</sub>	D <sub>2</sub>	D <sub>1</sub>	D <sub>0</sub>
CMD:	0	0	SEL 1	SEL 0	1	0	1	0	0	1
PAR:	0	1	TRACK ADDRESS 0-255							

Seek operations are not verified. A subsequent read or write operation must be performed to determine if the correct track is located.

**READ DRIVE STATUS COMMAND**

This command is used to interrogate the drive status. Upon completion the result register will hold the final drive status.

	A <sub>1</sub>	A <sub>0</sub>	D <sub>7</sub>	D <sub>6</sub>	D <sub>5</sub>	D <sub>4</sub>	D <sub>3</sub>	D <sub>2</sub>	D <sub>1</sub>	D <sub>0</sub>
CMD:	0	0	SEL 1	SEL 0	1	0	1	1	0	0

RESULT: EACH BIT INDICATES CURRENT STATE OF INPUT PINS.

A <sub>1</sub>	A <sub>0</sub>	D <sub>7</sub>	D <sub>6</sub>	D <sub>5</sub>	D <sub>4</sub>	D <sub>3</sub>	D <sub>2</sub>	D <sub>1</sub>	D <sub>0</sub>
0	1	RDY 1*	WR FAULT	INDEX	WR PROT	RDY 0*	TRACK CNT 0	OPI	

IF A DRIVE NOT READY RESULT IS RETURNED, THE READ STATUS MUST BE ISSUED TO CLEAR THE CONDITION.

\*Note the two ready bits are zero latching. Therefore, to clear the drive not ready condition, assuming the drive is ready, and to detect it via software, one must issue this command twice.

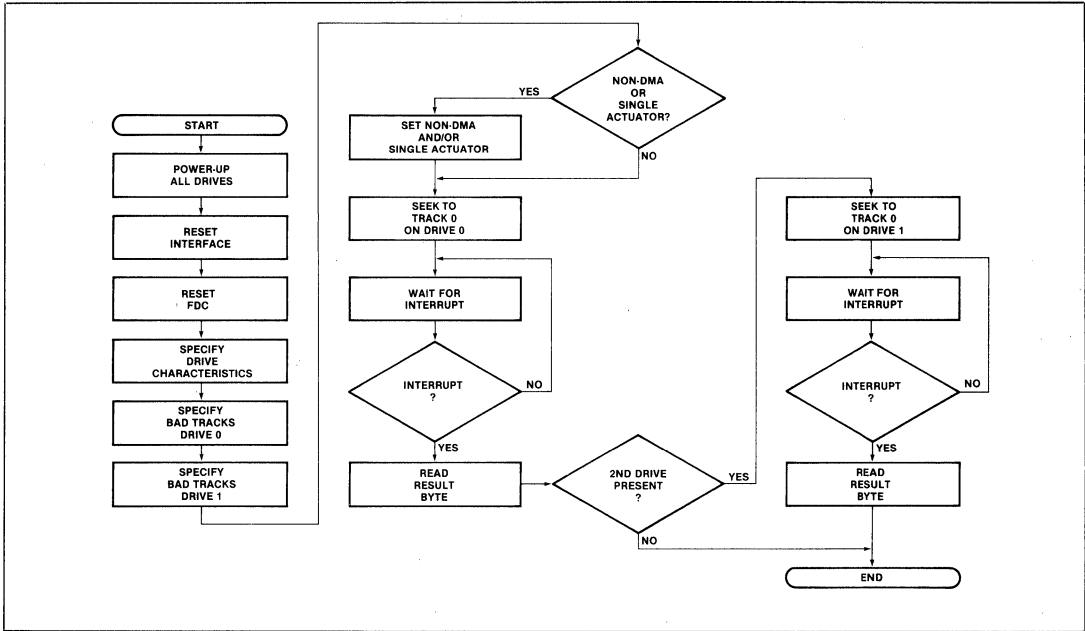


Figure 19. Initialization of the 8271 by the User

**Read/Write Special Registers**

This command is used to access special registers within the 8271.

	A <sub>1</sub>	A <sub>0</sub>	D <sub>7</sub>	D <sub>6</sub>	D <sub>5</sub>	D <sub>4</sub>	D <sub>3</sub>	D <sub>2</sub>	D <sub>1</sub>	D <sub>0</sub>
CMD:	0	0	SEL 1	SEL 0	COMMAND OPCODE					
PAR:	0	1	REGISTER ADDRESS							

Command code:

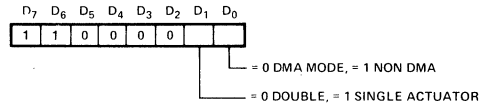
- 3DH Read Special Register
- 3AH Write Special Register

For both commands, the first parameter is the register address; for Write commands a second parameter specifies data to be written. Only the Read Special Register command supplies a result.

**Table 4. Special Registers**

Description	Register Address in Hex	Comment
Scan Sector Number	06	See Scan Description
Scan MSB of Count	14	See Scan Description
Scan LSB of Count	13	See Scan Description
Surface 0 Current Track	12	
Surface 1 Current Track	1A	
Mode Register	17	See Mode Register Description
Drive Control Output Port	23	See Drive Output Port Description
Drive Control Input Port	22	See Drive Input Port Description
Surface 0 Bad Track 1	10	
Surface 0 Bad Track 2	11	
Surface 1 Bad Track 1	18	
Surface 1 Bad Track 2	19	

**Mode Register Write Parameter Format**



**Bits 6 & 7**

Must be one.

**Bits 5-2**

(Not used). Must be set to zero.

**\*Bit 1**

**Double/Single Actuator:** Selects single or double actuator mode. If the single actuator mode is selected, the FDC assumes that the physical track location of both disks is always the same. This mode facilitates control of a drive which has a single actuator mechanism to move two heads.

**\*Bit 0**

**Data Transfer Mode:** This bit selects the data transfer mode. If this bit is a zero, the FDC operates in the DMA mode (DMA Request/ACK). If this bit is a one, the FDC operates in non-DMA mode. When the FDC is operating in DMA mode, interrupts are generated at the completion of commands. If the non-DMA mode is selected, the FDC generates an interrupt for every data byte transferred.

\*Bits 0 and 1 are initialized to zero.

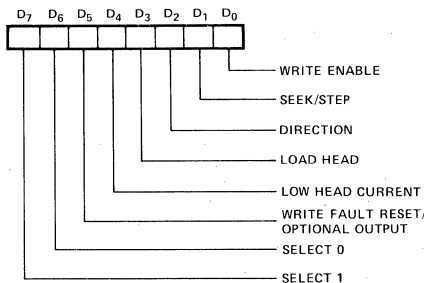
**Non-DMA Transfers in DMA Mode**

If the user desires, he may retain the use of interrupts generated upon command completions. This mode is accomplished by selecting the DMA capability, but using the DMA REQ/ACK pins as effective INT and CS signals, respectively.

**Drive Control Input Port**

Reading this port will give the CPU exactly the data that the FDC sees at the corresponding pins. Reading this port will update the drive not ready status, but will not clear the status. (See Read Drive Status Command for Bit locations.)

**Drive Control Output Port Format**



Each of these signals correspond to the chip pin of the same name. On standard-sized drives with write fault detection logic, bit 5 is set to generate the write fault reset signal. This signal is used to clear a write fault indication within the drive. On mini-sized drives, this bit can be used to turn on or off the drive motor prior to initiating a drive operation. A time delay after turn on may be necessary for the drive to come up to speed. The register must be read prior to writing the register in order to save the states of the remaining bits. When the register is subsequently written to modify bit 5, the remaining bits must be restored to their previous states.

**IBM DISKETTE GENERAL FORMAT INFORMATION**

The IBM Flexible Diskette used for data storage and retrieval is organized into concentric circular paths or TRACKS. There are 77 tracks on either one or both sides (surfaces) of the diskette. On double-sided diskettes, the corresponding top and bottom tracks are referred to as a CYLINDER. Each track is further divided into fixed length sections or SECTORS. The number of sectors per track — 26, 15 or 8 — is determined when a track is formatted and is dependent on the sector length — 128, 256 or 512 bytes respectively — specified.

All tracks on the diskette are referenced to a physical index mark (a small hole in the diskette). Each time the hole passes a photodetector cell (one revolution of the diskette), an Index pulse is generated to indicate the logical beginning of a track. This index pulse is used to initiate a track formatting operation.

**Track Format**

Each Diskette Surface is divided into 77 tracks with each track divided into fixed length sectors. A sector can hold a whole record or a part of a record. If the record is shorter than the sector length, the unused bytes are filled with binary zeros. If a record is longer than the sector length, the record is written over as many sectors as its length requires. The sector size that provides the most efficient use of diskette space can be chosen depending upon the record length required.

Tracks are numbered from 00 (outer-most) to 76 (inner-most) and are used as follows:

- TRACK 00 reserved as System Label Track
- TRACKS 01 through 74 used for data
- TRACKS 75 and 76 used as alternates.

Each sector consists of an ID field (which holds a unique address for the sector) and a data field.

The ID field is seven bytes long and is written for each sector when the track is formatted. Each ID field consists of an ID field Address Mark, a Cylinder Number byte which identifies the track number, a Head Number byte which specifies the head used (top or bottom) to access the sector, a Record Number byte identifying the sector number (1 through 26 for 128 byte sectors), an N-byte specifying the byte length of the sector and two CRC (Cyclic Redundancy Check) bytes.

The Gaps separating the index mark and the ID and data fields are written on a track when it is formatted. These gaps provide both an interval for switching the drive electronics from reading or writing and compensation for rotational speed and other diskette-to-diskette and drive-to-drive manufacturing tolerances to ensure that data written on a diskette by one system can be read by another (diskette interchangeability).

**IBM Format Implementation Summary**

**Track Format**

The disk has 77 tracks, numbered physically from 00 to 76, with track 00 being the outermost track. There are logically 75 data tracks and two alternate tracks. Any two tracks may be initialized as bad tracks. The data tracks are numbered logically in sequence from 00 to 74, skipping over bad tracks (alternate tracks replace bad tracks). Note: In IBM format track 00 cannot be a bad track.

**Sector Format**

Each track is divided into 26, 15, or 8 sectors of 128, 256, or 512 bytes length respectively. The first sector is numbered 01, and is physically the first sector after the physical index mark. The logical sequence of the remaining sectors may be nonsequential physically. The location of these is determined at initialization by CPU software.

Each sector consists of an ID field and a data field. All fields are separated by gaps. The beginning of each field is indicated by 6 bytes of (00)H followed by a one byte address mark.

**Address Marks**

Address Marks are unique bit patterns one byte in length which are used to identify the beginning of ID and Data fields. Address Mark bytes are unique from all other data

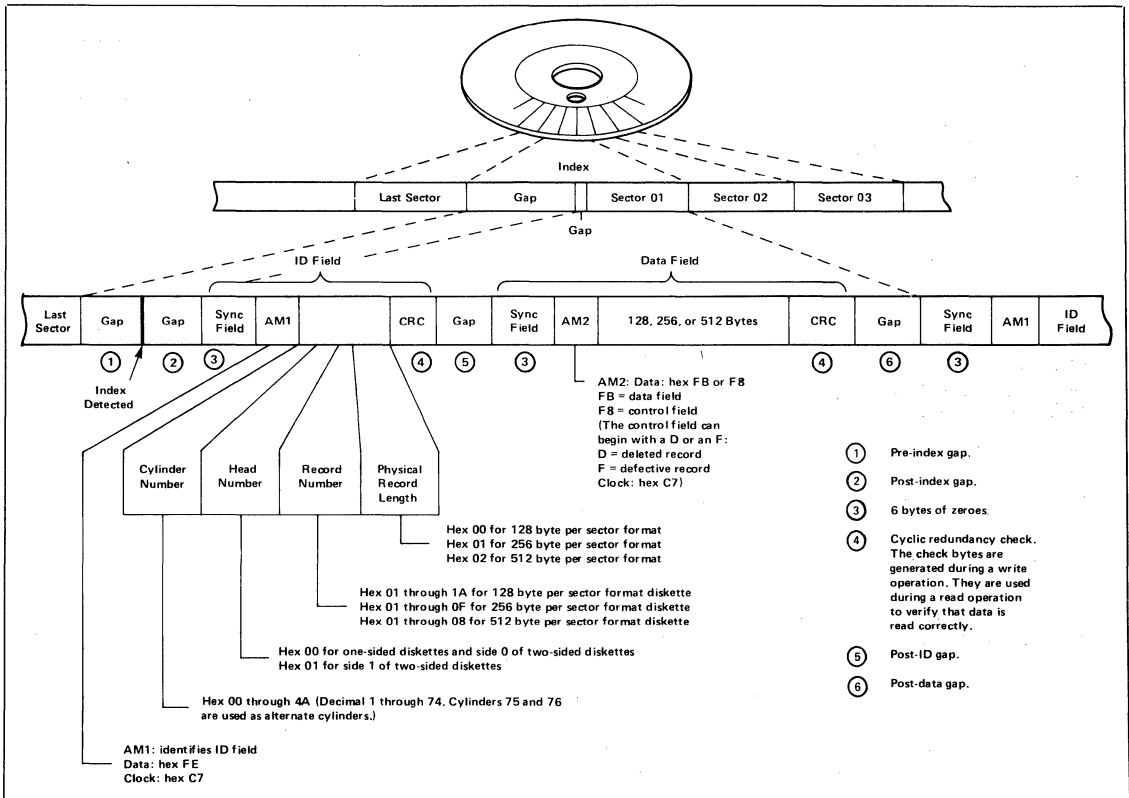


Figure 20. Track Format

bytes in that certain bit cells do not contain a clock bit (all other data bytes have clock bits in every bit cell.) There are four different types of Address Marks used. Each of these is used to identify different types of fields.

**Index Address Mark**

The Index Address Mark is located at the beginning of each track and is a fixed number of bytes in front of the first record.

**ID Address Mark**

The ID Address Mark byte is located at the beginning of each ID field on the diskette.

**Data Address Mark**

The Data Address Mark byte is located at the beginning of each non-deleted Data Field on the diskette.

**Deleted Data Address Mark**

The Deleted Data Address Mark byte is located at the beginning of each deleted Data Field on the diskette.

Address Mark Summary	Clock Pattern	Data Pattern
Index Address Mark	D7	FC
ID Address Mark	C7	FE
Data Address Mark	C7	FB
Deleted Data Address Mark	C7	F8
Bad Track ID Address Mark	C7	FE

**ID Field**

MARK	C	H	R	N	CRC	CRC
------	---	---	---	---	-----	-----

C = Cylinder (Track) Address, 00-74  
 H = Head Address  
 R = Record (Sector) Address, 01-26  
 N = Record (Sector) Length, 00-02  
 Note: Sector Length =  $128 \times 2^N$  bytes  
 CRC = 16 Bit CRC Character (See Below)

**Data Field**

MARK	DATA	CRC	CRC
------	------	-----	-----

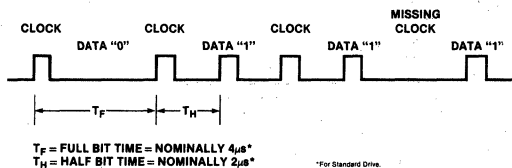
Data is 128, 256, or 512 bytes long.  
 Note: All marks, data, ID characters and CRC characters are recorded and read most significant bit first.

**CRC Character**

The 16-bit CRC character is generated using the generator polynomial  $X^{16} + X^{12} + X^5 + 1$ , normally initialized to (FF)<sub>H</sub>. It is generated from all characters (except the CRC in the ID or data field), including the data (not the clocks) in the address mark. It is recorded and read most significant bit first.

**Data Format**

Data is written (general case) in the following manner:



**References**

"The IBM Diskette for Standard Data Interchange," IBM Document GA21-9182-0. "System 32," Chapter 8, IBM Document GA21-9176-0.

**Bad Track Format**

The Bad Track Format is the same as the good track format except that the bad track ID field is initialized as follows:

$$C = H = R = N = (FF)_H$$

When formatting, bad track registers should be set to  $FF_H$  for the drive during the formatting, thus specifying no bad tracks. Thus, all tracks are left available for formatting.

The track following the bad track(s) should be one higher in number than track before the bad track(s).

Upon completion of the format the bad tracks should be set up using the write special register command. The 8271 will then generate an extra step pulse to cross the bad track, locating a new track that now happens to be an extra track out.

**Format Track**

**Format Command**

	A <sub>1</sub>	A <sub>0</sub>	D <sub>7</sub>	D <sub>6</sub>	D <sub>5</sub>	D <sub>4</sub>	D <sub>3</sub>	D <sub>2</sub>	D <sub>1</sub>	D <sub>0</sub>
CMD:	0	0	SEL 1	SEL 0	1	0	0	0	1	1
PAR:	0	1	TRACK ADDRESS							
PAR:	0	1	GAP 3 SIZE MINUS 6							
PAR:	0	1	RECORD LENGTH				NO. OF SECTORS/TRACK			
PAR:	0	1	GAP 5 SIZE MINUS 6							
PAR:	0	1	GAP 1 SIZE MINUS 6							

The format command can be used to initialize a disk track compatible with the IBM 3740 format. A Shugart "IBM Type" mini-floppy format may also be generated.

The Format command can be used to initialize a diskette, one track at a time. When format command is used, the program must supply ID fields for each sector on the track. During command execution, the supplied ID fields (track head sector addresses and the sector length) are written sequentially on the diskette. The ID address marks originate from the 8271 and are written automatically as the first byte of each ID field. The CRC character is written in the last two bytes of the ID field and is derived from the data written in the first five bytes. During the formatting operation, the data field of each sector is filled with data pattern  $(E5)_H$ . The CRC, derived from the data pattern is also appended to the last byte.

1. The parameter 2 ( $D_7 - D_9$ ) of the Format command specify record length, the bits are coded the same way as in the Read Data commands.
2. The programmable gap sizes (gap 3, gap 5, and gap 1) must be programmed such that the 6 bytes of zero (sync) are subtracted from the intended gap size i.e., if gap 1 is intended to be 16 bytes long, programmed length must be  $16 - 6 = 10$  bytes (of  $FF_H$ 's).

**Mini-Floppy Disk Format**

The mini-floppy disk format differs from the standard disk format in the following ways:

1. Gap 5 and the Index Address mark have been eliminated.
2. There are fewer sectors/tracks.

**GAPS**

The following is the gap size and description summary:

- Gap 1 Programmable
- Gap 2 17 Bytes
- Gap 3 Programmable
- Gap 4 Variable
- Gap 5 Programmable

The last six bytes of gaps 1,2,3 and 5 are  $(00)_H$ , all other bytes in the gaps are  $(FF)_H$ . The Gap 1,3 and 5 count specified by the user are the number of bytes of  $(FF)_H$ . Gap 4 is written until the leading edge of the index pulse. If a Gap 5 size of zero is specified, the Index Mark is not written.

**Gap 1:** This gap separates the index address mark of the index pulse from the first ID mark. It is used to protect the first ID field from a write on the last physical sector of the current track.  
 N bytes  $FF$ 's  
 6 bytes 0's for sync

**Gap 2:** This gap separates the ID field from the data mark and field such that during a write only the data field will be changed even if the write gate turns on early, due to drive speed changes.  
 11 bytes  $FF$ 's  
 6 bytes 0's for sync

**Gap 3:** This gap separates a data area from the next ID field. It is used so that during drive speed changes the next ID mark will not be overwritten, thus causing loss of data.  
 N bytes  $FF$ 's  
 6 bytes 0's for sync

**Gap 4:** This gap fills out the rest of the disk and is used for slack during formatting. During drive speed variations this gap will shrink or grow if the disk is re-formatted.  
 $FF$ 's only

**Gap 5:** This gap separates the last sector from the Index Address mark and is used to assure that the index address mark is not destroyed by writing on the last physical data sector on the track.  
 N bytes  $FF$ 's  
 6 bytes 0's for sync

The number of  $FF$  bytes is programmable for gaps 1, 3 and 5.

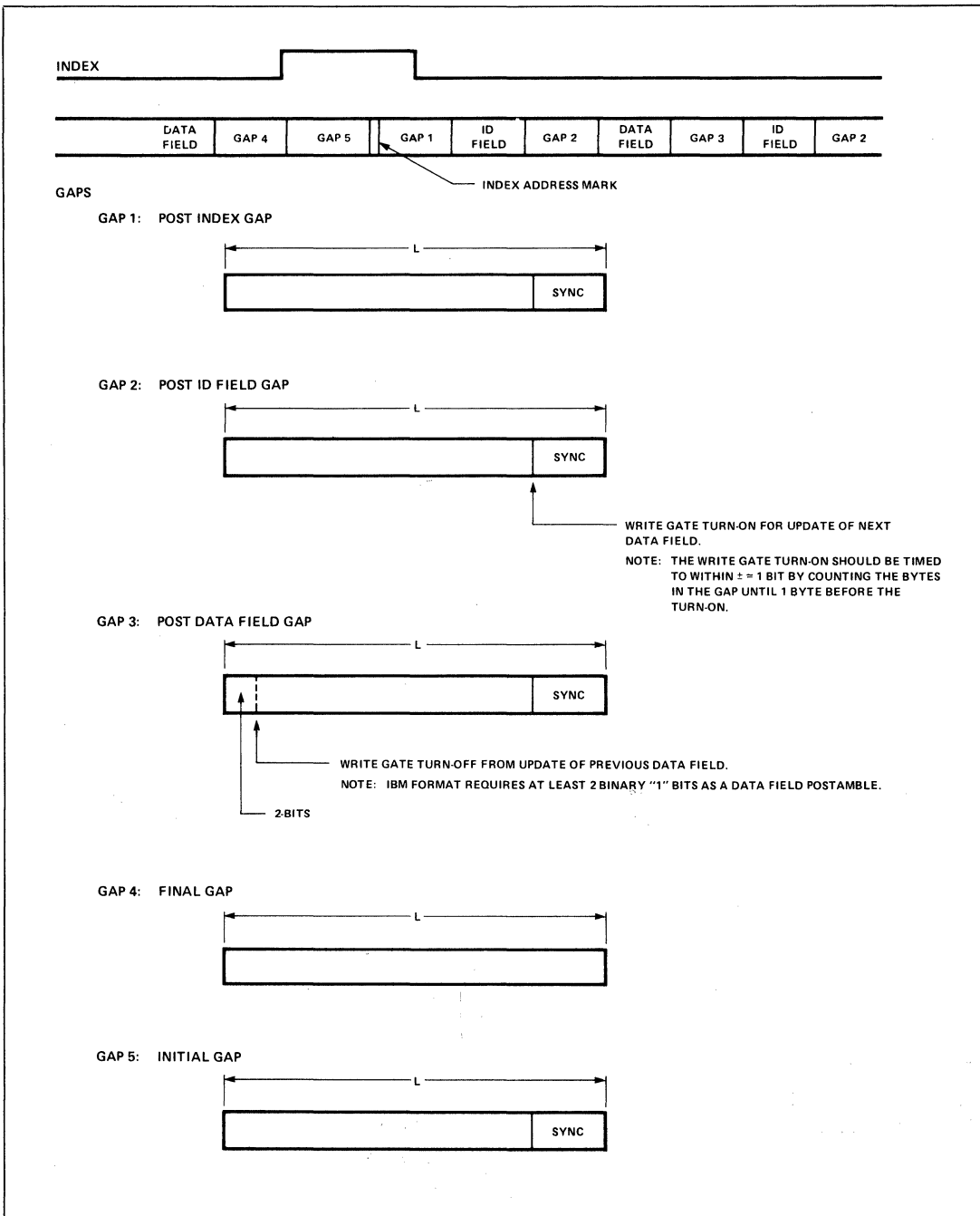


Figure 21. Track Format



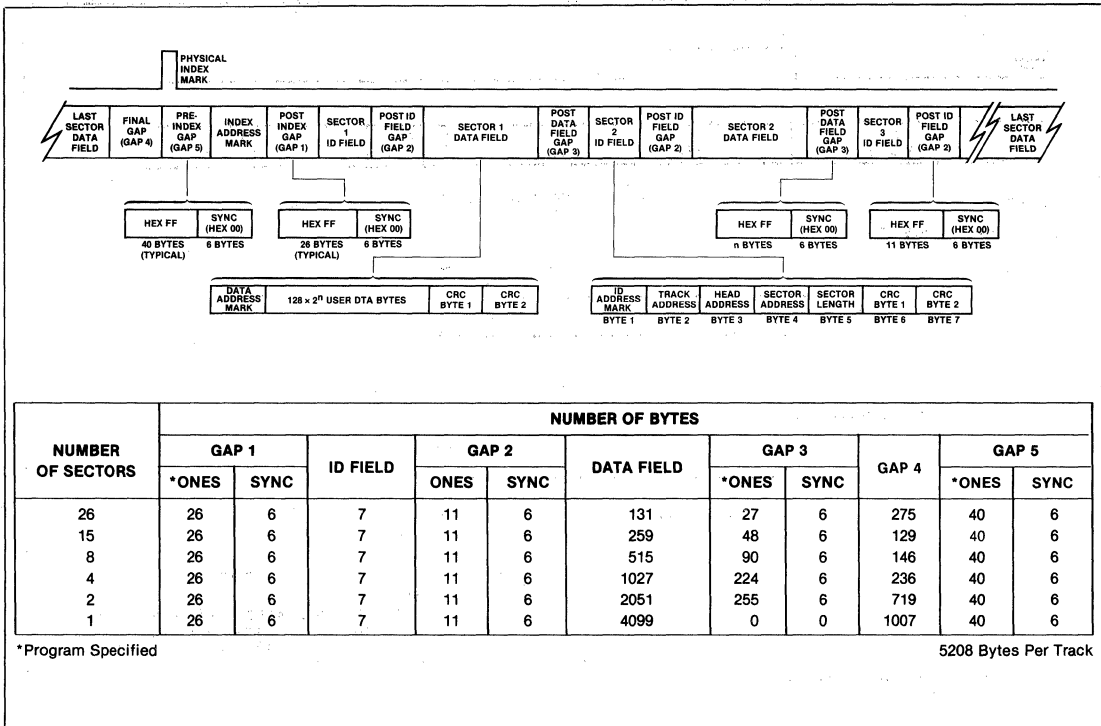


Figure 22. Standard Diskette Track Format

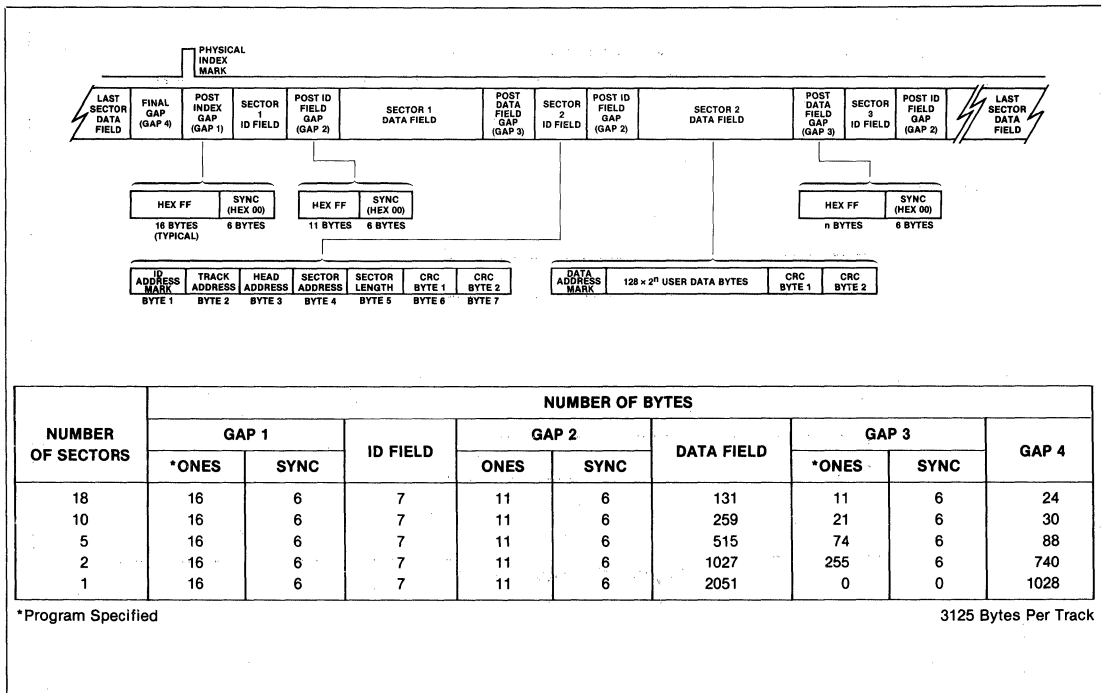


Figure 23. Mini-Diskette Track Format

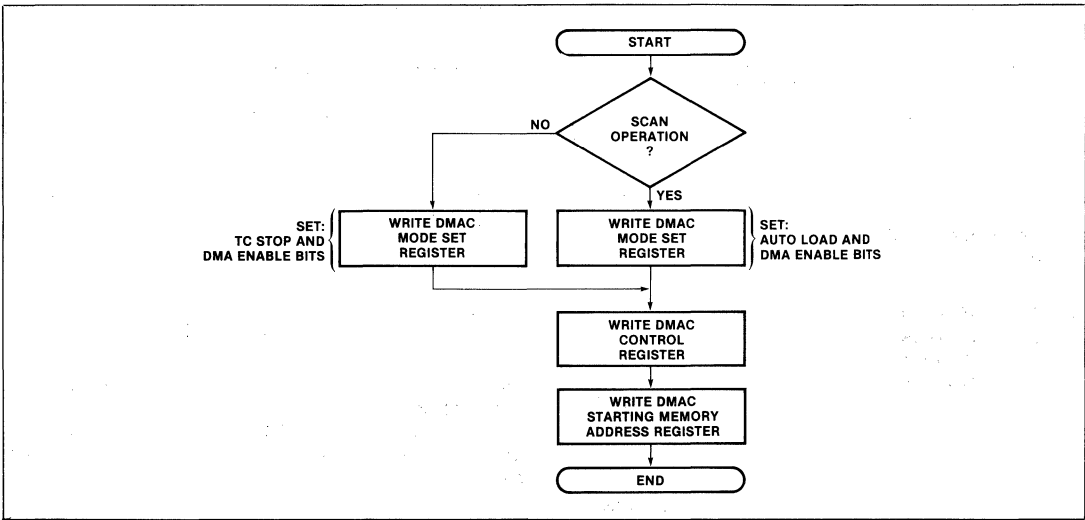


Figure 24. User DMA Channel Initialization Flowchart

**Read ID Command**

	A <sub>1</sub>	A <sub>0</sub>	D <sub>7</sub>	D <sub>6</sub>	D <sub>5</sub>	D <sub>4</sub>	D <sub>3</sub>	D <sub>2</sub>	D <sub>1</sub>	D <sub>0</sub>
CMD:	0	0	SEL 1	SEL 0	0	1	1	0	1	1
PAR:	0	1	TRACK ADDRESS							
PAR:	0	1	0	0	0	0	0	0	0	0
PAR:	0	1	NUMBER OF ID FIELDS							

The Read ID command transfers the specified number of ID fields into memory (beginning with the first ID field after Index). The CRC character is checked but not transferred.

These fields are entered into memory in the order in which they are physically located on the disk, with the first field being the one starting at the index pulse.

**Data Processing Commands**

All the routine Read/Write commands examine specific drive status lines before beginning execution, perform an implicit seek to the track address and load the drive's read/write head. Regardless of the type of command (i.e., read, write or verify), the 8271 first reads the ID field(s) to verify that the correct track has been located (see sector not found completion code) and also to locate the addressed sector. When a transfer is complete (or cannot be completed), the 8271 sets the interrupt request bit in the status register and provides an indication of the outcome of the operation in the result register.

If a CRC error is detected during a multisector transfer, processing is terminated with the sector in error. The address of the failing sector number can be determined by examining the Scan Sector Number register using the Read Special Register command.

Full power of the multisector read/write commands can be realized by doing DMA transfer using Intel® 8257 DMA Controller. For example, in a 128 byte per sector multisector write command, the entire data block (containing 128 bytes times the number of sectors) can be located in a disk memory buffer. Upon completion of the command phase, the 8271 begins execution by accessing the desired track, verifying the ID field, and locating the data field of the first record to be written. The 8271 then DMA-accesses the first sector and starts counting and writing one byte at a time until all 128 bytes are written. It then locates the data field of the next sector and repeats the procedure until all the specified sectors have been written. Upon completion of the execution phase the 8271 enters into the result phase and interrupts the CPU for availability of status and completion results. Note that all read/write commands, single or multisector are executed without CPU intervention.

Note, execution of multi-sector operations are faster if the sectors are *not* interleaved.

**128 Byte Single Record Format**

	A <sub>1</sub>	A <sub>0</sub>	D <sub>7</sub>	D <sub>6</sub>	D <sub>5</sub>	D <sub>4</sub>	D <sub>3</sub>	D <sub>2</sub>	D <sub>1</sub>	D <sub>0</sub>
CMD:	0	0	SEL 1	SEL 0	COMMAND OPCODE					
PAR:	0	1	TRACK ADDR 0-255							
PAR:	0	1	SECTOR 0-255							

**Commands**

- READ DATA 12
- READ DATA AND DELETED DATA 16
- WRITE DATA 0A
- WRITE DELETED DATA 0E
- VERIFY DATA AND DELETED DATA 1E

**Opcode**

**Variable Length/Multi-Record Format**

	A <sub>1</sub>	A <sub>0</sub>	D <sub>7</sub>	D <sub>6</sub>	D <sub>5</sub>	D <sub>4</sub>	D <sub>3</sub>	D <sub>2</sub>	D <sub>1</sub>	D <sub>0</sub>
CMD:	0	0	SEL 1	SEL 0	COMMAND OPCODE					
PAR:	0	1	TRACK ADDR 0-255							
PAR:	0	1	SECTOR 0-255							
PAR:	0	1	LENGTH				NO. OF SECTORS			

D<sub>7</sub>-D<sub>5</sub> of Parameter 2 determine the length of the disk record.

0 0 0	128 Bytes
0 0 1	256 Bytes
0 1 0	512 Bytes
0 1 1	1024 Bytes
1 0 0	2048 Bytes
1 0 1	4096 Bytes
1 1 0	8192 Bytes
1 1 1	16,384 Bytes

**Commands**

**Opcode**

READ DATA	13
READ DATA AND DELETED DATA	17
WRITE DATA	0B
WRITE DELETED DATA	0F
VERIFY DATA AND DELETED DATA	1F
SCAN DATA	00
SCAN DATA AND DELETED DATA	04

**Read Commands**

Read Data, Read Data and Deleted Data.

**Function**

The read command transfers data from a specified disk record or group of records to memory. The operation of this command is outlined in execution phase table.

**Write Commands**

Write Data, Write Deleted Data.

**Function**

The write command transfers data from memory to a specified disk record or group of records.

**Verify Command**

Verify Data and Deleted Data.

**Function**

The verify command is identical to the read data and deleted data command except that the data is not transferred to memory. This command is used to check that a record or a group of records has been written correctly by verifying the CRC character.

**Scan Commands**

	A <sub>1</sub>	A <sub>0</sub>	D <sub>7</sub>	D <sub>6</sub>	D <sub>5</sub>	D <sub>4</sub>	D <sub>3</sub>	D <sub>2</sub>	D <sub>1</sub>	D <sub>0</sub>
CMD:	0	0	SEL 1	SEL 0	0	0	0	S.DATA S.DELED	0	0
PAR:	0	1	TRACK ADDR 0-255							
PAR:	0	1	SECTOR 0-255							
PAR:	0	1	LENGTH				NO. OF SECTORS			
PAR:	0	1	SCAN TYPE				STEP SIZE			
PAR:	0	1	FIELD LENGTH (KEY)							

Command D<sub>2</sub> = 0 Scan Data  
D<sub>2</sub> = 1 Scan Data and Deleted Data

Scan Commands, Scan Data and Scan Data and Deleted Data, are used to search a specific data pattern or "key" from memory. The 8271 FDC operation during a scan is unique in that data is read from memory and from the diskette simultaneously.

During the scan operation, the key is compared repetitively (using the 8257 DMA Controller in auto load mode) with the data read from the diskette (e.g., an eight byte key would be compared with the first eight bytes (1-8) read from the diskette, the second eight bytes (9-16), the third eight bytes (17-24), etc.). The scan operation is concluded when the key is located or when the specified number of sectors have been searched without locating the key. When concluded, the 8271 FDC requests an interrupt. The program must then read the result register to determine if the scan was successful (if the key was located). If successful, several of the FDC's special registers can be examined (read special registers command) to determine more specific information relating to the scan (i.e., the sector number in which the key was located, and the number of bytes within the sector that were not compared when the key was located).

The 8271 does not do a sliding scan, it does a fixed block linear search. This means the key in memory is compared to an equal length block in a sector; when these blocks meet the scan conditions the scan will stop. Otherwise, the scan continues until all the sectors specified have been searched.

The following factors regarding key length must be considered when establishing a key in memory.

1. When searching multiple sectors, the length of the key must be evenly divisible into the sector length to prevent the key from being split at subsequent sector boundaries. Since the character FFH is not compared, the key in memory can be padded to the required length using this character. For example, if the actual pattern compared on the diskette is twelve characters in length, the field length should be sixteen and four bytes of FFH

would be appended to the key. Consequently, the last block of sixteen bytes compared within the first sector would end at the sector boundary and the first byte of the next sector would be compared with the first byte of the key. Splitting data over sector boundaries will not work properly since the FDC expects the start of key at each sector boundary.

- Since the first byte of the key is compared with the first byte of the sector, when the pattern does not begin with the first byte of the sector, the key must be offset using the character FF<sub>16</sub>. For example, if the first byte of a nine byte pattern begins on the fifth byte of the sector, four bytes of FF<sub>16</sub> are prefixed to the key (and three bytes of FF<sub>16</sub> are appended to the key to meet the length requirement) so that the first actual comparison begins on the fifth byte.

The Scan Commands require five parameters:

**Parameter 0, Track Address**

Specifies the track number containing the sectors to be scanned. Legal values range from 00<sub>H</sub> to 4C<sub>H</sub> (0 to 76) for a standard diskette and from 00<sub>H</sub> to 22<sub>H</sub> (0 to 34) for a mini-sized diskette.

**Parameter 1, Sector Address**

Specifies the first sector to be scanned. The number of sectors scanned is specified in parameter 2, and the order in which sectors are scanned is specified in parameter 3.

**Parameter 2, Sector Length/Number of Sectors**

The sector length field (bits 7-5) specifies the number of data bytes allocated to each sector (see parameter 2, routine read and write commands for field interpretation). The number of sectors field (bits 4-0) specifies the number of sectors to be scanned. The number specified ranges from one sector to the physical number of sectors on the track.

**Parameter 3**

- D<sub>7</sub>-D<sub>6</sub>: Indicate scan type
- 00-EQ Scan for each character within the field length (key) equal to the corresponding character within the disk sector. The scan stops after the first equal condition is met.
  - 01-GEQ Scan for each character within the disk sector greater than or equal to the corresponding character within the field length (key). The scan stops after the first greater than or equal condition is met.

- 10-LEQ Scan for each character within the disk sector less than or equal to the corresponding character within the field length (key). The scan stops after the first less than or equal condition is met.

- D<sub>5</sub>-D<sub>0</sub>: Step Size: The Step Size field specifies the offset to the next sector in a multisector scan. In this case, the next sector address is generated by adding the Step Size to the current sector address.

**Parameter 4, Field Length**

Specifies the number of bytes to be compared (length of key). While the range of legal values is from 1 to 255, the field length specified should be evenly divisible into the sector length to prevent the key from being split at sector boundaries, if the multisector scan commands are used.

**Scan Command Results**

More detailed information about the completion of Scan Commands may be obtained by executing Read Special Register commands.

**Read Special Register**

Parameter (Hex)	Results
06	The <u>sector number</u> of the sector in which the specified scan data pattern was located.
14	MSB Count — The number of 128 byte blocks remaining to be compared in the current sector when the scan data pattern was located. This register is decremented with each 128 byte block read.
13	LSB Count — The number of bytes remaining to be compared in the current sector when the scan data pattern is located. This register is initialized to 128 and is decremented with each byte compared.

Upon a scan met condition, the equation below can be used to determine the last byte in the located pattern.

$$\text{Pointer} = \text{sector length} - ((\text{Register 14H}) * 128 + (\text{Register 13H}))$$

**8271 Scan Command Example**

Assume there are only 2 records on track 0 with the following data:

Record 01: 01 02 03 04 05 06 07 08 000...00

Record 02: 01 02 AA 55 00 00 00 00 .....00

Command	Field Length <sup>[1]</sup>	Starting Sector #	# of Sectors	Key <sup>[2]</sup>	Completion Code <sup>[3]</sup>	Special Registers <sup>[4]</sup>			Comment
						R06	R14	R13	
* SCAN EQ	2	1	1	01,02	SME	01	0	127D	Met in first field
SCAN EQ	2	1	1	02,03	SNM	X	X	X	Not met
SCAN EQ	2	1	1	FF <sup>[5]</sup> ,05	SNM	X	X	X	Not met with don't care
* SCAN EQ	2	1	1	FF <sup>[5]</sup> ,06	SME	01	0	123D	Met with don't care
* SCAN EQ	2	1	2	AA,55	SME	02	0	125D	Met in Record 02
* SCAN EQ	2	2	1	01,02	SME	02	0	127D	Starting sector ≠ 1
* SCAN EQ	4	1	1	05,06,07,08	SME	01	0	121D	Field, Key length = 4
* SCAN GEQ	4	1	1	05,06,07,08	SME	01	0	121D	GEQ-SME
* SCAN GEQ	4	1	1	05,04,07,08	SMNE	01	0	121D	GEQ-SMNE
* SCAN GEQ	4	1	2	00,03,AA,44 <sup>[6]</sup>	SNM	X	X	X	GEQ-SNM
* SCAN LEQ	4	1	1	01,03,FF,04	SMNE	01	0	125D	LEQ-SMNE
* SCAN LEQ	4	1	1	01,02,FF,04	SME	01	0	125D	LEQ-SME

NOTES:

- Field Length — Each record is partitioned into a number of fields equal to the record size divided by the field length. Note that the record size should be evenly divisible by the field length to insure proper operation of multi record scan. Also, maximum field length = 256 bytes.
- Key — The key is a string of bytes located in the user system memory. The key length should equal the field length. By programming the 8257 DMA Controller into the auto load mode, the key will be recursively read in by the chip (once per field).
- Completion Code — Shows how Scan command was met or not met.  
 SNM — SCAN Not Met — 0 0 (also Good Complete)  
 SME — SCAN Met Equal — 0 1  
 SMNE — SCAN Met Not Equal — 1 0
- Special Registers  
 R06 — This register contains the record number where the scan was met.  
 R14 — This register contains the MSB count and is decremented every 128 characters.

Length ( ℓ ) (D7-D5 of PAR 2)	Record Size	$R14 = 2^{\ell} - 1$ (Initialize at Beginning of Record)
000	128 Bytes	0
001	256 Bytes	1
010	512 Bytes	3
011	1024 Bytes	7
•	•	•
•	•	•
•	•	•

R13 — This register contains a modulo 128 LSB count which is initialized to 128 at beginning of each record. This count is decremented after each character is compared except for the last character in a pattern match situation.

- The OFFH character in the key is treated as a don't care character position.
- The Scan comparison is done on a byte by byte basis. That is, byte 1 of each field is compared to byte 1 of the key, byte 2 of each field is compared to byte 2 of the key, etc.

**ABSOLUTE MAXIMUM RATINGS\***

Ambient Temperature Under Bias . . . . . 0°C to 70°C<sup>1</sup>  
 Storage Temperature . . . . . – 65°C to + 150°C  
 Voltage on Any Pin with  
 Respect to Ground . . . . . – 0.5V to + 7V  
 Power Dissipation . . . . . 1 Watt

*\*NOTICE: Stresses above those listed under “Absolute Maximum Ratings” may cause permanent damage to the device. This is a stress rating only and functional operation of the device at these or any other conditions above those indicated in the operational sections of this specification is not implied. Exposure to absolute maximum rating conditions for extended periods may affect device reliability.*

**D.C. CHARACTERISTICS** ( $V_{CC} = +5.0V \pm 5\%$   
 8271 and 8271-8:  $T_A = 0^\circ C$  to  $70^\circ C$ ; 8271-6:  $T_A = 0^\circ C$  to  $50^\circ C$ )

Symbol	Parameter	Min.	Max.	Unit	Test Conditions
$V_{IL}$	Input Low Voltage	– 0.5	0.8	V	
$V_{IH}$	Input High Voltage	2.0	( $V_{CC} + 0.5$ )	V	
$V_{OLD}$	Output Low Voltage (Data Bus)		0.45	V	$I_{OL} = 2.0$ mA
$V_{OLI}$	Output Low Voltage (Interface Pins)		0.5	V	$I_{OL} = 1.6$ mA
$V_{OH}$	Output High Voltage	2.4		V	$I_{OH} = -220$ $\mu$ A
$I_{IL}$	Input Load Current		$\pm 10$	$\mu$ A	$V_{IN} = V_{CC}$ to 0V
$I_{OZ}$	Off-State Output Current		$\pm 10$	$\mu$ A	$V_{OUT} = V_{CC}$ to 0V
$I_{CC}$	$V_{CC}$ Supply Current		180	mA	

**CAPACITANCE** ( $T_A = 25^\circ C$ ;  $V_{CC} = GND = 0V$ )

Symbol	Parameter	Min.	Typ.	Max.	Unit	Test Conditions
$C_{IN}$	Input Capacitance			10	pF	$t_c = 1$ MHz
$C_{I/O}$	I/O Capacitance			20	pF	Unmeasured Pins Returned to GND

**NOTE:** 1. Ambient temperature under bias for 8271-6 is  $0^\circ C$  to  $50^\circ C$ .

**A.C. CHARACTERISTICS** ( $V_{CC} = +5.0V \pm 5\%$   
 (8271 and 8271-8:  $T_A = 0^\circ C$  to  $70^\circ C$ ; 8271-6:  $T_A = 0^\circ C$  to  $50^\circ C$ )

**READ CYCLE**

Symbol	Parameter	Min.	Max.	Unit	Test Conditions
$t_{AC}$	Select Setup to $\overline{RD}$	0		ns	Note 2
$t_{CA}$	Select Hold from $\overline{RD}$	0		ns	Note 2
$t_{RR}$	$\overline{RD}$ Pulse Width	250		ns	
$t_{AD}$	Data Delay from Address		250	ns	Note 2
$t_{RD}$	Data Delay from $\overline{RD}$		150	ns	$C_L = 150$ pF, Note 2
$t_{DF}$	Output Float Delay	20	100	ns	$C_L = 20$ pF for Minimum; 150 pF for Maximum
$t_{DC}$	DACK Setup to $\overline{RD}$	25		ns	
$t_{CD}$	DACK Hold from $\overline{RD}$	25		ns	
$t_{KD}$	Data Delay from DACK		250	ns	

**A.C. CHARACTERISTICS (Continued)**
**WRITE CYCLE**

Symbol	Parameter	Min.	Max.	Unit	Test Conditions
$t_{AC}$	Select Setup to $\overline{WR}$	0		ns	
$t_{CA}$	Select Hold from $\overline{WR}$	0		ns	
$t_{WW}$	$\overline{WR}$ Pulse Width	250		ns	
$t_{DW}$	Data Setup to $\overline{WR}$	150		ns	
$t_{WD}$	Data Hold from $\overline{WR}$	0		ns	
$t_{DC}$	DACK Setup to $\overline{WR}$	25		ns	
$t_{CD}$	DACK Hold from $\overline{WR}$	25		ns	

**DMA**

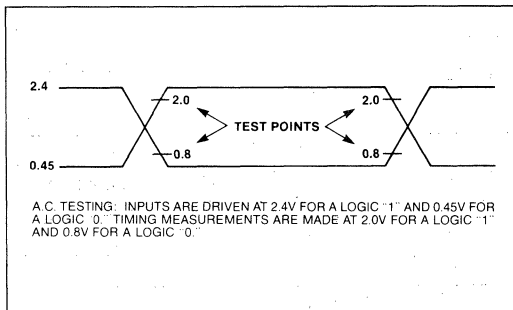
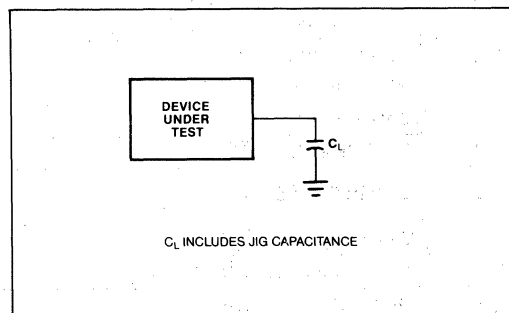
Symbol	Parameter	Min.	Max.	Unit	Test Conditions
$t_{CQ}$	Request Hold from $\overline{WR}$ or $\overline{RD}$ (for Non-Burst Mode)		150	ns	

**OTHER TIMINGS**

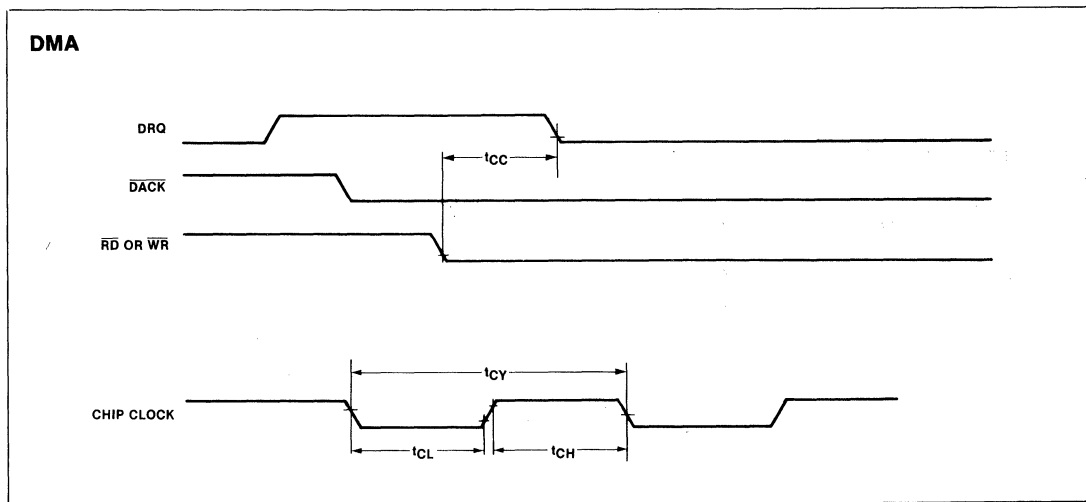
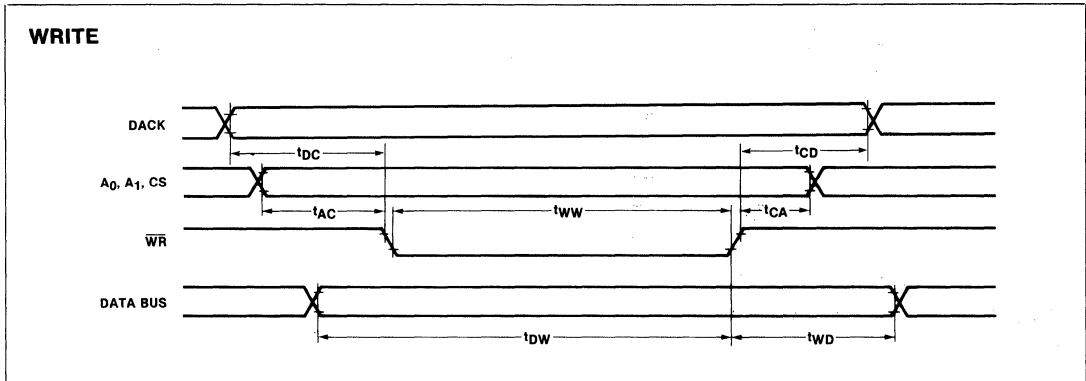
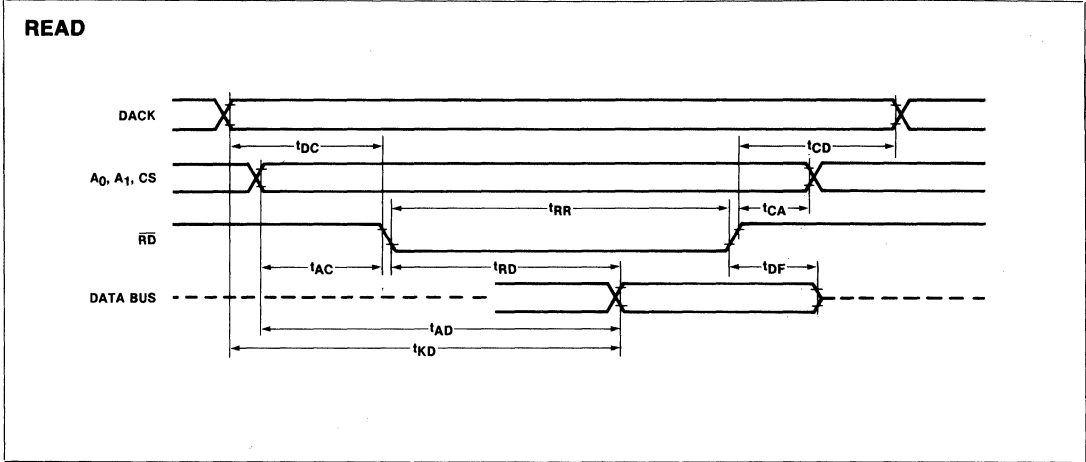
Symbol	Parameter	8271/8271-6		Unit	Test Conditions
		Min.	Max.		
$t_{RSTW}$	Reset Pulse Width	10		$t_{CY}$	
$t_r$	Input Signal Rise Time		20	ns	
$t_f$	Input Signal Fall Time		20	ns	
$t_{RSTS}$	Reset to First IOWR	2		$t_{CY}$	
$t_{CY}$	Clock Period	250			Note 3
$t_{CL}$	Clock Low Period	110		ns	
$t_{CH}$	Clock High Period	125		ns	
$t_{DS}$	Data Window Setup to Unseparated Clock and Data	50		ns	
$t_{DH}$	Data Window Hold from Unseparated Clock and Data	0		ns	

**NOTES:**

- All timing measurements are made at the reference voltages unless otherwise specified: Input "1" at 2.0V, "0" at 0.8V  
Output "1" at 2.0V, "0" at 0.8V
- $t_{AD}$ ,  $t_{RD}$ ,  $t_{AC}$ , and  $t_{CA}$  are not concurrent specs.
- Standard Floppy:  $t_{CY} = 250 \text{ ns} \pm 0.4\%$       Mini-Floppy:  $t_{CY} = 500 \text{ ns} \pm 0.4\%$

**A.C. TESTING INPUT, OUTPUT WAVEFORM**

**A.C. TESTING LOAD CIRCUIT**


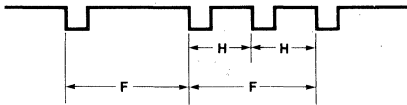
WAVEFORMS





WAVEFORMS (Continued)

READ DATA



\* $t_{CY} = 250 \text{ ns}$       \*\* $t_{CY} = 500 \text{ ns}$

$F = 16 t_{CY} \pm 8 t_{CY}$   
 $H = 8 t_{CY} \pm 4 t_{CY}$

\*STANDARD FLEXIBLE DISK DRIVE TIMING  
 \*\*MINI-FLOPPY TIMING

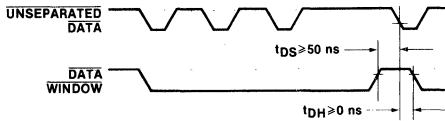
WRITE DATA



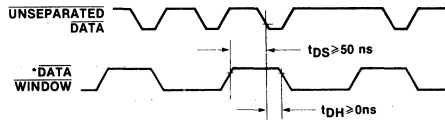
PULSE WIDTH  $PW = t_{CY} \pm 30 \text{ ns}$   
 $H$  (HALF BIT CELL) =  $8 t_{CY}$   
 $F$  (FULL BIT CELL) =  $16 t_{CY}$

\* $t_{CY} = 250 \text{ ns} \pm 0.4\%$       \*\* $t_{CY} = 500 \text{ ns} \pm 0.4\%$   
 $250 \text{ ns} \pm 30 \text{ ns}$                        $500 \text{ ns} \pm 30 \text{ ns}$   
 $2.0 \mu\text{s} \pm 8 \text{ ns}$                          $4.0 \mu\text{s} \pm 16 \text{ ns}$   
 $4.0 \mu\text{s} \pm 16 \text{ ns}$                          $8.0 \mu\text{s} \pm 32 \text{ ns}$

SINGLE-SHOT DATA SEPARATOR



PLO DATA SEPARATOR



\*DATA WINDOW MAY BE 180° OUT OF PHASE  
 IN PLO DATA SEPARATION MODE.



# 8272 SINGLE/DOUBLE DENSITY FLOPPY DISK CONTROLLER

- IBM Compatible in Both Single and Double Density Recording Formats
- Programmable Data Record Lengths: 128, 256, 512, or 1024 Bytes/Sector
- Multi-Sector and Multi-Track Transfer Capability
- Drive Up to 4 Floppy Disks
- Data Scan Capability — Will Scan a Single Sector or an Entire Cylinder's Worth of Data Fields, Comparing on a Byte by Byte Basis, Data in the Processor's Memory with Data Read from the Diskette
- Data Transfers in DMA or Non-DMA Mode
- Parallel Seek Operations on Up to Four Drives
- Compatible with Most Microprocessors Including 8080A, 8085A, 8086 and 8088
- Single-Phase 8 MHz Clock
- Single +5 Volt Power Supply
- Available in 40-Pin Plastic Dual-in-Line Package

The 8272 is an LSI Floppy Disk Controller (FDC) Chip, which contains the circuitry and control functions for interfacing a processor to 4 Floppy Disk Drives. It is capable of supporting either IBM 3740 single density format (FM), or IBM System 34 Double Density format (MFM) including double sided recording. The 8272 provides control signals which simplify the design of an external phase locked loop, and write precompensation circuitry. The FDC simplifies and handles most of the burdens associated with implementing a Floppy Disk Drive Interface.

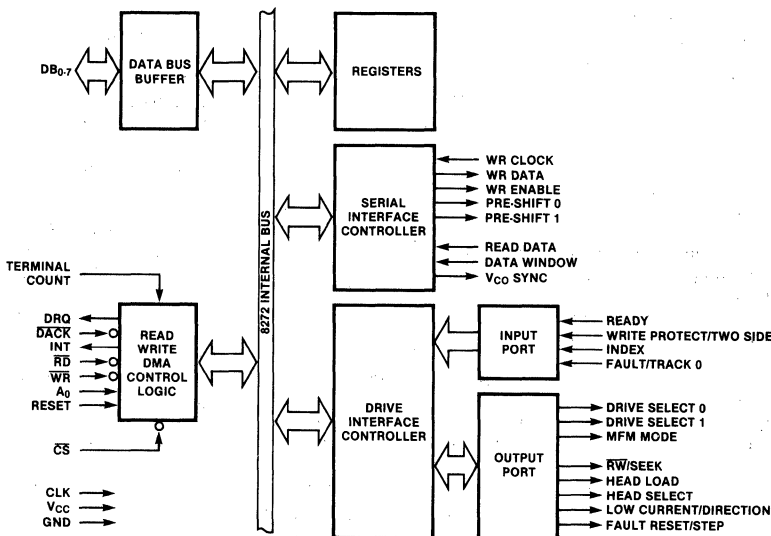


Figure 1. 8272 Internal Block Diagram

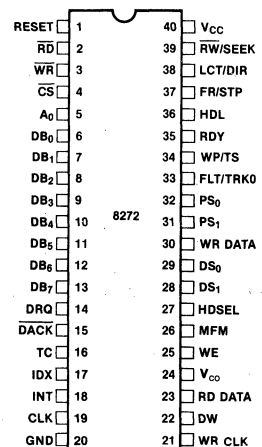


Figure 2. Pin Configuration

Table 1. Pin Description

Symbol	Pin No.	Type	Connection To	Name and Function
RST	1	I	$\mu$ P	<b>Reset:</b> Places FDC in idle state. Resets output lines to FDD to "0" (low).
$\overline{\text{RD}}$	2	I <sup>[1]</sup>	$\mu$ P	<b>Read:</b> Control signal for transfer of data from FDC to Data Bus, when "0" (low).
$\overline{\text{WR}}$	3	I <sup>[1]</sup>	$\mu$ P	<b>Write:</b> Control signal for transfer of data to FDC via Data Bus, when "0" (low).
$\overline{\text{CS}}$	4	I	$\mu$ P	<b>Chip Select:</b> IC selected when "0" (low), allowing $\overline{\text{RD}}$ and $\overline{\text{WR}}$ to be enabled.
A <sub>0</sub>	5	I <sup>[1]</sup>	$\mu$ P	<b>Data/Status Register Select:</b> Selects Data Reg (A <sub>0</sub> = 1) or Status Reg (A <sub>0</sub> = 0) content be sent to Data Bus.
DB <sub>0</sub> -DB <sub>7</sub>	6-13	I/O <sup>[1]</sup>	$\mu$ P	<b>Data Bus:</b> Bidirectional 8-Bit Data Bus.
DRQ	14	O	DMA	<b>Data DMA Request:</b> DMA Request is being made by FDC when DRQ "1."
$\overline{\text{DACK}}$	15	I	DMA	<b>DMA Acknowledge:</b> DMA cycle is active when "0" (low) and Controller is performing DMA transfer.
TC	16	I	DMA	<b>Terminal Count:</b> Indicates the termination of a DMA transfer when "1" (high) <sup>[2]</sup> .
IDX	17	I	FDD	<b>Index:</b> Indicates the beginning of a disk track.
INT	18	O	$\mu$ P	<b>Interrupt:</b> Interrupt Request Generated by FDC.
CLK	19	I		<b>Clock:</b> Single Phase 8 MHz Squarewave Clock.
GND	20			<b>Ground:</b> D.C. Power Return.

 Note 1: Disabled when  $\overline{\text{CS}}=1$ .

Note 2: TC must be activated to terminate the Execution Phase of any command.

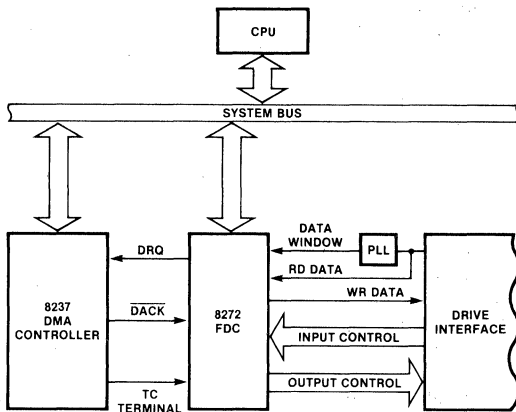
Symbol	Pin No.	Type	Connection To	Name and Function
V <sub>CC</sub>	40			<b>D.C. Power:</b> +5V
$\overline{\text{RW}}/\text{SEEK}$	39	O	FDD	<b>Read Write / SEEK:</b> When "1" (high) Seek mode selected and when "0" (low) Read/Write mode selected.
LCT/DIR	38	O	FDD	<b>Low Current/Direction:</b> Lowers Write current on inner tracks in Read/Write mode, determines direction head will step in Seek mode.
FR/STP	37	O	FDD	<b>Fault Reset/Step:</b> Resets fault FF in FDD in Read/Write mode, provides step pulses to move head to another cylinder in Seek mode.
HDL	36	O	FDD	<b>Head Load:</b> Command which causes read/write head in FDD to contact diskette.
RDY	35	I	FDD	<b>Ready:</b> Indicates FDD is ready to send or receive data.
WP/TS	34	I	FDD	<b>Write Protect / Two-Side:</b> Senses Write Protect status in Read/Write mode, and Two Side Media in Seek mode.
FLT/TRK0	33	I	FDD	<b>Fault/Track 0:</b> Senses FDD fault condition in Read/Write mode and Track 0 condition in Seek mode.
PS <sub>1</sub> ,PS <sub>0</sub>	31,32	O	FDD	<b>Precompensation (pre-shift):</b> Write precompensation status during MFM mode. Determines early, late, and normal times.
WR DATA	30	O	FDD	<b>Write Data:</b> Serial clock and data bits to FDD.
DS <sub>1</sub> ,DS <sub>0</sub>	28,29	O	FDD	<b>Drive Select:</b> Selects FDD unit.
HSEL	27	O	FDD	<b>Head Select:</b> Head 1 selected when "1" (high) Head 0 selected when "0" (low).

Table 1. Pin Description (Continued)

Symbol	Pin No.	Type	Connection To	Name and Function
MFM	26	O	PLL	<b>MFM Mode:</b> MFM mode when "1," FM mode when "0."
WE	25	O	FDD	<b>Write Enable:</b> Enables write data into FDD.
VCO	24	O	PLL	<b>VCO Sync:</b> Inhibits VCO in PLL when "0" (low), enables VCO when "1."
RD DATA	23	I	FDD	<b>Read Data:</b> Read data from FDD, containing clock and data bits.

Symbol	Pin No.	Type	Connection To	Name and Function
DW	22	I	PLL	<b>Data Window:</b> Generated by PLL, and used to sample data from FDD.
WR CLK	21	I		<b>Write Clock:</b> Write data rate to FDD FM = 500 kHz, MFM = 1 MHz, with a pulse width of 250 ns for both FM and MFM.  Must be enabled for all operations, both Read and Write.

8272 SYSTEM BLOCK DIAGRAM



bytes to fully specify the operation which the processor wishes the FDC to perform. The following commands are available.

- |                    |                                  |
|--------------------|----------------------------------|
| Read Data          | Write Data                       |
| Read ID            | Format a Track                   |
| Read Deleted Data  | Write Deleted Data               |
| Read a Track       | Seek                             |
| Scan Equal         | Recalibrate (Restore to Track 0) |
| Scan High or Equal | Sense Interrupt Status           |
| Scan Low or Equal  | Sense Drive Status               |
| Specify            |                                  |

FEATURES

Address mark detection circuitry is internal to the FDC which simplifies the phase locked loop and read electronics. The track stepping rate, head load time, and head unload time may be programmed by the user. The 8272 offers many additional features such as multiple sector transfers in both read and write modes with a single command, and full IBM compatibility in both single (FM) and double density (MFM) modes.

DESCRIPTION

Hand-shaking signals are provided in the 8272 which make DMA operation easy to incorporate with the aid of an external DMA Controller chip, such as the 8237. The FDC will operate in either DMA or Non-DMA mode. In the Non-DMA mode, the FDC generates interrupts to the processor for every transfer of a data byte between the CPU and the 8272. In the DMA mode, the processor need only load a command into the FDC and all data transfers occur under control of the 8272 and DMA controller.

There are 15 separate commands which the 8272 will execute. Each of these commands require multiple 8-bit

8272 REGISTERS — CPU INTERFACE

The 8272 contains two registers which may be accessed by the main system processor; a Status Register and a Data Register. The 8-bit Main Status Register contains the status information of the FDC, and may be accessed at any time. The 8-bit Data Register (actually consists of several registers in a stack with only one register presented to the data bus at a time), stores data, commands, parameters, and FDD status information. Data bytes are read out of, or written into, the Data Register in order to program or obtain the results after execution of a command. The Status Register may only be read and is used to facilitate the transfer of data between the processor and 8272.

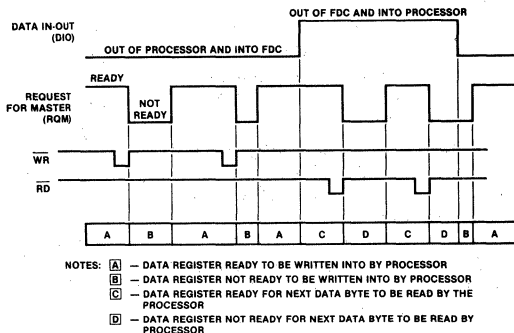
The relationship between the Status/Data registers and the signals RD, WR, and A<sub>0</sub> is shown below.

A <sub>0</sub>	RD	WR	FUNCTION
0	0	1	Read Main Status Register
0	1	0	Illegal
0	0	0	Illegal
1	0	0	Illegal
1	0	1	Read from Data Register
1	1	0	Write into Data Register

The bits in the Main Status Register are defined as follows:

BIT NUMBER	NAME	SYMBOL	DESCRIPTION
DB <sub>0</sub>	FDD 0 Busy	D <sub>0</sub> B	FDD number 0 is in the Seek mode.
DB <sub>1</sub>	FDD 1 Busy	D <sub>1</sub> B	FDD number 1 is in the Seek mode.
DB <sub>2</sub>	FDD 2 Busy	D <sub>2</sub> B	FDD number 2 is in the Seek mode.
DB <sub>3</sub>	FDD 3 Busy	D <sub>3</sub> B	FDD number 3 is in the Seek mode.
DB <sub>4</sub>	FDC Busy	CB	A read or write command is in process.
DB <sub>5</sub>	Non-DMA mode	NDM	The FDC is in the non-DMA mode. This bit is set only during the execution phase in non-DMA mode. Transition to "0" state indicates execution phase has ended.
DB <sub>6</sub>	Data Input/Output	DIO	Indicates direction of data transfer between FDC and Data Register. If DIO = "1" then transfer is from Data Register to the Processor. If DIO = "0", then transfer is from the Processor to Data Register.
DB <sub>7</sub>	Request for Master	RQM	Indicates Data Register is ready to send or receive data to or from the Processor. Both bits DIO and RQM should be used to perform the handshaking functions of "ready" and "direction" to the processor.

The DIO and RQM bits in the Status Register indicate when Data is ready and in which direction data will be transferred on the Data Bus.



STATUS REGISTER TIMING

The 8272 is capable of executing 15 different commands. Each command is initiated by a multi-byte transfer from the processor, and the result after execution of the command may also be a multi-byte transfer back to the processor. Because of this multi-byte interchange of information between the 8272 and the processor, it is convenient to consider each command as consisting of three phases:

**Command Phase:** The FDC receives all information required to perform a particular operation from the processor.

**Execution Phase:** The FDC performs the operation it was instructed to do.

**Result Phase:** After completion of the operation, status and other housekeeping information are made available to the processor.

During Command or Result Phases the Main Status Register (described earlier) must be read by the processor before each byte of information is written into or read from the Data Register. Bits D<sub>6</sub> and D<sub>7</sub> in the Main Status Register must be in a 0 and 1 state, respectively, before each byte of the command word may be written into the 8272. Many of the commands require multiple bytes, and as a result the Main Status Register must be read prior to each byte transfer to the 8272. On the other hand, during the Result Phase, D<sub>6</sub> and D<sub>7</sub> in the Main Status Register must both be 1's (D<sub>6</sub> = 1 and D<sub>7</sub> = 1) before reading each byte from the Data Register. Note, this reading of the Main Status Register before each byte transfer to the 8272 is required in only the Command and Result Phases, and NOT during the Execution Phase.

During the Execution Phase, the Main Status Register need not be read. If the 8272 is in the Non-DMA Mode, then the receipt of each data byte (if 8272 is reading data from FDD) is indicated by an Interrupt signal on pin 18 (INT = 1). The generation of a Read signal (RD = 0) will reset the Interrupt as well as output the Data onto the Data Bus. For example, if the processor cannot handle Interrupts fast enough (every 13 μs for MFM mode) then it may poll the Main Status Register and then bit D<sub>7</sub> (RQM) functions just like the Interrupt signal. If a Write

Command is in process then the WR signal performs the reset to the Interrupt signal.

If the 8272 is in the DMA Mode, no Interrupts are generated during the Execution Phase. The 8272 generates DRQ's (DMA Requests) when each byte of data is available. The DMA Controller responds to this request with both a DACK = 0 (DMA Acknowledge) and a RD = 0 (Read signal). When the DMA Acknowledge signal goes low (DACK = 0) then the DMA Request is reset (DRQ = 0). If a Write Command has been programmed then a WR signal will appear instead of RD. After the Execution Phase has been completed (Terminal Count has occurred) then an Interrupt will occur (INT = 1). This signifies the beginning of the Result Phase. When the first byte of data is read during the Result Phase, the Interrupt is automatically reset (INT = 0).

It is important to note that during the Result Phase all bytes shown in the Command Table must be read. The Read Data Command, for example, has seven bytes of

data in the Result Phase. All seven bytes must be read in order to successfully complete the Read Data Command. The 8272 will not accept a new command until all seven bytes have been read. Other commands may require fewer bytes to be read during the Result Phase.

The 8272 contains five Status Registers. The Main Status Register mentioned above may be read by the processor at any time. The other four Status Registers (ST0, ST1, ST2, and ST3) are only available during the Result Phase, and may be read only after successfully completing a command. The particular command which has been executed determines how many of the Status Registers will be read.

The bytes of data which are sent to the 8272 to form the Command Phase, and are read out of the 8272 in the Result Phase, must occur in the order shown in the Command Table. That is, the Command Code must be sent first and the other bytes sent in the prescribed sequence. No foreshortening of the Command or Result Phases are allowed. After the last byte of data in the Command Phase is sent to the 8272 the Execution Phase automatically starts. In a similar fashion, when

the last byte of data is read out in the Result Phase, the command is automatically ended and the 8272 is ready for a new command. A command may be aborted by simply sending a Terminal Count signal to pin 16 (TC = 1). This is a convenient means of ensuring that the processor may always get the 8272's attention even if the disk system hangs up in an abnormal manner.

**POLLING FEATURE OF THE 8272**

After the Specify command has been sent to the 8272, the Drive Select Lines DS0 and DS1 will automatically go into a polling mode. In between commands (and between step pulses in the SEEK command) the 8272 polls all four FDDs looking for a change in the Ready line from any of the drives. If the Ready line changes state (usually due to a door opening or closing) then the 8272 will generate an interrupt. When Status Register 0 (ST0) is read (after Sense Interrupt Status is issued), Not Ready (NR) will be indicated. The polling of the Ready line by the 8272 occurs continuously between instructions, thus notifying the processor which drives are on or off line.

Table 2. 8272 Command Set

PHASE	R/W	DATA BUS							REMARKS	PHASE	R/W	DATA BUS							REMARKS										
		D7	D6	D5	D4	D3	D2	D1				D0	D7	D6	D5	D4	D3	D2		D1	D0								
<b>READ DATA</b>															<b>WRITE DATA</b>														
Command	W	MT	MFM	SK	0	0	1	1	0	Command Codes	Command	W	MT	MFM	0	0	0	1	0	1	Command Codes								
	W	0	0	0	0	0	HDS	DS1	DS0	W		0	0	0	0	0	HDS	DS1	DS0	W									
	W								C	Sector ID information		W								C	Sector ID information								
	W								H	prior to Command		W								H	prior to Command								
	W								R	execution		W								R	execution								
	W								N			W								N									
	W								EOT			W								EOT									
Execution	W								GPL		W								GPL										
	W								DTL		W								DTL										
										Data transfer										Data transfer									
										between the FDD										between the main-									
									and main-system										system and FDD										
Result	R				ST 0						Status information	R				ST 0						Status information							
	R				ST 1						after Command	R				ST 1						after Command							
	R				ST 2						execution	R				ST 2						execution							
	R								C		R								C										
	R								H	Sector ID information	R								H	Sector ID information									
	R								R	after command	R								R	after Command									
<b>READ DELETED DATA</b>															<b>WRITE DELETED DATA</b>														
Command	W	MT	MFM	SK	0	1	1	0	0	Command Codes	Command	W	MT	MFM	0	0	1	0	0	1	Command Codes								
	W	0	0	0	0	0	HDS	DS1	DS0	W		0	0	0	0	0	HDS	DS1	DS0	W									
	W								C	Sector ID information		W								C	Sector ID information								
	W								H	prior to Command		W								H	prior to Command								
	W								R	execution		W								R	execution								
	W								N			W								N									
	W								EOT			W								EOT									
Execution	W								GPL		W								GPL										
	W								DTL		W								DTL										
										Data transfer										Data transfer									
										between the FDD										between the FDD									
									and main-system										and main-system										
Result	R				ST 0						Status information	R				ST 0						Status information							
	R				ST 1						after Command	R				ST 1						after Command							
	R				ST 2						execution	R				ST 2						execution							
	R								C		R								C										
	R								H	Sector ID information	R								H	Sector ID information									
	R								R	after Command	R								R	after Command									

Note: 1. Symbols used in this table are described at the end of this section.  
 2. A<sub>0</sub> = 1 for all operations.  
 3. X = Don't care, usually made to equal binary 0.



**Table 3. Command Mnemonics**

SYMBOL	NAME	DESCRIPTION
A <sub>0</sub>	Address Line 0	A <sub>0</sub> controls selection of Main Status Register (A <sub>0</sub> = 0) or Data Register (A <sub>0</sub> = 1).
C	Cylinder Number	C stands for the current selected Cylinder track number 0 through 76 of the medium.
D	Data	D stands for the data pattern which is going to be written into a Sector.
D <sub>7</sub> -D <sub>0</sub>	Data Bus	8-bit Data Bus where D <sub>7</sub> is the most significant bit, and D <sub>0</sub> is the least significant bit.
DS0, DS1	Drive Select	DS stands for a selected drive number 0 or 1.
DTL	Data Length	When N is defined as 00, DTL stands for the data length which users are going to read out or write into the Sector.
EOT	End of Track	EOT stands for the final Sector number of a Cylinder.
GPL	Gap Length	GPL stands for the length of Gap 3 (spacing between Sectors excluding VCO Sync Field).
H	Head Address	H stands for head number 0 or 1, as specified in ID field.
HDS	Head Select	HDS stands for a selected head number 0 or 1 (H = HDS in all command words).
HLT	Head Load Time	HLT stands for the head load time in the FDD (2 to 254 ms in 2 ms increments).
HUT	Head Unload Time	HUT stands for the head unload time after a read or write operation has occurred (16 to 240 ms in 16 ms increments).
MFM	FM or MFM Mode	If MF is low, FM mode is selected and if it is high, MFM mode is selected.
MT	Multi-Track	If MT is high, a multi-track operation is to be performed (a cylinder under both HD0 and HD1 will be read or written).
N	Number	N stands for the number of data bytes written in a Sector.

SYMBOL	NAME	DESCRIPTION
NCN	New Cylinder Number	NCN stands for a new Cylinder number, which is going to be reached as a result of the Seek operation. Desired position of Head.
ND	Non-DMA Mode	ND stands for operation in the Non-DMA Mode.
PCN	Present Cylinder Number	PCN stands for the Cylinder number at the completion of SENSE INTERRUPT STATUS Command. Position of Head at present time.
R	Record	R stands for the Sector number, which will be read or written.
R/W	Read/Write	R/W stands for either Read (R) or Write (W) signal.
SC	Sector	SC indicates the number of Sectors per Cylinder.
SK	Skip	SK stands for Skip Deleted Data Address Mark.
SRT	Step Rate Time	SRT stands for the Stepping Rate for the FDD (1 to 16 ms in 1 ms increments). The same Stepping Rate applies to all drives (F=1 ms, E=2 ms, etc.).
ST 0 ST 1 ST 2 ST 3	Status 0 Status 1 Status 2 Status 3	ST 0-3 stand for one of four registers which store the status information after a command has been executed. This information is available during the result phase after command execution. These registers should not be confused with the main status register (selected by A <sub>0</sub> = 0). ST 0-3 may be read only after a command has been executed and contain information relevant to that particular command.
STP		During a Scan operation, if STP = 1, the data in contiguous sectors is compared byte by byte with data sent from the processor (or DMA), and if STP = 2, then alternate sectors are read and compared.

## COMMAND DESCRIPTIONS

During the Command Phase, the Main Status Register must be polled by the CPU before each byte is written into the Data Register. The DIO (DB6) and RQM (DB7) bits in the Main Status Register must be in the "0" and "1" states respectively, before each byte of the command may be written into the 8272. The beginning of the execution phase for any of these commands will cause DIO and RQM to switch to "1" and "0" states respectively.

### READ DATA

A set of nine (9) byte words are required to place the FDC into the Read Data Mode. After the Read Data command has been issued the FDC loads the head (if it is in the unloaded state), waits the specified head settling time (defined in the Specify Command), and begins reading ID Address Marks and ID fields. When the current sector number ("R") stored in the ID Register (IDR)

compares with the sector number read off the diskette, then the FDC outputs data (from the data field) byte-by-byte to the main system via the data bus.

After completion of the read operation from the current sector, the Sector Number is incremented by one, and the data from the next sector is read and output on the data bus. This continuous read function is called a "Multi-Sector Read Operation." The Read Data Command may be terminated by the receipt of a Terminal Count signal. Upon receipt of this signal, the FDC stops outputting data to the processor, but will continue to read data from the current sector, check CRC (Cyclic Redundancy Count) bytes, and then at the end of the sector terminate the Read Data Command.

The amount of data which can be handled with a single command to the FDC depends upon MT (multi-track), MFM (MFM/FM), and N (Number of Bytes/Sector). Table 4 below shows the Transfer Capacity.

**Table 4. Transfer Capacity**

Multi-Track MT	MFM/FM MFM	Bytes/Sector N	Maximum Transfer Capacity (Bytes/Sector) (Number of Sectors)	Final Sector Read from Diskette
0	0	00	(128) (26) = 3,328	26 at Side 0
0	1	01	(256) (26) = 6,656	or 26 at Side 1
1	0	00	(128) (52) = 6,656	26 at Side 1
1	1	01	(256) (52) = 13,312	
0	0	01	(256) (15) = 3,840	15 at Side 0
0	1	02	(512) (15) = 7,680	or 15 at Side 1
1	0	01	(256) (30) = 7,680	15 at Side 1
1	1	02	(512) (30) = 15,360	
0	0	02	(512) (8) = 4,096	8 at Side 0
0	1	03	(1024) (8) = 8,192	or 8 at Side 1
1	0	02	(512) (16) = 8,192	8 at Side 1
1	1	03	(1024) (16) = 16,384	



The "multi-track" function (MT) allows the FDC to read data from both sides of the diskette. For a particular cylinder, data will be transferred starting at Sector 1, Side 0 and completing at Sector L, Side 1 (Sector L = last sector on the side). Note, this function pertains to only one cylinder (the same track) on each side of the diskette.

When N = 0, then DTL defines the data length which the FDC must treat as a sector. If DTL is smaller than the actual data length in a Sector, the data beyond DTL in the Sector, is not sent to the Data Bus. The FDC reads (internally) the complete Sector performing the CRC check, and depending upon the manner of command termination, may perform a Multi-Sector Read Operation. When N is non-zero, then DTL has no meaning and should be set to 0FFF.

At the completion of the Read Data Command, the head is not unloaded until after Head Unload Time Interval (specified in the Specify Command) has elapsed. If the processor issues another command before the head unloads then the head settling time may be saved between subsequent reads. This time out is particularly valuable when a diskette is copied from one drive to another.

If the FDC detects the Index Hole twice without finding the right sector, (indicated in "R"), then the FDC sets the ND (No Data) flag in Status Register 1 to a 1 (high), and terminates the Read Data Command. (Status Register 0 also has bits 7 and 6 set to 0 and 1 respectively.)

After reading the ID and Data Fields in each sector, the FDC checks the CRC bytes. If a read error is detected (incorrect CRC in ID field), the FDC sets the DE (Data Error) flag in Status Register 1 to a 1 (high), and if a CRC error occurs in the Data Field the FDC also sets the DD (Data Error in Data Field) flag in Status Register 2 to a 1 (high), and terminates the Read Data Command. (Status Register 0 also has bits 7 and 6 set to 0 and 1 respectively.)

If the FDC reads a Deleted Data Address Mark off the diskette, and the SK bit (bit D5 in the first Command Word) is not set (SK = 0), then the FDC sets the CM (Control Mark) flag in Status Register 2 to a 1 (high), and terminates the Read Data Command, after reading all the data in the Sector. If SK = 1, the FDC skips the sector with the Deleted Data Address Mark and reads the next sector.

During disk data transfers between the FDC and the processor, via the data bus, the FDC must be serviced by the processor every 27  $\mu$ s in the FM Mode, and every 13  $\mu$ s in the MFM Mode, or the FDC sets the OR (Over Run) flag in Status Register 1 to a 1 (high), and terminates the Read Data Command.

If the processor terminates a read (or write) operation in the FDC, then the ID Information in the Result Phase is dependent upon the state of the MT bit and EOT byte. Table 5 shows the values for C, H, R, and N, when the processor terminates the Command.

**Table 5. ID Information When Processor Terminates Command**

MT	EOT	Final Sector Transferred to Processor	ID Information at Result Phase			
			C	H	R	N
0	1A 0F 0B	Sector 1 to 25 at Side 0 Sector 1 to 14 at Side 0 Sector 1 to 7 at Side 0	NC	NC	R+1	NC
	1A 0F 0B	Sector 26 at Side 0 Sector 15 at Side 0 Sector 8 at Side 0	C+1	NC	R=01	NC
	1A 0F 0B	Sector 1 to 25 at Side 1 Sector 1 to 14 at Side 1 Sector 1 to 7 at Side 1	NC	NC	R+1	NC
	1A 0F 0B	Sector 26 at Side 1 Sector 15 at Side 1 Sector 8 at Side 1	C+1	NC	R=01	NC
1	1A 0F 0B	Sector 1 to 25 at Side 0 Sector 1 to 14 at Side 0 Sector 1 to 7 at Side 0	NC	NC	R+1	NC
	1A 0F 0B	Sector 26 at Side 0 Sector 15 at Side 0 Sector 8 at Side 0	NC	LSB	R=01	NC
	1A 0F 0B	Sector 1 to 25 at Side 1 Sector 1 to 14 at Side 1 Sector 1 to 7 at Side 1	NC	NC	R+1	NC
	1A 0F 0B	Sector 26 at Side 1 Sector 15 at Side 1 Sector 8 at Side 1	C+1	LSB	R=01	NC

Notes: 1. NC (No Change): The same value as the one at the beginning of command execution.

2. LSB (Least Significant Bit): The least significant bit of H is complemented.

**WRITE DATA**

A set of nine (9) bytes are required to set the FDC into the Write Data mode. After the Write Data command has been issued the FDC loads the head (if it is in the unloaded state), waits the specified head settling time (defined in the Specify Command), and begins reading ID Fields. When the current sector number ("R"), stored in the ID Register (IDR) compares with the sector number read off the diskette, then the FDC takes data from the processor byte-by-byte via the data bus, and outputs it to the FDD.

After writing data into the current sector, the Sector Number stored in "R" is incremented by one, and the next data field is written into. The FDC continues this "Multi-Sector Write Operation" until the issuance of a Terminal Count signal. If a Terminal Count signal is sent to the FDC it continues writing into the current sector to complete the data field. If the Terminal Count signal is received while a data field is being written then the remainder of the data field is filled with 00 (zeros).

The FDC reads the ID field of each sector and checks the CRC bytes. If the FDC detects a read error (incorrect CRC) in one of the ID Fields, it sets the DE (Data Error) flag of Status Register 1 to a 1 (high), and terminates the Write Data Command. (Status Register 0 also has bits 7 and 6 set to 0 and 1 respectively.)

The Write Command operates in much the same manner as the Read Command. The following items are the same; refer to the Read Data Command for details:

- Transfer Capacity
- EN (End of Cylinder) Flag
- ND (No Data) Flag

- Head Unload Time Interval
- ID Information when the processor terminates command (see Table 2)
- Definition of DTL when N = 0 and when N ≠ 0

In the Write Data mode, data transfers between the processor and FDC must occur every 31 μs in the FM mode, and every 15 μs in the MFM mode. If the time interval between data transfers is longer than this then the FDC sets the OR (Over Run) flag in Status Register 1 to a 1 (high), and terminates the Write Data Command.

**WRITE DELETED DATA**

This command is the same as the Write Data Command except a Deleted Data Address Mark is written at the beginning of the Data Field instead of the normal Data Address Mark.

**READ DELETED DATA**

This command is the same as the Read Data Command except that when the FDC detects a Data Address Mark at the beginning of a Data Field (and SK = 0 (low)), it will read all the data in the sector and set the CM flag in Status Register 2 to a 1 (high), and then terminate the command. If SK = 1, then the FDC skips the sector with the Data Address Mark and reads the next sector.

**READ A TRACK**

This command is similar to READ DATA Command except that the entire data field is read continuously from each of the sectors of a track. Immediately after encountering the INDEX HOLE, the FDC starts reading all data fields on the track as continuous blocks of data. If the FDC finds an error in the ID or DATA CRC check bytes, it continues to read data from the track. The FDC compares the ID information read from each sector with the value stored in the IDR, and sets the ND flag of Status Register 1 to a 1 (high) if there is no comparison. Multi-track or skip operations are not allowed with this command.

This command terminates when EOT number of sectors have been read. If the FDC does not find an ID Address Mark on the diskette after it encounters the INDEX HOLE for the second time, then it sets the MA (missing address mark) flag in Status Register 1 to a 1 (high), and terminates the command. (Status Register 0 has bits 7 and 6 set to 0 and 1 respectively.)

**READ ID**

The READ ID Command is used to give the present position of the recording head. The FDC stores the values from the first ID Field it is able to read. If no proper ID Address Mark is found on the diskette, before the INDEX HOLE is encountered for the second time then the MA (Missing Address Mark) flag in Status Register 1 is set to a 1 (high), and if no data is found then the ND (No Data) flag is also set in Status Register 1 to a 1 (high) and the command is terminated.

**FORMAT A TRACK**

The Format Command allows an entire track to be formatted. After the INDEX HOLE is detected, Data is written on the Diskette: Gaps, Address Marks, ID Fields and Data Fields, all per the IBM System 34 (Double Density) or System 3740 (Single Density) Format are recorded. The particular format which will be written is controlled by the values programmed into N (number of bytes/sector), SC (sectors/cylinder), GPL (Gap Length), and D (Data Pattern) which are supplied by the processor during the Command Phase. The Data Field is filled with the Byte of data stored in D. The ID Field for each sector is supplied by the processor; that is, four data requests per sector are made by the FDC for C (Cylinder Number), H (Head Number), R (Sector Number) and N (Number of Bytes/Sector). This allows the diskette to be formatted with nonsequential sector numbers, if desired.

After formatting each sector, the processor must send new values for C, H, R, and N to the 8272 for each sector on the track. The contents of the R register is incremented by one after each sector is formatted, thus, the R register contains a value of R + 1 when it is read during the Result Phase. This incrementing and formatting continues for the whole track until the FDC encounters the INDEX HOLE for the second time, whereupon it terminates the command.

If a FAULT signal is received from the FDD at the end of a write operation, then the FDC sets the EC flag of Status Register 0 to a 1 (high), and terminates the command after setting bits 7 and 6 of Status Register 0 to 0 and 1 respectively. Also the loss of a READY signal at the beginning of a command execution phase causes command termination.

Table 6 shows the relationship between N, SC, and GPL for various sector sizes:

**Table 6. Sector Size Relationships**

FORMAT	SECTOR SIZE	N	SC	GPL <sup>1</sup>	GPL <sup>2</sup>	REMARKS
FM Mode	128 bytes/Sector	00	1A <sub>(16)</sub>	07 <sub>(16)</sub>	1B <sub>(16)</sub>	IBM Diskette 1
	256	01	0F <sub>(16)</sub>	0E <sub>(16)</sub>	2A <sub>(16)</sub>	
	512	02	08	1B <sub>(16)</sub>	3A <sub>(16)</sub>	
FM Mode	1024 bytes/Sector	03	04	—	—	
	2048	04	02	—	—	
	4096	05	01	—	—	
MFM Mode	256	01	1A <sub>(16)</sub>	0E <sub>(16)</sub>	36 <sub>(16)</sub>	IBM Diskette 2D
	512	02	0F <sub>(16)</sub>	1B <sub>(16)</sub>	54 <sub>(16)</sub>	
	1024	03	08	35 <sub>(16)</sub>	74 <sub>(16)</sub>	
	2048	04	04	—	—	
	4096	05	02	—	—	
	8192	06	01	—	—	

Note: 1. Suggested values of GPL in Read or Write Commands to avoid splice point between data field and ID field of contiguous sections.  
 2. Suggested values of GPL in format command.

**SCAN COMMANDS**

The SCAN Commands allow data which is being read from the diskette to be compared against data which is being supplied from the main system (Processor in NON-DMA mode, and DMA Controller in DMA mode). The FDC compares the data on a byte-by-byte basis, and looks for a sector of data which meets the conditions of  $D_{FDD} = D_{Processor}$ ,  $D_{FDD} \leq D_{Processor}$ , or  $D_{FDD} \geq D_{Processor}$ . Ones complement arithmetic is used for comparison (FF = largest number, 00 = smallest number). After a whole sector of data is compared, if the conditions are not met, the sector number is incremented ( $R + STP \rightarrow R$ ), and the scan operation is continued. The scan operation continues until one of the following conditions occur; the conditions for scan are met (equal, low, or high), the last sector on the track is reached (EOT), or the terminal count signal is received.

If the conditions for scan are met then the FDC sets the SH (Scan Hit) flag of Status Register 2 to a 1 (high), and terminates the Scan Command. If the conditions for scan are not met between the starting sector (as specified by R) and the last sector on the cylinder (EOT), then the FDC sets the SN (Scan Not Satisfied) flag of Status Register 2 to a 1 (high), and terminates the Scan Command. The receipt of a TERMINAL COUNT signal from the Processor or DMA Controller during the scan operation will cause the FDC to complete the comparison of the particular byte which is in process, and then to terminate the command. Table 7 shows the status of bits SH and SN under various conditions of SCAN.

**Table 7. Scan Status Codes**

COMMAND	STATUS REGISTER 2		COMMENTS
	BIT 2 = SN	BIT 3 = SH	
Scan Equal	0	1	$D_{FDD} = D_{Processor}$ $D_{FDD} \neq D_{Processor}$
	1	0	
Scan Low or Equal	0	1	$D_{FDD} = D_{Processor}$ $D_{FDD} < D_{Processor}$ $D_{FDD} \neq D_{Processor}$
	0	0	
	1	0	
Scan High or Equal	0	1	$D_{FDD} = D_{Processor}$ $D_{FDD} > D_{Processor}$ $D_{FDD} \neq D_{Processor}$
	0	0	
	1	0	

If the FDC encounters a Deleted Data Address Mark on one of the sectors (and SK = 0), then it regards the sector as the last sector on the cylinder, sets CM (Control Mark) flag of Status Register 2 to a 1 (high) and terminates the command. If SK = 1, the FDC skips the sector with the Deleted Address Mark, and reads the next sector. In the second case (SK = 1), the FDC sets the CM (Control Mark) flag of Status Register 2 to a 1 (high) in order to show that a Deleted Sector had been encountered.

When either the STP (contiguous sectors STP = 01, or alternate sectors STP = 02 sectors are read) or the MT (Multi-Track) are programmed, it is necessary to remember that the last sector on the track must be read. For example, if STP = 02, MT = 0, the sectors are numbered sequentially 1 through 26, and we start the Scan Command at sector 21; the following will happen. Sectors 21, 23, and 25 will be read, then the next sector (26) will be skipped and the Index Hole will be encountered before the EOT value of 26 can be read. This will result in an abnormal termination of the command. If the EOT had been set at 25 or the scanning started at sector 20, then the Scan Command would be completed in a normal manner.

During the Scan Command data is supplied by either the processor or DMA Controller for comparison against the data read from the diskette. In order to avoid having the OR (Over Run) flag set in Status Register 1, it is necessary to have the data available in less than 27  $\mu$ s (FM Mode) or 13  $\mu$ s (MFM Mode). If an Overrun occurs the FDC terminates the command.

**SEEK**

The read/write head within the FDD is moved from cylinder to cylinder under control of the Seek Command. The FDC compares the PCN (Present Cylinder Number) which is the current head position with the NCN (New Cylinder Number), and performs the following operation if there is a difference:

PCN < NCN: Direction signal to FDD set to a 1 (high), and Step Pulses are issued. (Step In.)

PCN > NCN: Direction signal to FDD set to a 0 (low), and Step Pulses are issued. (Step Out.)

The rate at which Step Pulses are issued is controlled by SRT (Stepping Rate Time) in the SPECIFY Command. After each Step Pulse is issued NCN is compared against PCN, and when NCN = PCN, then the SE (Seek End) flag is set in Status Register 0 to a 1 (high), and the command is terminated.

During the Command Phase of the Seek operation the FDC is in the FDC BUSY state, but during the Execution Phase it is in the NON BUSY state. While the FDC is in the NON BUSY state, another Seek Command may be issued, and in this manner parallel seek operations may be done on up to 4 Drives at once.

If an FDD is in a NOT READY state at the beginning of the command execution phase or during the seek operation, then the NR (NOT READY) flag is set in Status Register 0 to a 1 (high), and the command is terminated.

**RECALIBRATE**

This command causes the read/write head within the FDD to retract to the Track 0 position. The FDC clears the contents of the PCN counter, and checks the status of the Track 0 signal from the FDD. As long as the Track 0 signal is low, the Direction signal remains 1 (high) and Step Pulses are issued. When the Track 0 signal goes high, the SE (SEEK END) flag in Status Register 0 is set to a 1 (high) and the command is terminated. If the Track 0 signal is still low after 77 Step Pulses have been issued, the FDC sets the SE (SEEK END) and EC (EQUIPMENT CHECK) flags of Status Register 0 to both 1s (highs), and terminates the command.

The ability to overlap RECALIBRATE Commands to multiple FDDs, and the loss of the READY signal, as described in the SEEK Command, also applies to the RECALIBRATE Command.

**SENSE INTERRUPT STATUS**

An Interrupt signal is generated by the FDC for one of the following reasons:

1. Upon entering the Result Phase of:
  - a. Read Data Command
  - b. Read a Track Command
  - c. Read ID Command
  - d. Read Deleted Data Command
  - e. Write Data Command
  - f. Format a Cylinder Command
  - g. Write Deleted Data Command
  - h. Scan Commands
2. Ready Line of FDD changes state
3. End of Seek or Recalibrate Command
4. During Execution Phase in the NON-DMA Mode

Interrupts caused by reasons 1 and 4 above occur during normal command operations and are easily discernible by the processor. However, interrupts caused by reasons 2 and 3 above may be uniquely identified with the aid of the Sense Interrupt Status Command. This command when issued resets the interrupt signal and via bits 5, 6, and 7 of Status Register 0 identifies the cause of the interrupt.

**Table 8. Seek, Interrupt Codes**

SEEK END BIT 5	INTERRUPT CODE		CAUSE
	BIT 6	BIT 7	
0	1	1	Ready Line changed state, either polarity
1	0	0	Normal Termination of Seek or Recalibrate Command
1	1	0	Abnormal Termination of Seek or Recalibrate Command

Neither the Seek or Recalibrate Command have a Result Phase. Therefore, it is mandatory to use the Sense Interrupt Status Command after these commands to effectively terminate them and to provide verification of the head position (PCN).

**SPECIFY**

The Specify Command sets the initial values for each of the three internal timers. The HUT (Head Unload Time) defines the time from the end of the Execution Phase of one of the Read/Write Commands to the head unload state. This timer is programmable from 16 to 240 ms in increments of 16 ms (01 = 16 ms, 02 = 32 ms . . . OF = 240 ms). The SRT (Step Rate Time) defines the time interval between adjacent step pulses. This timer is programmable from 1 to 16 ms in increments of 1 ms (F = 1 ms, E = 2 ms, D = 3 ms, etc.). The HLT (Head Load Time) defines the time between when the Head Load signal goes high and when the Read/Write operation starts. This timer is programmable from 2 to 254 ms in increments of 2 ms (01 = 2 ms, 02 = 4 ms, 03 = 6 ms . . . FE = 254 ms).

The time intervals mentioned above are a direct function of the clock (CLK on pin 19). Times indicated above are for an 8 MHz clock, if the clock was reduced to 4 MHz (mini-floppy application) then all time intervals are increased by a factor of 2.

The choice of DMA or NON-DMA operation is made by the ND (NON-DMA) bit. When this bit is high (ND = 1) the NON-DMA mode is selected, and when ND = 0 the DMA mode is selected.

**SENSE DRIVE STATUS**

This command may be used by the processor whenever it wishes to obtain the status of the FDDs. Status Register 3 contains the Drive Status information.

**INVALID**

If an invalid command is sent to the FDC (a command not defined above), then the FDC will terminate the command. No interrupt is generated by the 8272 during this condition. Bit 6 and bit 7 (DIO and RQM) in the Main Status Register are both high ("1") indicating to the processor that the 8272 is in the Result Phase and the contents of Status Register 0 (STO) must be read. When the processor reads Status Register 0 it will find a 80H indicating an invalid command was received.

A Sense Interrupt Status Command must be sent after a Seek or Recalibrate interrupt, otherwise the FDC will consider the next command to be an Invalid Command.

In some applications the user may wish to use this command as a No-Op command, to place the FDC in a stand-by or no operation state.

Table 9. Status Registers

BIT			DESCRIPTION
NO.	NAME	SYMBOL	
<b>STATUS REGISTER 0</b>			
D <sub>7</sub>	Interrupt Code	IC	D <sub>7</sub> = 0 and D <sub>6</sub> = 0 Normal Termination of Command, (NT). Command was completed and properly executed.
D <sub>6</sub>			D <sub>7</sub> = 0 and D <sub>6</sub> = 1 Abnormal Termination of Command, (AT). Execution of Command was started, but was not successfully completed.
			D <sub>7</sub> = 1 and D <sub>6</sub> = 0 Invalid Command Issue, (IC). Command which was issued was never started.
			D <sub>7</sub> = 1 and D <sub>6</sub> = 1 Abnormal Termination because during command execution the ready signal from FDD changed state.
D <sub>5</sub>	Seek End	SE	When the FDC completes the SEEK Command, this flag is set to 1 (high).
D <sub>4</sub>	Equipment Check	EC	If a fault Signal is received from the FDD, or if the Track 0 Signal fails to occur after 77 Step Pulses (Recalibrate Command) then this flag is set.
D <sub>3</sub>	Not Ready	NR	When the FDD is in the not-ready state and a read or write command is issued, this flag is set. If a read or write command is issued to Side 1 of a single sided drive, then this flag is set.
D <sub>2</sub>	Head Address	HD	This flag is used to indicate the state of the head at interrupt.
D <sub>1</sub>	Unit Select 1	US 1	These flags are used to indicate a Drive Unit Number at Interrupt
D <sub>0</sub>	Unit Select 0	US 0	
<b>STATUS REGISTER 1</b>			
D <sub>7</sub>	End of Cylinder	EN	When the FDC tries to access a Sector beyond the final Sector of a Cylinder, this flag is set.
D <sub>6</sub>			Not used. This bit is always 0 (low).
D <sub>5</sub>	Data Error	DE	When the FDC detects a CRC error in either the ID field or the data field, this flag is set.
D <sub>4</sub>	Over Run	OR	If the FDC is not serviced by the main-systems during data transfers, within a certain time interval, this flag is set.
D <sub>3</sub>			Not used. This bit always 0 (low).
D <sub>2</sub>	No Data	ND	During execution of READ DATA, WRITE DELETED DATA or SCAN Command, if the FDC cannot find the Sector specified in the IDR Register, this flag is set.
			During executing the READ ID Command, if the FDC cannot read the ID field without an error, then this flag is set.
			During the execution of the READ A Cylinder Command, if the starting sector cannot be found, then this flag is set.

BIT			DESCRIPTION
NO.	NAME	SYMBOL	
<b>STATUS REGISTER 1 (CONT.)</b>			
D <sub>1</sub>	Not Writable	NW	During execution of WRITE DATA, WRITE DELETED DATA or Format A Cylinder Command, if the FDC detects a write protect signal from the FDD, then this flag is set.
D <sub>0</sub>	Missing Address Mark	MA	If the FDC cannot detect the ID Address Mark after encountering the index hole twice, then this flag is set.
			If the FDC cannot detect the Data Address Mark or Deleted Data Address Mark, this flag is set. Also at the same time, the MD (Missing Address Mark in Data Field) of Status Register 2 is set.
<b>STATUS REGISTER 2</b>			
D <sub>7</sub>			Not used. This bit is always 0 (low).
D <sub>6</sub>	Control Mark	CM	During executing the READ DATA or SCAN Command, if the FDC encounters a Sector which contains a Deleted Data Address Mark, this flag is set.
D <sub>5</sub>	Data Error in Data Field	DD	If the FDC detects a CRC error in the data field then this flag is set.
D <sub>4</sub>	Wrong Cylinder	WC	This bit is related with the ND bit, and when the contents of C on the medium is different from that stored in the IDR, this flag is set.
D <sub>3</sub>	Scan Equal Hit	SH	During execution, the SCAN Command, if the condition of "equal" is satisfied, this flag is set.
D <sub>2</sub>	Scan Not Satisfied	SN	During executing the SCAN Command, if the FDC cannot find a Sector on the cylinder which meets the condition, then this flag is set.
D <sub>1</sub>	Bad Cylinder	BC	This bit is related with the ND bit, and when the content of C on the medium is different from that stored in the IDR and the content of C is FF, then this flag is set.
D <sub>0</sub>	Missing Address Mark in Data Field	MD	When data is read from the medium, if the FDC cannot find a Data Address Mark or Deleted Data Address Mark, then this flag is set.
<b>STATUS REGISTER 3</b>			
D <sub>7</sub>	Fault	FT	This bit is used to indicate the status of the Fault signal from the FDD.
D <sub>6</sub>	Write Protected	WP	This bit is used to indicate the status of the Write Protected signal from the FDD.
D <sub>5</sub>	Ready	RDY	This bit is used to indicate the status of the Ready signal from the FDD.
D <sub>4</sub>	Track 0	T0	This bit is used to indicate the status of the Track 0 signal from the FDD.
D <sub>3</sub>	Two Side	TS	This bit is used to indicate the status of the Two Side signal from the FDD.
D <sub>2</sub>	Head Address	HD	This bit is used to indicate the status of Side Select signal to the FDD.
D <sub>1</sub>	Unit Select 1	US 1	This bit is used to indicate the status of the Unit Select 1 signal to the FDD.
D <sub>0</sub>	Unit Select 0	US 0	This bit is used to indicate the status of the Unit Select 0 signal to the FDD.

**ABSOLUTE MAXIMUM RATINGS\***

Operating Temperature .....	-10°C to +70°C
Storage Temperature .....	-40°C to +125°C
All Output Voltages .....	-0.5 to +7 Volts
All Input Voltages .....	-0.5 to +7 Volts
Supply Voltage $V_{CC}$ .....	-0.5 to +7 Volts
Power Dissipation .....	1 Watt

 \* $T_A = 25^\circ\text{C}$ 

*NOTICE: Stress above those listed under "Absolute Maximum Ratings" may cause permanent damage to the device. This is a stress rating only and functional operation of the device at these or any other conditions above those indicated in the operational sections of this specification is not implied. Exposure to absolute maximum rating conditions for extended periods may affect device reliability.*

**D.C. CHARACTERISTICS** ( $T_A = 0^\circ\text{C}$  to  $+70^\circ\text{C}$ ,  $V_{CC} = +5\text{V} \pm 5\%$ )

Symbol	Parameter	Limits		Unit	Test Conditions
		Min.	Max.		
$V_{IL1}$	Input Low Voltage	-0.5	0.8	V	
$V_{IH1}$	Input High Voltage	2.0	$V_{CC} + 0.5$	V	
$V_{IL2}$	(CLK & WR CLK)	-0.5	0.65	V	
$V_{IH2}$	(CLK & WR CLK)	2.4	$V_{CC} + 0.5$	V	
$V_{OL}$	Output Low Voltage		0.45	V	$I_{OL} = 2.0\text{ mA}$
$V_{OH}$	Output High Voltage	2.4	$V_{CC}$	V	$I_{OH} = -200\ \mu\text{A}$
$I_{CC}$	$V_{CC}$ Supply Current		150	mA	
$I_{IL}$	Input Load Current (All Input Pins)		10 -10	$\mu\text{A}$ $\mu\text{A}$	$V_{IN} = V_{CC}$ $V_{IN} = 0\text{V}$
$I_{LOH}$	High Level Output Leakage Current		10	$\mu\text{A}$	$V_{OUT} = V_{CC}$
$I_{LOL}$	Low Level Output Leakage Current		-10	$\mu\text{A}$	$V_{OUT} = +0.45\text{V}$

**CAPACITANCE** ( $T_A = 25^\circ\text{C}$ ,  $f_c = 1\text{ MHz}$ ,  $V_{CC} = 0\text{V}$ )

Symbol	Parameter	Limits		Unit	Test Conditions
		Min.	Max.		
$C_{IN(\phi)}$	Clock Input Capacitance		20	pF	All Pins Except Pin Under Test Tied to AC Ground
$C_{IN}$	Input Capacitance		10	pF	
$C_{OUT}$	Output Capacitance		20	pF	

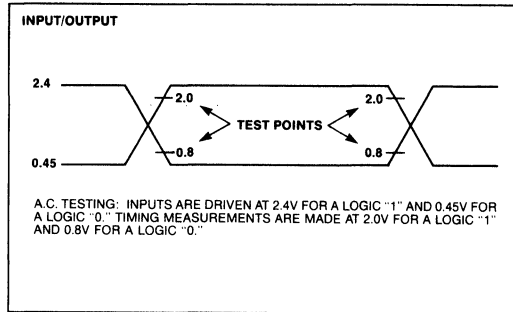
**A.C. CHARACTERISTICS** ( $T_A = 0^\circ\text{C}$  to  $70^\circ\text{C}$ ;  $V_{CC} = +5.0\text{V} \pm 5\%$ )

Symbol	Parameter	Min.	Max.	Unit	Test Conditions
$t_{CY}$	Clock Period	125		ns	Note 4
$t_{CH}$	Clock High Period	40		ns	
$t_{RST}$	Reset Width	14		$t_{CY}$	
<b>Read Cycle</b>					
$t_{AR}$	Select Setup to $\overline{RD}$	0		ns	
$t_{RA}$	Select Hold from $\overline{RD}$	0		ns	
$t_{RR}$	$\overline{RD}$ Pulse Width	250		ns	
$t_{RD}$	Data Delay from $\overline{RD}$		200	ns	
$t_{DF}$	Output Float Delay	20	100	ns	
<b>Write Cycle</b>					
$t_{AW}$	Select Setup to $\overline{WR}$	0		ns	
$t_{WA}$	Select Hold from $\overline{WR}$	0		ns	
$t_{WW}$	$\overline{WR}$ Pulse Width	250		ns	
$t_{DW}$	Data Setup to $\overline{WR}$	150		ns	
$t_{WD}$	Data Hold from $\overline{WR}$	5		ns	
<b>Interrupts</b>					
$t_{RI}$	INT Delay from $\overline{RD}$		500	ns	
$t_{WI}$	INT Delay from $\overline{WR}$		500	ns	
<b>DMA</b>					
$t_{RQCY}$	DRQ Cycle Period	13		$\mu\text{s}$	8 MHz clock 8 MHz clock 8 MHz clock
$t_{AKRQ}$	$\overline{DACK}$ to DRQ		200	ns	
$t_{RQR}$	DRQ to $\overline{RD}$	800		ns	
$t_{RQW}$	DRQ to $\overline{WR}$	250		ns	
$t_{RQRW}$	DRQ to $\overline{RD}$ or $\overline{WR}$		12	$\mu\text{s}$	
<b>FDD Interface</b>					
$t_{WCY}$	WCK Cycle Time	TYP 1			
		2 or 4		$\mu\text{s}$	MFM = 0
$t_{WCH}$	WCK High Time	1 or 2		$\mu\text{s}$	MFM = 1
		250	100	350	ns
$t_{CP}$	Pre-Shift Delay from WCK		20	100	ns
$t_{CD}$	WDA Delay from WCK		20	100	ns
$t_{WDD}$	Write Data Width		$t_{WCH} - 50$		ns
$t_{WE}$	WE to WCK or WE to WCK Delay		20		ns
$t_{WWCY}$	Window Cycle Time	2			MFM = 0
		1			MFM = 1
$t_{WRD}$	Window Setup to RDD		15		ns
$t_{RDW}$	Window Hold from RDD		15		ns
$t_{RDD}$	RDD Active Time (HIGH)		40		ns
<b>FDD SEEK/DIRECTION/STEP</b>					
$t_{US}$	$US_{0,1}$ Setup to $\overline{RW/SEEK}$		12		$\mu\text{s}$
$t_{SD}$	$\overline{RW/SEEK}$ Setup to LCT/DIR		6.8		$\mu\text{s}$
$t_{DS}$	$\overline{RW/SEEK}$ Hold from LCT/DIR		30		$\mu\text{s}$
$t_{DST}$	LCT/DIR Setup to FR/STEP		1		$\mu\text{s}$
$t_{STD}$	LCT/DIR Hold from FR/STEP		24		$\mu\text{s}$
$t_{STU}$	$DS_{0,1}$ Hold from FR/STEP		5		$\mu\text{s}$
$t_{STP}$	STEP Active Time (High)	5			$\mu\text{s}$
$t_{SC}$	STEP Cycle Time		33		$\mu\text{s}$
$t_{FR}$	FAULT RESET Active Time (High)		8	10	$\mu\text{s}$
$t_{IDX}$	INDEX Pulse Width	625			$\mu\text{s}$
$t_{TC}$	Terminal Count Width		1		$t_{CY}$

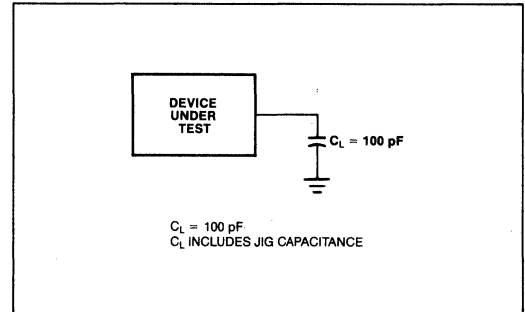
**NOTES:**

1. Typical values for  $T_A = 25^\circ\text{C}$  and nominal supply voltage.
2. The former values are used for standard floppy and the latter values are used for mini-floppies.
3.  $t_{SC} = 33\mu\text{s}$  min. is for different drive units. In the case of same unit,  $t_{SC}$  can be ranged from 1 ms to 16 ms with 8 MHz clock period, and 2 ms to 32 ms with 4MHz clock, under software control.
4. From 2.0V $\uparrow$  to +2.0V $\downarrow$ .

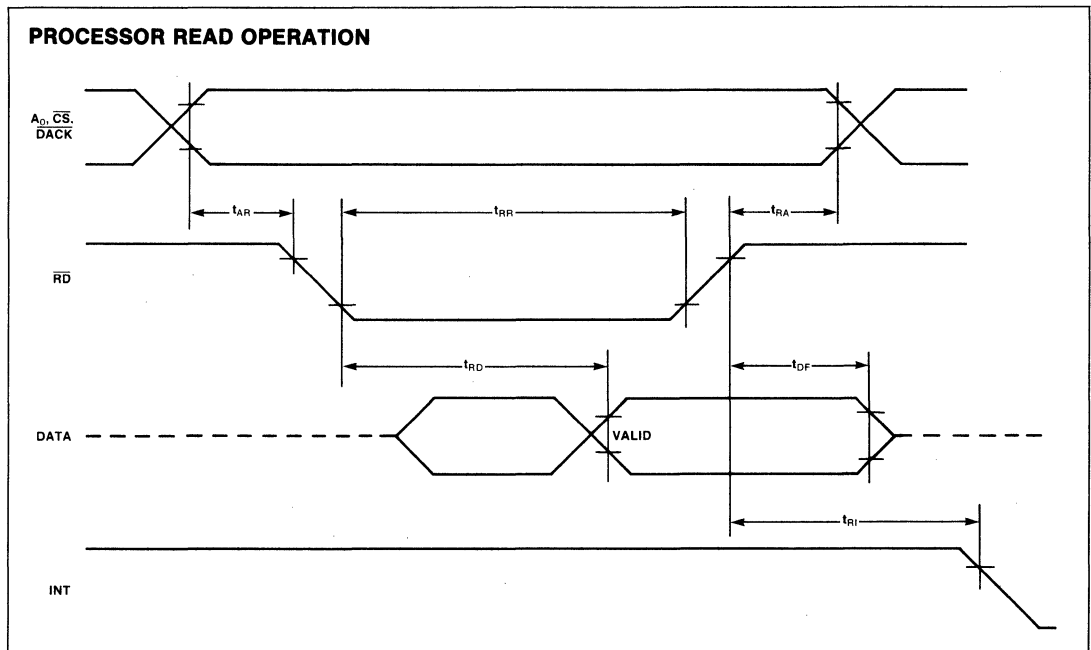
**A.C. TESTING INPUT, OUTPUT WAVEFORM**



**A.C. TESTING LOAD CIRCUIT**

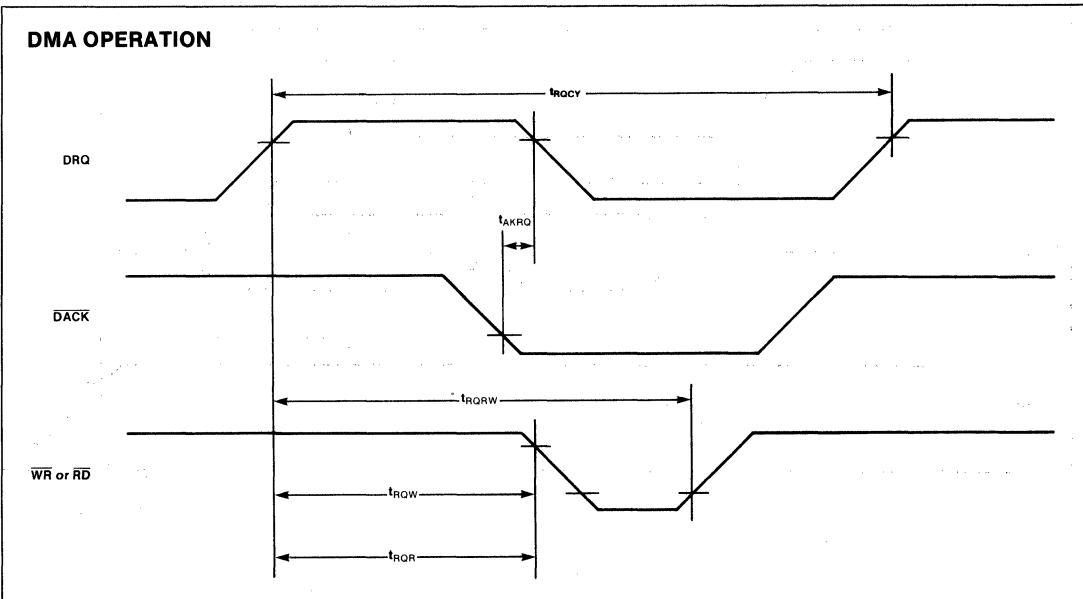
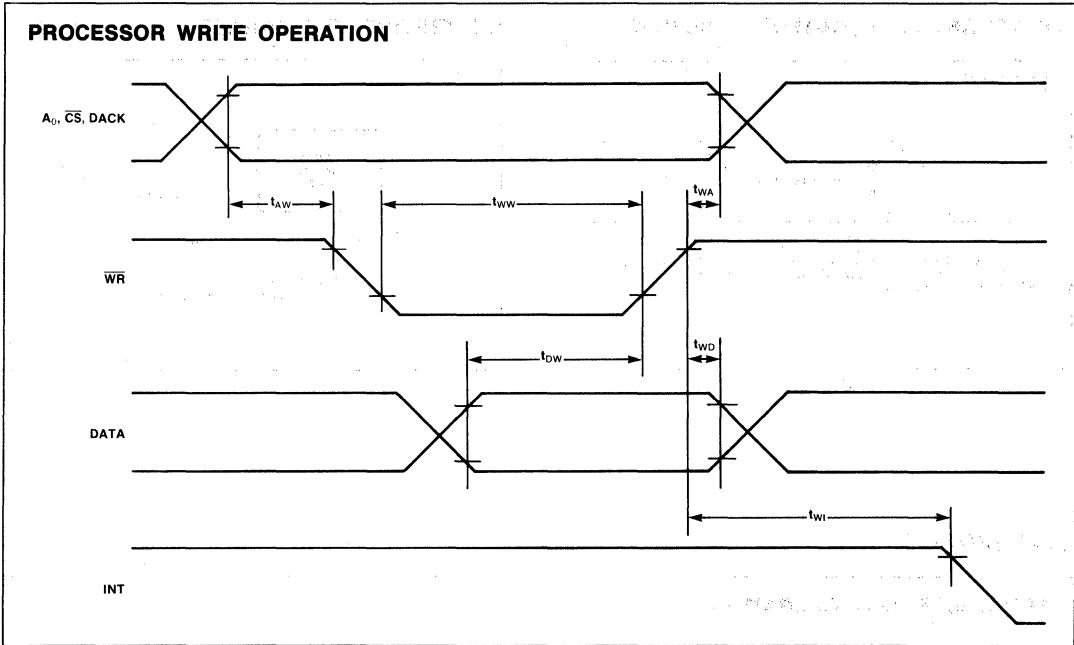


**WAVEFORMS**

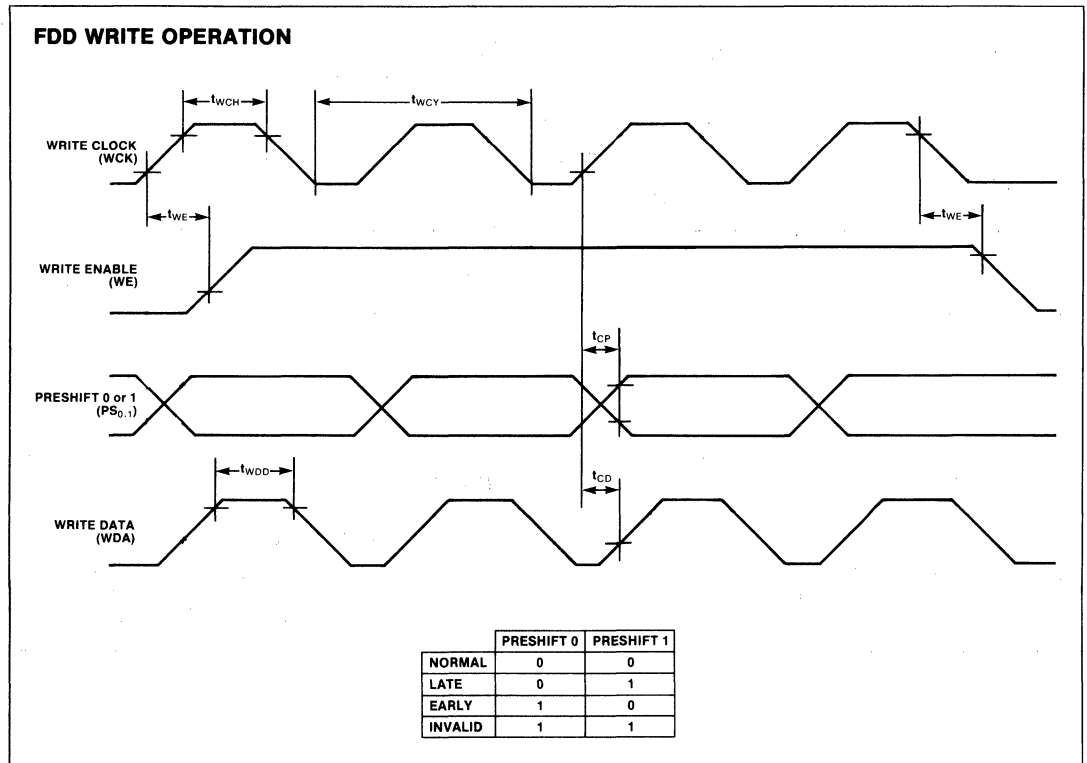
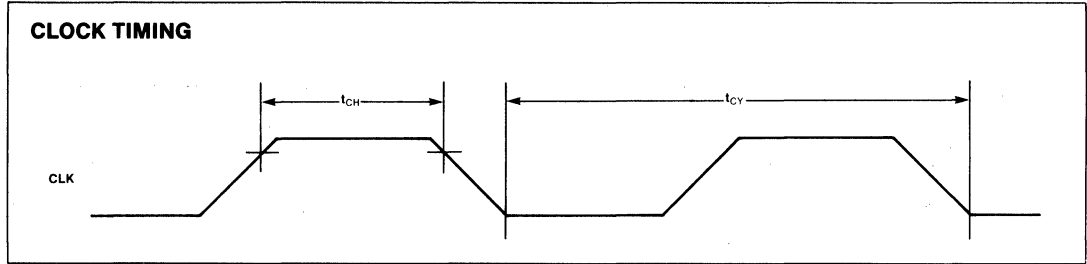




WAVEFORMS (Continued)

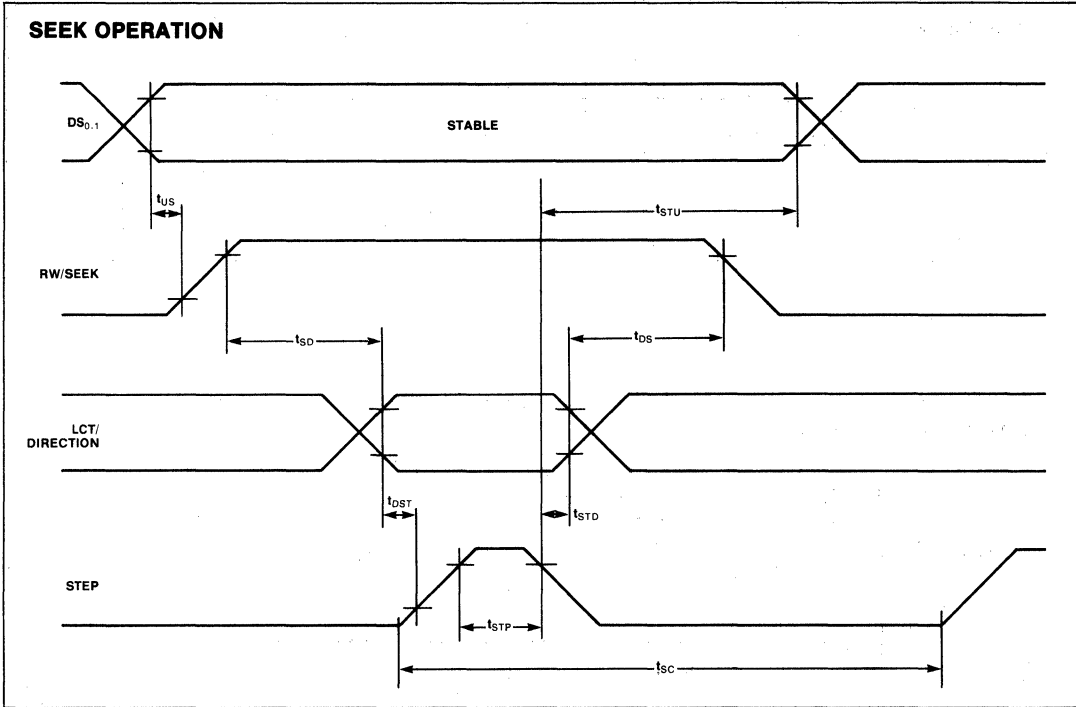


WAVEFORMS (Continued)

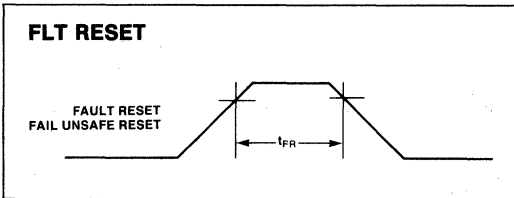


WAVEFORMS (Continued)

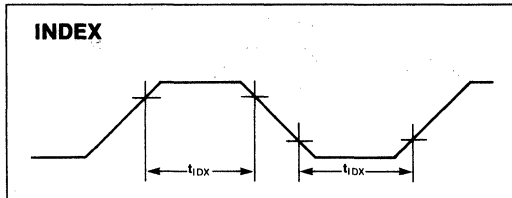
SEEK OPERATION



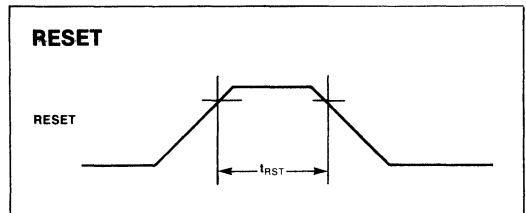
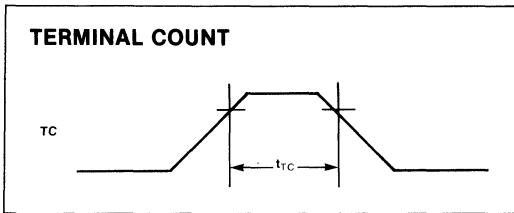
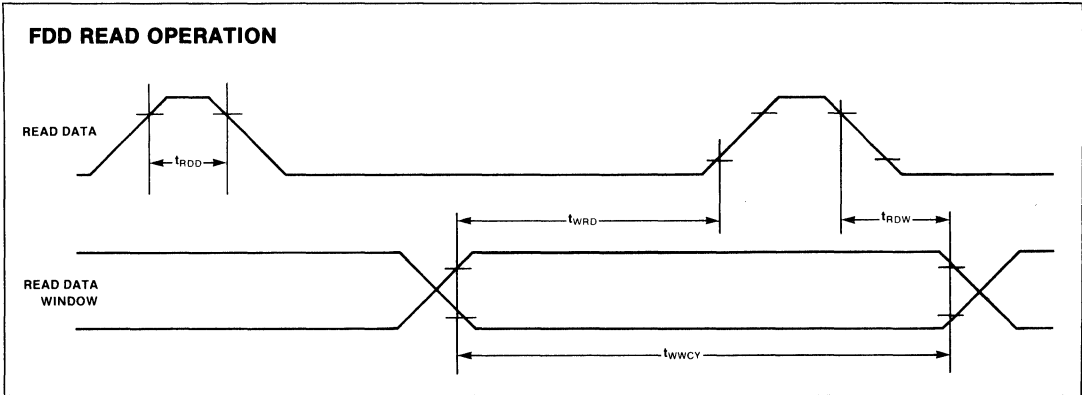
FLT RESET



INDEX



WAVEFORMS (Continued)





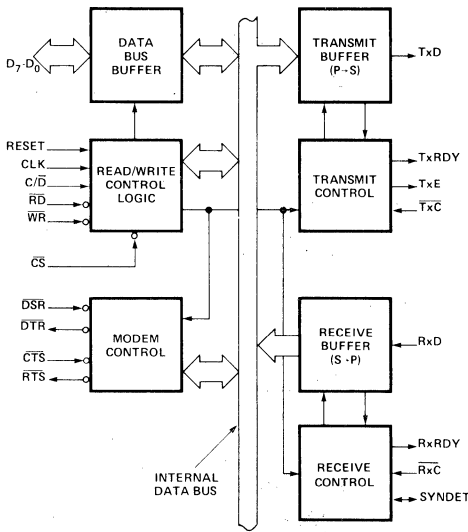
# ***Data Communications***

---

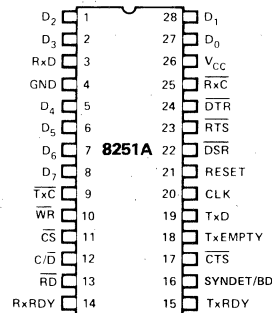
# 8251A PROGRAMMABLE COMMUNICATION INTERFACE

- Synchronous and Asynchronous Operation
- Synchronous 5–8 Bit Characters; Internal or External Character Synchronization; Automatic Sync Insertion
- Asynchronous 5–8 Bit Characters; Clock Rate—1, 16 or 64 Times Baud Rate; Break Character Generation; 1, 1½, or 2 Stop Bits; False Start Bit Detection; Automatic Break Detect and Handling
- Synchronous Baud Rate—DC to 64K Baud
- Asynchronous Baud Rate—DC to 19.2K Baud
- Full-Duplex, Double-Buffered Transmitter and Receiver
- Error Detection—Parity, Overrun and Framing
- Compatible with an Extended Range of Intel Microprocessors
- 28-Pin DIP Package
- All Inputs and Outputs are TTL Compatible
- Single +5V Supply
- Single TTL Clock

The Intel® 8251A is the enhanced version of the industry standard, Intel 8251 Universal Synchronous/Asynchronous Receiver/Transmitter (USART), designed for data communications with Intel's microprocessor families such as MCS-68, 80, 85, and iAPX-86, 88. The 8251A is used as a peripheral device and is programmed by the CPU to operate using virtually any serial data transmission technique presently in use (including IBM "bi-sync"). The USART accepts data characters from the CPU in parallel format and then converts them into a continuous serial data stream for transmission. Simultaneously, it can receive serial data streams and convert them into parallel data characters for the CPU. The USART will signal the CPU whenever it can accept a new character for transmission or whenever it has received a character for the CPU. The CPU can read the complete status of the USART at any time. These include data transmission errors and control signals such as SYNDET, TxEMPTY. The chip is fabricated using N-channel silicon gate technology.



**Figure 1. Block Diagram**



**Figure 2. Pin Configuration**

## FEATURES AND ENHANCEMENTS

The 8251A is an advanced design of the industry standard USART, the Intel® 8251. The 8251A operates with an extended range of Intel microprocessors and maintains compatibility with the 8251. Familiarization time is minimal because of compatibility and involves only knowing the additional features and enhancements, and reviewing the AC and DC specifications of the 8251A.

The 8251A incorporates all the key features of the 8251 and has the following additional features and enhancements:

- 8251A has double-buffered data paths with separate I/O registers for control, status, Data In, and Data Out, which considerably simplifies control programming and minimizes CPU overhead.
- In asynchronous operations, the Receiver detects and handles "break" automatically, relieving the CPU of this task.
- A refined Rx initialization prevents the Receiver from starting when in "break" state, preventing unwanted interrupts from a disconnected USART.
- At the conclusion of a transmission, TxD line will always return to the marking state unless SBRK is programmed.
- Tx Enable logic enhancement prevents a Tx Disable command from halting transmission until all data previously written has been transmitted. The logic also prevents the transmitter from turning off in the middle of a word.
- When External Sync Detect is programmed, Internal Sync Detect is disabled, and an External Sync Detect status is provided via a flip-flop which clears itself upon a status read.
- Possibility of false sync detect is minimized by ensuring that if double character sync is programmed, the characters be contiguously detected and also by clearing the Rx register to all ones whenever Enter Hunt command is issued in Sync mode.
- As long as the 8251A is not selected, the  $\overline{RD}$  and  $\overline{WR}$  do not affect the internal operation of the device.
- The 8251A Status can be read at any time but the status update will be inhibited during status read.
- The 8251A is free from extraneous glitches and has enhanced AC and DC characteristics, providing higher speed and better operating margins.
- Synchronous Baud rate from DC to 64K.

## FUNCTIONAL DESCRIPTION

### General

The 8251A is a Universal Synchronous/Asynchronous Receiver/Transmitter designed for a wide range of Intel microcomputers such as 8048, 8080, 8085, 8086 and 8088. Like other I/O devices in a microcomputer system, its functional configuration is programmed by the system's software for maximum flexibility. The 8251A can support most serial data techniques in use, including IBM "bi-sync."

In a communication environment an interface device must convert parallel format system data into serial format for transmission and convert incoming serial format data into parallel system data for reception. The interface device must also delete or insert bits or characters that are functionally unique to the communication technique. In essence, the interface should appear "transparent" to the CPU, a simple input or output of byte-oriented system data.

### Data Bus Buffer

This 3-state, bidirectional, 8-bit buffer is used to interface the 8251A to the system Data Bus. Data is transmitted or received by the buffer upon execution of INput or OUTput instructions of the CPU. Control words, Command words and Status information are also transferred through the Data Bus Buffer. The Command Status, Data-In and Data-Out registers are separate, 8-bit registers communicating with the system bus through the Data Bus Buffer.

This functional block accepts inputs from the system Control bus and generates control signals for overall device operation. It contains the Control Word Register and Command Word Register that store the various control formats for the device functional definition.

### RESET (Reset)

A "high" on this input forces the 8251A into an "Idle" mode. The device will remain at "Idle" until a new set of control words is written into the 8251A to program its functional definition. Minimum RESET pulse width is  $6 t_{CY}$  (clock must be running).

A command reset operation also puts the device into the "Idle" state.

### CLK (Clock)

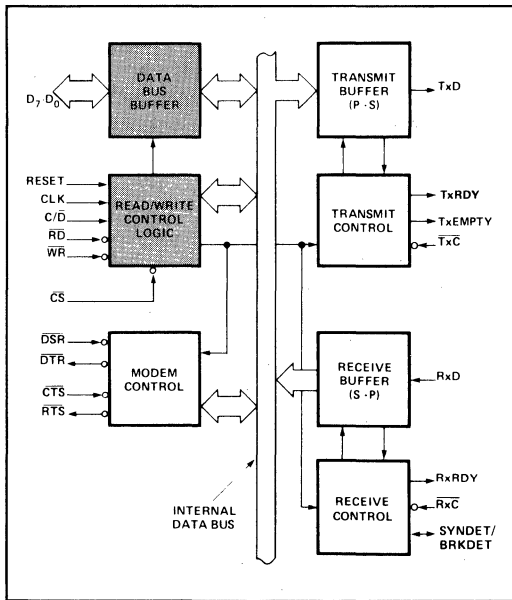
The CLK input is used to generate internal device timing and is normally connected to the Phase 2 (TTL) output of the Clock Generator. No external inputs or outputs are referenced to CLK but the frequency of CLK must be greater than 30 times the Receiver or Transmitter data bit rates.

### $\overline{WR}$ (Write)

A "low" on this input informs the 8251A that the CPU is writing data or control words to the 8251A.

### $\overline{RD}$ (Read)

A "low" on this input informs the 8251A that the CPU is reading data or status information from the 8251A.



**Figure 3. 8251A Block Diagram Showing Data Bus Buffer and Read/Write Logic Functions**

C/D	$\overline{RD}$	$\overline{WR}$	$\overline{CS}$	
0	0	1	0	8251A DATA $\Rightarrow$ DATA BUS
0	1	0	0	DATA BUS $\Rightarrow$ 8251A DATA
1	0	1	0	STATUS $\Rightarrow$ DATA BUS
1	1	0	0	DATA BUS $\Rightarrow$ CONTROL
X	1	1	0	DATA BUS $\Rightarrow$ 3-STATE
X	X	X	1	DATA BUS $\Rightarrow$ 3-STATE

### C/D (Control/Data)

This input, in conjunction with the  $\overline{WR}$  and  $\overline{RD}$  inputs, informs the 8251A that the word on the Data Bus is either a data character, control word or status information.

1 = CONTROL/STATUS; 0 = DATA.

### $\overline{CS}$ (Chip Select)

A "low" on this input selects the 8251A. No reading or writing will occur unless the device is selected. When  $\overline{CS}$  is high, the Data Bus is in the float state and  $\overline{RD}$  and  $\overline{WR}$  have no effect on the chip.

### Modem Control

The 8251A has a set of control inputs and outputs that can be used to simplify the interface to almost any modem. The modem control signals are general purpose in nature and can be used for functions other than modem control, if necessary.

### $\overline{DSR}$ (Data Set Ready)

The  $\overline{DSR}$  input signal is a general-purpose, 1-bit inverting input port. Its condition can be tested by the CPU using a Status Read operation. The  $\overline{DSR}$  input is normally used to test modem conditions such as Data Set Ready.

### $\overline{DTR}$ (Data Terminal Ready)

The  $\overline{DTR}$  output signal is a general-purpose, 1-bit inverting output port. It can be set "low" by programming the appropriate bit in the Command Instruction word. The  $\overline{DTR}$  output signal is normally used for modem control such as Data Terminal Ready.

### $\overline{RTS}$ (Request to Send)

The  $\overline{RTS}$  output signal is a general-purpose, 1-bit inverting output port. It can be set "low" by programming the appropriate bit in the Command Instruction word. The  $\overline{RTS}$  output signal is normally used for modem control such as Request to Send.

### $\overline{CTS}$ (Clear to Send)

A "low" on this input enables the 8251A to transmit serial data if the Tx Enable bit in the Command byte is set to a "one." If either a Tx Enable off or  $\overline{CTS}$  off condition occurs while the Tx is in operation, the Tx will transmit all the data in the USART, written prior to Tx Disable command before shutting down.



## Transmitter Buffer

The Transmitter Buffer accepts parallel data from the Data Bus Buffer, converts it to a serial bit stream, inserts the appropriate characters or bits (based on the communication technique) and outputs a composite serial stream of data on the TxD output pin on the falling edge of  $\overline{\text{TxC}}$ . The transmitter will begin transmission upon being enabled if  $\overline{\text{CTS}} = 0$ . The TxD line will be held in the marking state immediately upon a master Reset or when Tx Enable or  $\overline{\text{CTS}}$  is off or the transmitter is empty.

## Transmitter Control

The Transmitter Control manages all activities associated with the transmission of serial data. It accepts and issues signals both externally and internally to accomplish this function.

## TxDY (Transmitter Ready)

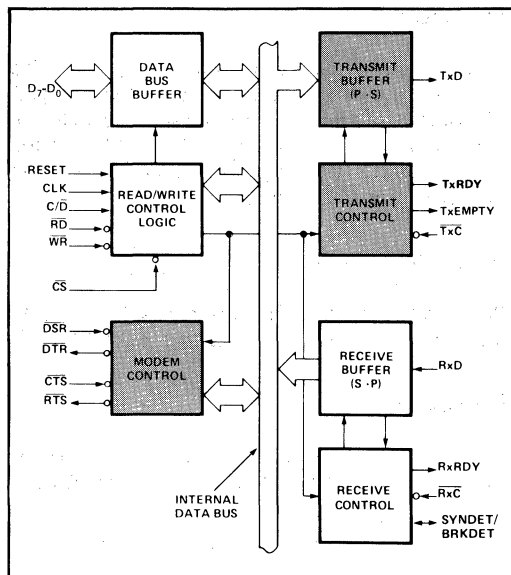
This output signals the CPU that the transmitter is ready to accept a data character. The TxDY output pin can be used as an interrupt to the system, since it is masked by TxEnable; or, for Polled operation, the CPU can check TxDY using a Status Read operation. TxDY is automatically reset by the leading edge of  $\overline{\text{WR}}$  when a data character is loaded from the CPU.

Note that when using the Polled operation, the TxDY status bit is *not* masked by TxEnable, but will only indicate the Empty/Full Status of the Tx Data Input Register.

## TxE (Transmitter Empty)

When the 8251A has no characters to send, the TxEMPTY output will go "high." It resets upon receiving a character from the CPU if the transmitter is enabled. TxEMPTY remains low when the transmitter is disabled even if it is actually empty. TxEMPTY can be used to indicate the end of a transmission mode, so that the CPU "knows" when to "turn the line around" in the half-duplex operational mode.

In the Synchronous mode, a "high" on this output indicates that a character has not been loaded and the SYNC character or characters are about to be or are being transmitted automatically as "fillers." TxEMPTY does not go low when the SYNC characters are being shifted out.



**Figure 4. 8251A Block Diagram Showing Modem and Transmitter Buffer and Control Functions**

## $\overline{\text{TxC}}$ (Transmitter Clock)

The Transmitter Clock controls the rate at which the character is to be transmitted. In the Synchronous transmission mode, the Baud Rate (1x) is equal to the  $\overline{\text{TxC}}$  frequency. In Asynchronous transmission mode, the baud rate is a fraction of the actual  $\overline{\text{TxC}}$  frequency. A portion of the mode instruction selects this factor; it can be 1, 1/16 or 1/64 the  $\overline{\text{TxC}}$ .

For Example:

If Baud Rate equals 110 Baud,  
 $\overline{\text{TxC}}$  equals 110 Hz in the 1x mode.  
 $\overline{\text{TxC}}$  equals 1.72 kHz in the 16x mode.  
 $\overline{\text{TxC}}$  equals 7.04 kHz in the 64x mode.

The falling edge of  $\overline{\text{TxC}}$  shifts the serial data out of the 8251A.

## Receiver Buffer

The Receiver accepts serial data, converts this serial input to parallel format, checks for bits or characters that are unique to the communication technique and sends an "assembled" character to the CPU. Serial data is input to RxD pin, and is clocked in on the rising edge of  $\overline{\text{RxC}}$ .

## Receiver Control

This functional block manages all receiver-related activities which consists of the following features.

The RxD initialization circuit prevents the 8251A from mistaking an unused input line for an active low data line in the "break condition." Before starting to receive serial characters on the RxD line, a valid "1" must first be detected after a chip master Reset. Once this has been determined, a search for a valid low (Start bit) is enabled. This feature is only active in the asynchronous mode, and is only done once for each master Reset.

The False Start bit detection circuit prevents false starts due to a transient noise spike by first detecting the falling edge and then strobing the nominal center of the Start bit (RxD = low).

Parity error detection sets the corresponding status bit.

The Framing Error status bit is set if the Stop bit is absent at the end of the data byte (asynchronous mode).

## RxRDY (Receiver Ready)

This output indicates that the 8251A contains a character that is ready to be input to the CPU. RxRDY can be connected to the interrupt structure of the CPU or, for polled operation, the CPU can check the condition of RxRDY using a Status Read operation.

RxEnable, when off, holds RxRDY in the Reset Condition. For Asynchronous mode, to set RxRDY, the Receiver must be enabled to sense a Start Bit and a complete character must be assembled and transferred to the Data Output Register. For Synchronous mode, to set RxRDY, the Receiver must be enabled and a character must finish assembly and be transferred to the Data Output Register.

Failure to read the received character from the Rx Data Output Register prior to the assembly of the next Rx Data character will set overrun condition error and the previous character will be written over and lost. If the Rx Data is being read by the CPU when the internal transfer is occurring, overrun error will be set and the old character will be lost.

## RxC (Receiver Clock)

The Receiver Clock controls the rate at which the character is to be received. In Synchronous Mode, the Baud Rate (1x) is equal to the actual frequency of  $\overline{RxC}$ . In Asynchronous Mode, the Baud Rate is a fraction of the actual  $\overline{RxC}$  frequency. A portion of the mode instruction selects this factor: 1, 1/16 or 1/64 the  $\overline{RxC}$ .

For example:

Baud Rate equals 300 Baud, if  
 $\overline{RxC}$  equals 300 Hz in the 1x mode;  
 $\overline{RxC}$  equals 4800 Hz in the 16x mode;  
 $\overline{RxC}$  equals 19.2 kHz in the 64x mode.

Baud Rate equals 2400 Baud, if  
 $\overline{RxC}$  equals 2400 Hz in the 1x mode;  
 $\overline{RxC}$  equals 38.4 kHz in the 16x mode;  
 $\overline{RxC}$  equals 153.6 kHz in the 64x mode.

Data is sampled into the 8251A on the rising edge of  $\overline{RxC}$ .

**NOTE:** In most communications systems, the 8251A will be handling both the transmission and reception operations of a single link. Consequently, the Receive and Transmit Baud Rates will be the same. Both  $\overline{TxC}$  and  $\overline{RxC}$  will require identical frequencies for this operation and can be tied together and connected to a single frequency source (Baud Rate Generator) to simplify the interface.

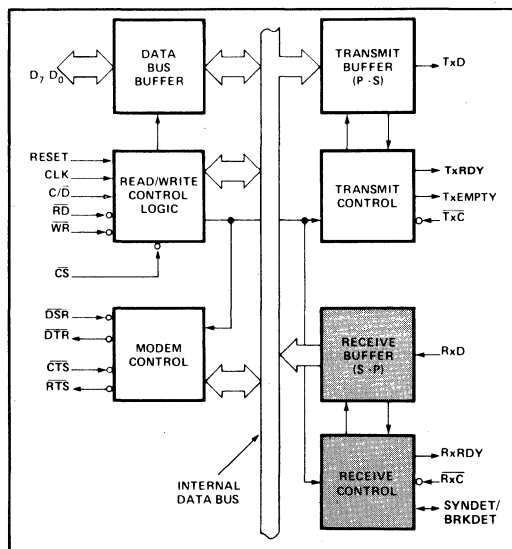


Figure 5. 8251A Block Diagram Showing Receiver Buffer and Control Functions

**SYNDET (SYNC Detect/  
BRKDET Break Detect)**

This pin is used in Synchronous Mode for SYNDET and may be used as either input or output, programmable through the Control Word. It is reset to output mode low upon RESET. When used as an output (internal Sync mode), the SYNDET pin will go "high" to indicate that the 8251A has located the SYNC character in the Receive mode. If the 8251A is programmed to use double Sync characters (bi-sync), then SYNDET will go "high" in the middle of the last bit of the second Sync character. SYNDET is automatically reset upon a Status Read operation.

When used as an input (external SYNC detect mode), a positive going signal will cause the 8251A to start assembling data characters on the rising edge of the next  $\overline{RxC}$ . Once in SYNC, the "high" input signal can be removed. When External SYNC Detect is programmed, Internal SYNC Detect is disabled.

**BREAK (Async Mode Only)**

This output will go high whenever the receiver remains low through two consecutive stop bit sequences (including the start bits, data bits, and parity bits). Break Detect may also be read as a Status bit. It is reset only upon a master chip Reset or Rx Data returning to a "one" state.

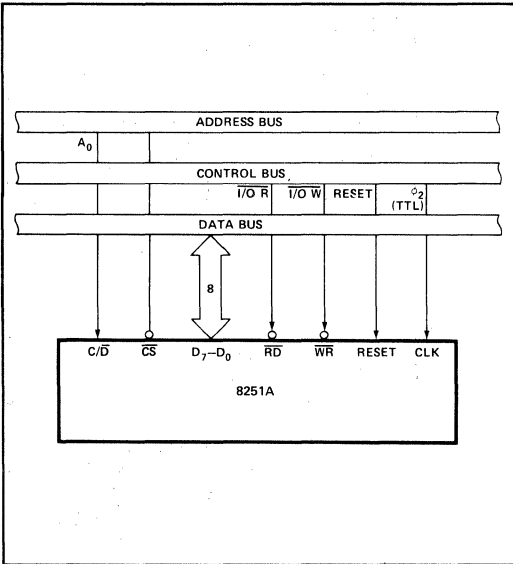


Figure 6. 8251A Interface to 8080 Standard System Bus

**DETAILED OPERATION DESCRIPTION**

**General**

The complete functional definition of the 8251A is programmed by the system's software. A set of control words must be sent out by the CPU to initialize the 8251A to support the desired communications format. These control words will program the: BAUD RATE, CHARACTER LENGTH, NUMBER OF STOP BITS, SYNCHRONOUS or ASYNCHRONOUS OPERATION, EVEN/ODD/OFF PARITY, etc. In the Synchronous Mode, options are also provided to select either internal or external character synchronization.

Once programmed, the 8251A is ready to perform its communication functions. The TxRDY output is raised "high" to signal the CPU that the 8251A is ready to receive a data character from the CPU. This output (TxRDY) is reset automatically when the CPU writes a character into the 8251A. On the other hand, the 8251A receives serial data from the MODEM or I/O device. Upon receiving an entire character, the RxRDY output is raised "high" to signal the CPU that the 8251A has a complete character ready for the CPU to fetch. RxRDY is reset automatically upon the CPU data read operation.

The 8251A cannot begin transmission until the Tx Enable (Transmitter Enable) bit is set in the Command Instruction and it has received a Clear To Send (CTS) input. The TxD output will be held in the marking state upon Reset.

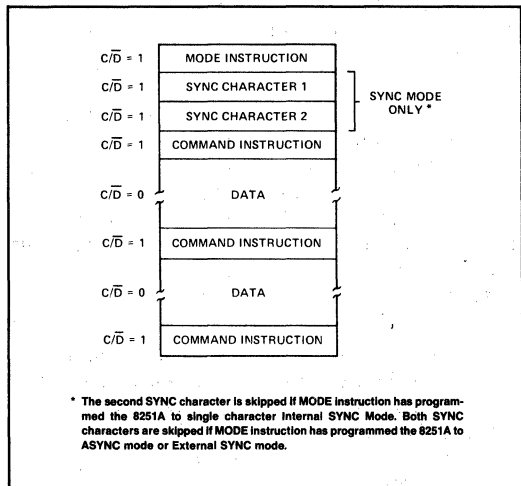


Figure 7. Typical Data Block

## Programming the 8251A

Prior to starting data transmission or reception, the 8251A must be loaded with a set of control words generated by the CPU. These control signals define the complete functional definition of the 8251A and must immediately follow a Reset operation (internal or external).

The control words are split into two formats:

1. Mode Instruction
2. Command Instruction

### Mode Instruction

This instruction defines the general operational characteristics of the 8251A. It must follow a Reset operation (internal or external). Once the Mode Instruction has been written into the 8251A by the CPU, SYNC characters or Command Instructions may be written.

### Command Instruction

This instruction defines a word that is used to control the actual operation of the 8251A.

Both the Mode and Command Instructions must conform to a specified sequence for proper device operation (see Figure 7). The Mode Instruction must be written immediately following a Reset operation, prior to using the 8251A for data communication.

All control words written into the 8251A after the Mode Instruction will load the Command Instruction. Command Instructions can be written into the 8251A at any time in the data block during the operation of the 8251A. To return to the Mode Instruction format, the master Reset bit in the Command Instruction word can be set to initiate an internal Reset operation which automatically places the 8251A back into the Mode Instruction format. Command Instructions must follow the Mode Instructions or Sync characters.

### Mode Instruction Definition

The 8251A can be used for either Asynchronous or Synchronous data communication. To understand how the Mode Instruction defines the functional operation of the 8251A, the designer can best view the device as two separate components, one Asynchronous and the other Synchronous, sharing

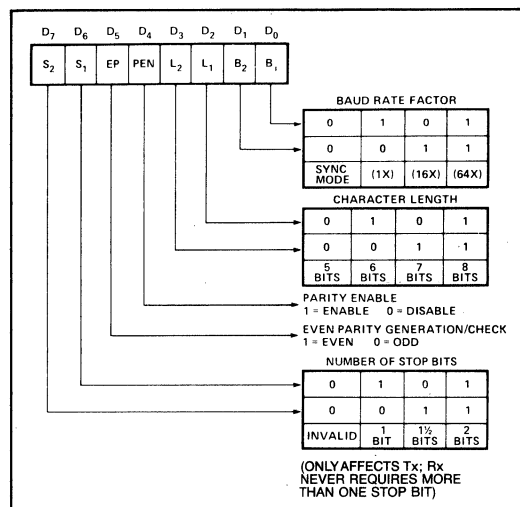
the same package. The format definition can be changed only after a master chip Reset. For explanation purposes the two formats will be isolated.

**NOTE:** When parity is enabled it is not considered as one of the data bits for the purpose of programming the word length. The actual parity bit received on the Rx Data line cannot be read on the Data Bus. In the case of a programmed character length of less than 8 bits, the least significant Data Bus bits will hold the data; unused bits are “don’t care” when writing data to the 8251A, and will be “zeros” when reading the data from the 8251A.

### Asynchronous Mode (Transmission)

Whenever a data character is sent by the CPU the 8251A automatically adds a Start bit (low level) followed by the data bits (least significant bit first), and the programmed number of Stop bits to each character. Also, an even or odd Parity bit is inserted prior to the Stop bit(s), as defined by the Mode Instruction. The character is then transmitted as a serial data stream on the TxD output. The serial data is shifted out on the falling edge of TxC at a rate equal to 1, 1/16, or 1/64 that of the TxC, as defined by the Mode Instruction. BREAK characters can be continuously sent to the TxD if commanded to do so.

When no data characters have been loaded into the 8251A the TxD output remains “high” (marking) unless a Break (continuously low) has been programmed.



**Figure 8. Mode Instruction Format, Asynchronous Mode**

### Asynchronous Mode (Receive)

The RxD line is normally high. A falling edge on this line triggers the beginning of a START bit. The validity of this START bit is checked by again strobing this bit at its nominal center (16X or 64X mode only). If a low is detected again, it is a valid START bit, and the bit counter will start counting. The bit counter thus locates the center of the data bits, the parity bit (if it exists) and the stop bits. If parity error occurs, the parity error flag is set. Data and parity bits are sampled on the RxD pin with the rising edge of  $\overline{RxC}$ . If a low level is detected as the STOP bit, the Framing Error flag will be set. The STOP bit signals the end of a character. Note that the receiver requires only *one* stop bit, regardless of the number of stop bits programmed. This character is then loaded into the parallel I/O buffer of the 8251A. The RxDY pin is raised to signal the CPU that a character is ready to be fetched. If a previous character has not been fetched by the CPU, the present character replaces it in the I/O buffer, and the **OVERRUN** Error flag is raised (thus the previous character is lost). All of the error flags can be reset by an Error Reset Instruction. The occurrence of any of these errors will not affect the operation of the 8251A.

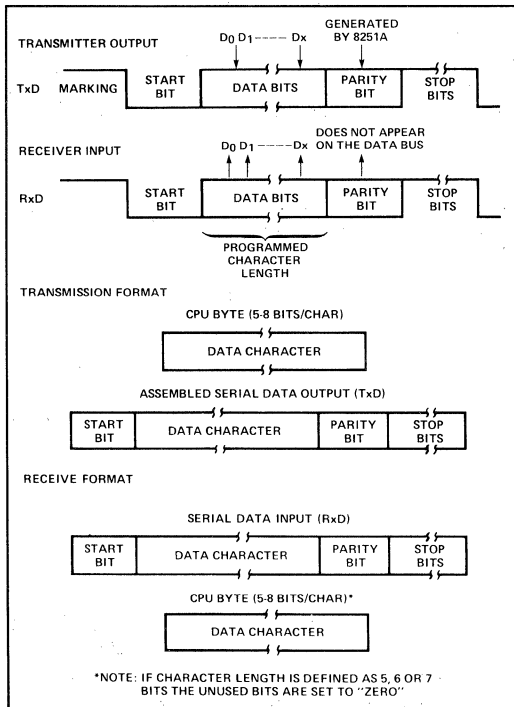
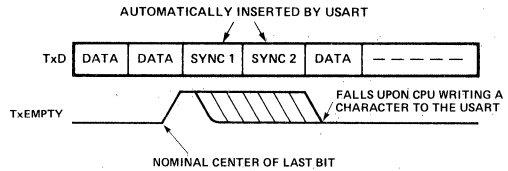


Figure 9. Asynchronous Mode

### Synchronous Mode (Transmission)

The TxD output is continuously high until the CPU sends its first character to the 8251A which usually is a SYNC character. When the  $\overline{CTS}$  line goes low, the first character is serially transmitted out. All characters are shifted out on the falling edge of  $\overline{TxC}$ . Data is shifted out at the same rate as the  $\overline{TxC}$ .

Once transmission has started, the data stream at the TxD output must continue at the  $\overline{TxC}$  rate. If the CPU does not provide the 8251A with a data character before the 8251A Transmitter Buffers become empty, the SYNC characters (or character if in single SYNC character mode) will be automatically inserted in the TxD data stream. In this case, the TxEMPTY pin is raised high to signal that the 8251A is empty and SYNC characters are being sent out. TxEMPTY does not go low when the SYNC is being shifted out (see figure below). The TxEMPTY pin is internally reset by a data character being written into the 8251A.



### Synchronous Mode (Receive)

In this mode, character synchronization can be internally or externally achieved. If the SYNC mode has been programmed, ENTER HUNT command should be included in the first command instruction word written. Data on the RxD pin is then sampled on the rising edge of  $\overline{RxC}$ . The content of the Rx buffer is compared at every bit boundary with the first SYNC character until a match occurs. If the 8251A has been programmed for two SYNC characters, the subsequent received character is also compared; when both SYNC characters have been detected, the USART ends the HUNT mode and is in character synchronization. The SYNDDET pin is then set high, and is reset automatically by a STATUS READ. If parity is programmed, SYNDDET will not be set until the middle of the parity bit instead of the middle of the last data bit.

In the external SYNC mode, synchronization is achieved by applying a high level on the SYNDDET pin, thus forcing the 8251A out of the HUNT mode. The high level can be removed after one  $\overline{RxC}$  cycle. An ENTER HUNT command has no effect in the asynchronous mode of operation.

Parity error and overrun error are both checked in the same way as in the Asynchronous Rx mode. Parity is checked when not in Hunt, regardless of whether the Receiver is enabled or not.

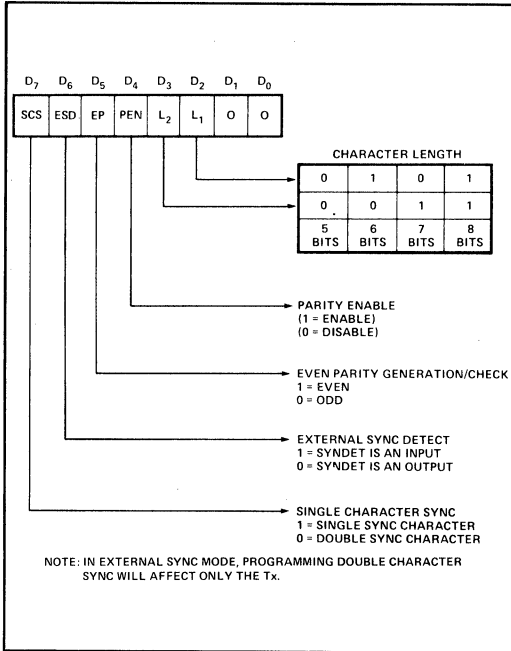


Figure 10. Mode Instruction Format, Synchronous Mode

The CPU can command the receiver to enter the HUNT mode if synchronization is lost. This will also set all the used character bits in the buffer to a "one," thus preventing a possible false SYNDET caused by data that happens to be in the Rx Buffer at ENTER HUNT time. Note that the SYNDET F/F is reset at each Status Read, regardless of whether internal or external SYNC has been programmed. This does not cause the 8251A to return to the HUNT mode. When in SYNC mode, but not in HUNT, Sync Detection is still functional, but only occurs at the "known" word boundaries. Thus, if one Status Read indicates SYNDET and a second Status Read also indicates SYNDET, then the programmed SYNDET characters have been received since the previous Status Read. (If double character sync has been programmed, then both sync characters have been contiguously received to gate a SYNDET indication.) When external SYNDET mode is selected, internal Sync Detect is disabled, and the SYNDET F/F may be set at any bit boundary.

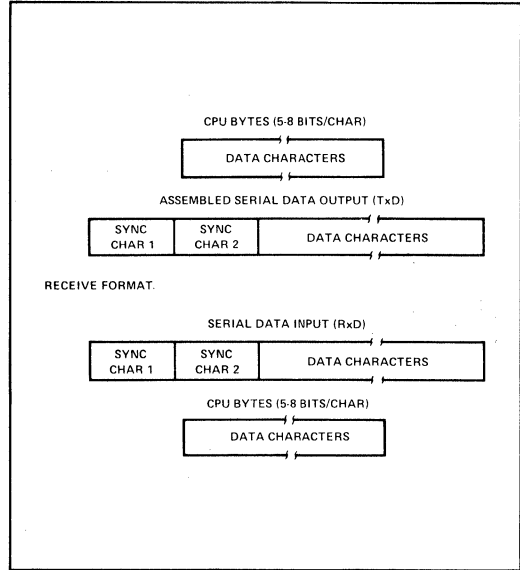


Figure 11. Data Format, Synchronous Mode

### COMMAND INSTRUCTION DEFINITION

Once the functional definition of the 8251A has been programmed by the Mode Instruction and the sync characters are loaded (if in Sync Mode) then the device is ready to be used for data communication. The Command Instruction controls the actual operation of the selected format. Functions such as: Enable Transmit/Receive, Error Reset and Modem Controls are provided by the Command Instruction.

Once the Mode Instruction has been written into the 8251A and Sync characters inserted, if necessary, then all further "control writes" (C/D = 1) will load a Command Instruction. A Reset Operation (internal or external) will return the 8251A to the Mode Instruction format.

**Note:** Internal Reset on Power-up

When power is first applied, the 8251A may come up in the Mode, Sync character or Command format. To guarantee that the device is in the Command Instruction format before the Reset command is issued, it is safest to execute the worst-case initialization sequence (sync mode with two sync characters). Loading three 00Hs consecutively into the device with C/D = 1 configures sync operation and writes two dummy 00H sync characters. An Internal Reset command (40H) may then be issued to return the device to the "Idle" state.

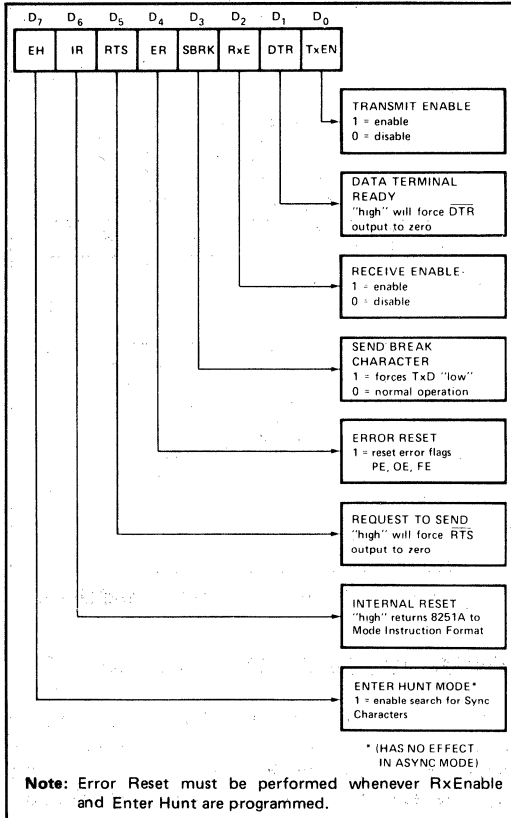


Figure 12. Command Instruction Format

**STATUS READ DEFINITION**

In data communication systems it is often necessary to examine the "status" of the active device to ascertain if errors have occurred or other conditions that require the processor's attention. The 8251A has facilities that allow the programmer to "read" the status of the device at any time during the functional operation. (Status update is inhibited during status read.)

A normal "read" command is issued by the CPU with C/D = 1 to accomplish this function.

Some of the bits in the Status Read Format have identical meanings to external output pins so that the 8251A can be used in a completely polled or interrupt-driven environment. TxRDY is an exception.

Note that status update can have a maximum delay of 28 clock periods from the actual event affecting the status.

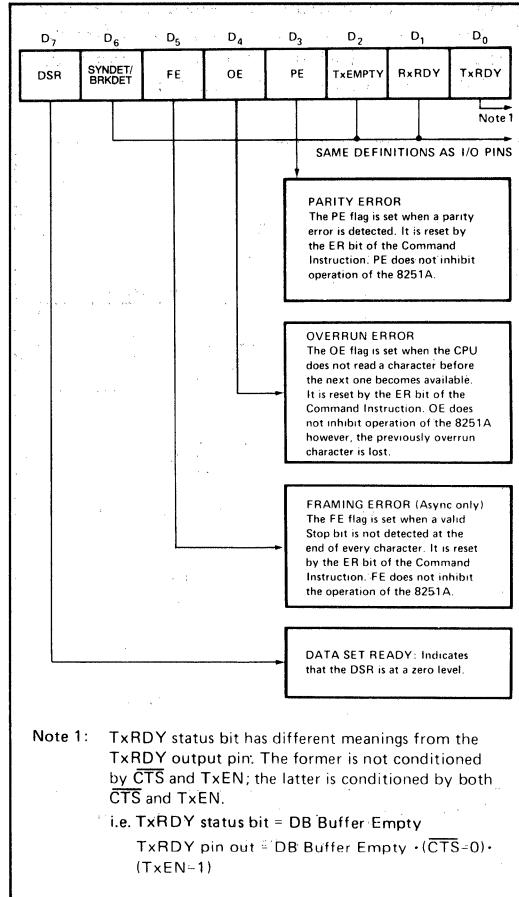


Figure 13. Status Read Format

**APPLICATIONS OF THE 8251A**

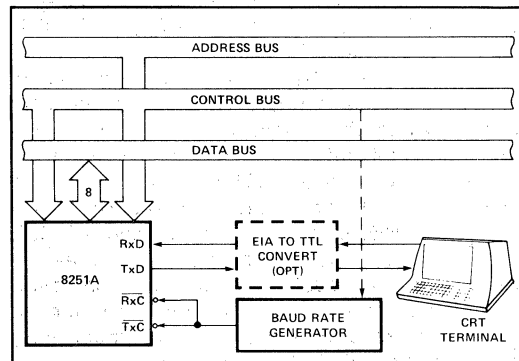
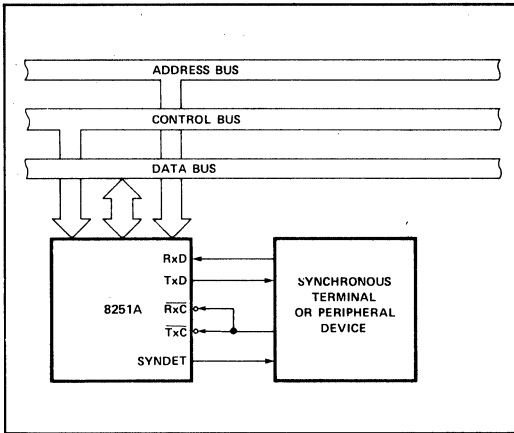
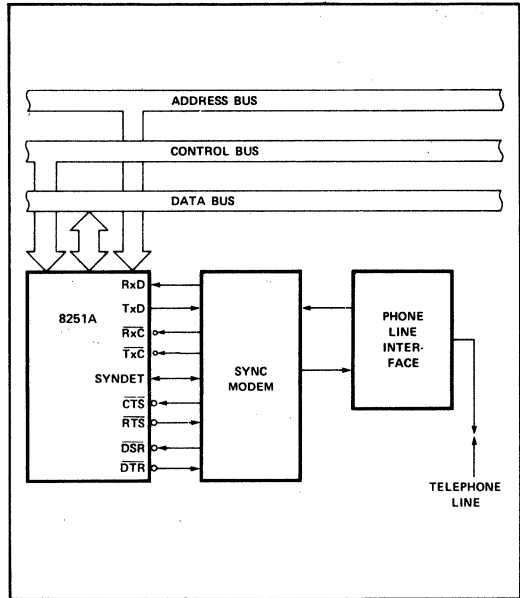


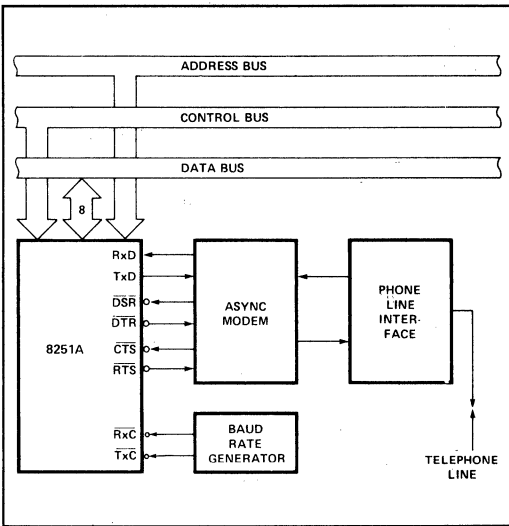
Figure 14. Asynchronous Serial Interface to CRT Terminal, DC-9600 Baud



**Figure 15. Synchronous Interface to Terminal or Peripheral Device**



**Figure 17. Synchronous Interface to Telephone Lines**



**Figure 16. Asynchronous Interface to Telephone Lines**



**ABSOLUTE MAXIMUM RATINGS\***

Ambient Temperature Under Bias	..... 0°C to 70°C
Storage Temperature	..... -65°C to +150°C
Voltage On Any Pin	
With Respect To Ground	..... -0.5V to +7V
Power Dissipation	..... 1 Watt

\*NOTICE: Stresses above those listed under "Absolute Maximum Ratings" may cause permanent damage to the device. This is a stress rating only and functional operation of the device at these or any other conditions above those indicated in the operational sections of this specification is not implied. Exposure to absolute maximum rating conditions for extended periods may affect device reliability.

**D.C. CHARACTERISTICS** ( $T_A = 0^\circ\text{C}$  to  $70^\circ\text{C}$ ,  $V_{CC} = 5.0\text{V} \pm 5\%$ ,  $\text{GND} = 0\text{V}$ )

Symbol	Parameter	Min.	Max.	Unit	Test Conditions
$V_{IL}$	Input Low Voltage	-0.5	0.8	V	
$V_{IH}$	Input High Voltage	2.2	$V_{CC}$	V	
$V_{OL}$	Output Low Voltage		0.45	V	$I_{OL} = 2.2\text{ mA}$
$V_{OH}$	Output High Voltage	2.4		V	$I_{OL} = -400\ \mu\text{A}$
$I_{OFL}$	Output Float Leakage		$\pm 10$	$\mu\text{A}$	$V_{OUT} = V_{CC}$ TO 0.45V
$I_{IL}$	Input Leakage		$\pm 10$	$\mu\text{A}$	$V_{IN} = V_{CC}$ TO 0.45V
$I_{CC}$	Power Supply Current		100	mA	All Outputs = High

**CAPACITANCE** ( $T_A = 25^\circ\text{C}$ ,  $V_{CC} = \text{GND} = 0\text{V}$ )

Symbol	Parameter	Min.	Max.	Unit	Test Conditions
$C_{IN}$	Input Capacitance		10	pF	$f_c = 1\text{MHz}$
$C_{I/O}$	I/O Capacitance		20	pF	Unmeasured pins returned to GND

**A.C. CHARACTERISTICS** ( $T_A = 0^\circ\text{C}$  to  $70^\circ\text{C}$ ,  $V_{CC} = 5.0\text{V} \pm 5\%$ ,  $\text{GND} = 0\text{V}$ )

**Bus Parameters** (Note 1)

**READ CYCLE**

Symbol	Parameter	Min.	Max.	Unit	Test Conditions
$t_{AR}$	Address Stable Before $\overline{\text{READ}}$ ( $\overline{\text{CS}}$ , $\text{C}/\overline{\text{D}}$ )	0		ns	Note 2
$t_{RA}$	Address Hold Time for $\overline{\text{READ}}$ ( $\overline{\text{CS}}$ , $\text{C}/\overline{\text{D}}$ )	0		ns	Note 2
$t_{RR}$	$\overline{\text{READ}}$ Pulse Width	250		ns	
$t_{RD}$	Data Delay from $\overline{\text{READ}}$		200	ns	3, $C_L = 150\text{ pF}$
$t_{DF}$	$\overline{\text{READ}}$ to Data Floating	10	100	ns	

**WRITE CYCLE**

Symbol	Parameter	Min.	Max.	Unit	Test Conditions
$t_{AW}$	Address Stable Before $\overline{\text{WRITE}}$	0		ns	
$t_{WA}$	Address Hold Time for $\overline{\text{WRITE}}$	0		ns	
$t_{WW}$	$\overline{\text{WRITE}}$ Pulse Width	250		ns	
$t_{DW}$	Data Set-Up Time for $\overline{\text{WRITE}}$	150		ns	
$t_{WD}$	Data Hold Time for $\overline{\text{WRITE}}$	20		ns	
$t_{RV}$	Recovery Time Between WRITES	6		$t_{CY}$	Note 4

**A.C. CHARACTERISTICS (Continued)**
**OTHER TIMINGS**

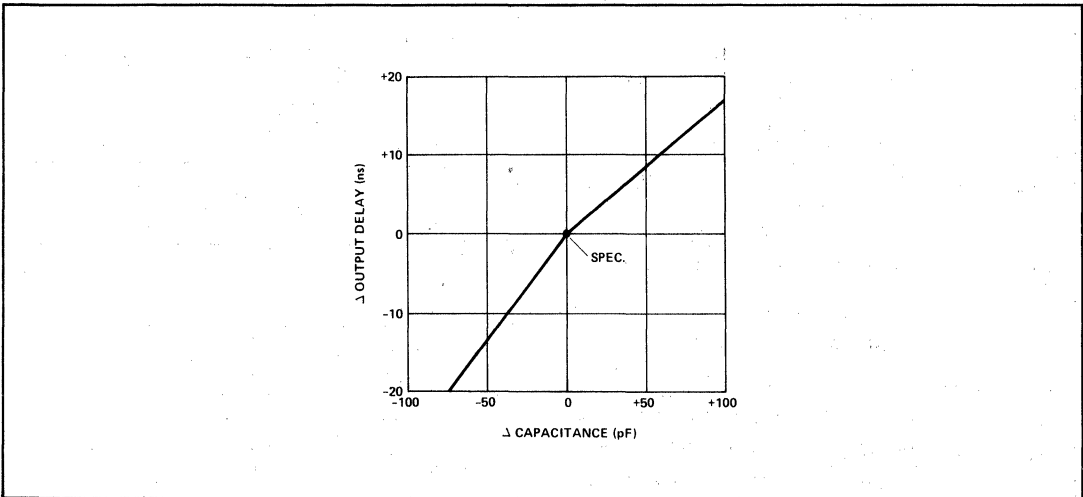
Symbol	Parameter	Min.	Max.	Unit	Test Conditions
$t_{CY}$	Clock Period	320	1350	ns	Notes 5, 6
$t_{CH}$	Clock High Pulse Width	140	$t_{CY}-90$	ns	
$t_{CL}$	Clock Low Pulse Width	90		ns	
$t_R, t_F$	Clock Rise and Fall Time		20	ns	
$t_{DTx}$	TxD Delay from Falling Edge of $\overline{TxC}$		1	$\mu$ s	
$f_{Tx}$	Transmitter Input Clock Frequency 1x Baud Rate 16x Baud Rate 64x Baud Rate	DC DC DC	64 310 615	kHz kHz kHz	
$t_{TPW}$	Transmitter Input Clock Pulse Width 1x Baud Rate 16x and 64x Baud Rate	12 1		$t_{CY}$ $t_{CY}$	
$t_{TPD}$	Transmitter Input Clock Pulse Delay 1x Baud Rate 16x and 64x Baud Rate	15 3		$t_{CY}$ $t_{CY}$	
$f_{Rx}$	Receiver Input Clock Frequency 1x Baud Rate 16x Baud Rate 64x Baud Rate	DC DC DC	64 310 615	kHz kHz kHz	
$t_{RPW}$	Receiver Input Clock Pulse Width 1x Baud Rate 16x and 64x Baud Rate	12 1		$t_{CY}$ $t_{CY}$	
$t_{RPD}$	Receiver Input Clock Pulse Delay 1x Baud Rate 16x and 64x Baud Rate	15 3		$t_{CY}$ $t_{CY}$	
$t_{TxRDY}$	TxRDY Pin Delay from Center of Last Bit		8	$t_{CY}$	Note 7
$t_{TxRDY\ CLEAR}$	TxRDY $\downarrow$ from Leading Edge of $\overline{WR}$		400	ns	Note 7
$t_{RxRDY}$	RxRDY Pin Delay from Center of Last Bit		26	$t_{CY}$	Note 7
$t_{RxRDY\ CLEAR}$	RxRDY $\downarrow$ from Leading Edge of $\overline{RD}$		400	ns	Note 7
$t_{IS}$	Internal SYNDET Delay from Rising Edge of $RxC$		26	$t_{CY}$	Note 7
$t_{ES}$	External SYNDET Set-Up Time After Rising Edge of $RxC$	18		$t_{CY}$	Note 7
$t_{TxEMPTY}$	TxEMPTY Delay from Center of Last Bit	20		$t_{CY}$	Note 7
$t_{WC}$	Control Delay from Rising Edge of WRITE ( $TxEn, \overline{DTR}, \overline{RTS}$ )	8		$t_{CY}$	Note 7
$t_{CR}$	Control to READ Set-Up Time ( $\overline{DSR}, \overline{CTS}$ )	20		$t_{CY}$	Note 7

### A.C. CHARACTERISTICS (Continued)

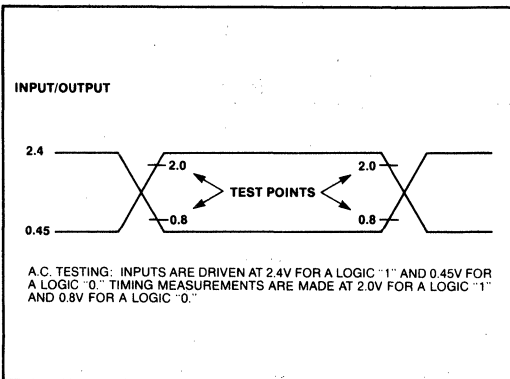
**NOTES:**

1. AC timings measured  $V_{OH} = 2.0$ ,  $V_{OL} = 0.8$ , and with load circuit of Figure 1.
2. Chip Select (CS) and Command/Data (C/D) are considered as Addresses.
3. Assumes that Address is valid before  $R_{DL}$ .
4. This recovery time is for Mode Initialization only. Write Data is allowed only when  $TxRDY = 1$ . Recovery Time between Writes for Asynchronous Mode is  $8 t_{CY}$  and for Synchronous Mode is  $16 t_{CY}$ .
5. The TxC and RxC frequencies have the following limitations with respect to CLK: For 1x Baud Rate,  $f_{TX}$  or  $f_{RX} \leq 1/(30 t_{CY})$ :  
For 16x and 64x Baud Rate,  $f_{TX}$  or  $f_{RX} \leq 1/(4.5 t_{CY})$ .
6. Reset Pulse Width =  $6 t_{CY}$  minimum; System Clock must be running during Reset.
7. Status update can have a maximum delay of 28 clock periods from the event affecting the status.

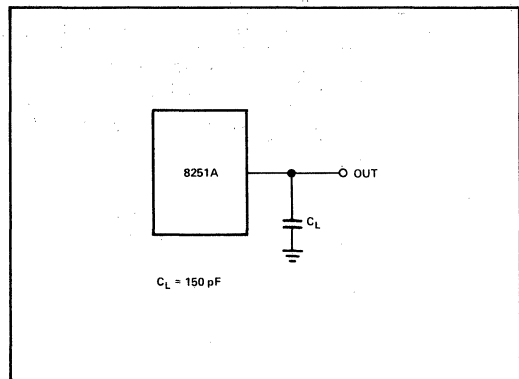
### TYPICAL $\Delta$ OUTPUT DELAY VS. $\Delta$ CAPACITANCE (pF)



### A.C. TESTING INPUT, OUTPUT WAVEFORM

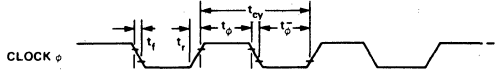


### A.C. TESTING LOAD CIRCUIT

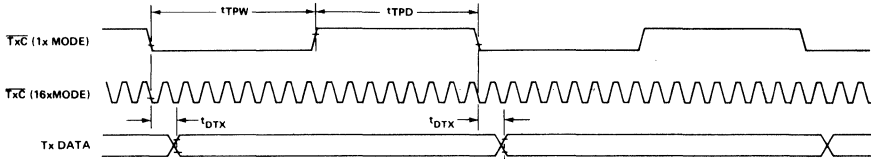


WAVEFORMS

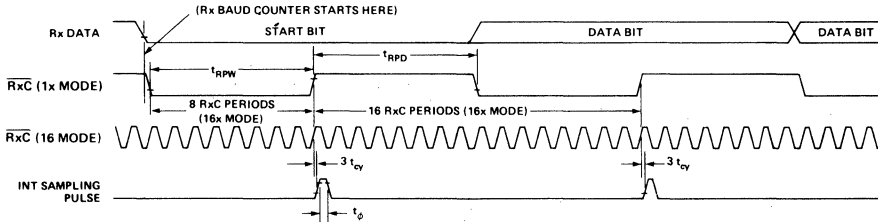
SYSTEM CLOCK INPUT



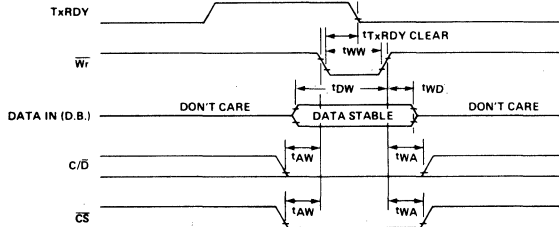
TRANSMITTER CLOCK AND DATA



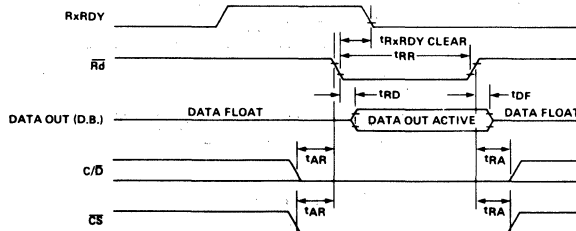
RECEIVER CLOCK AND DATA



WRITE DATA CYCLE (CPU → USART)

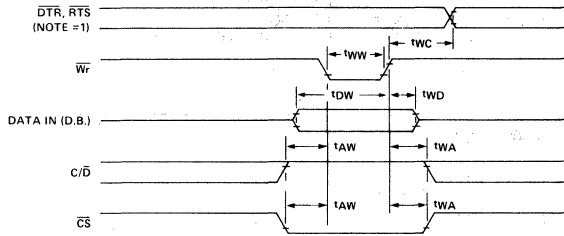


READ DATA CYCLE (CPU ← USART)

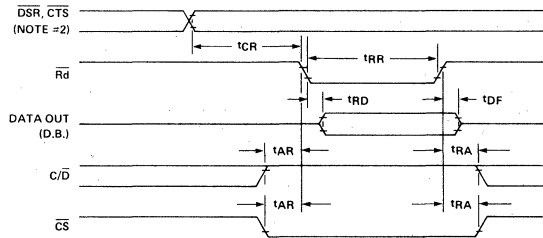


WAVEFORMS (Continued)

WRITE CONTROL OR OUTPUT PORT CYCLE (CPU → USART)

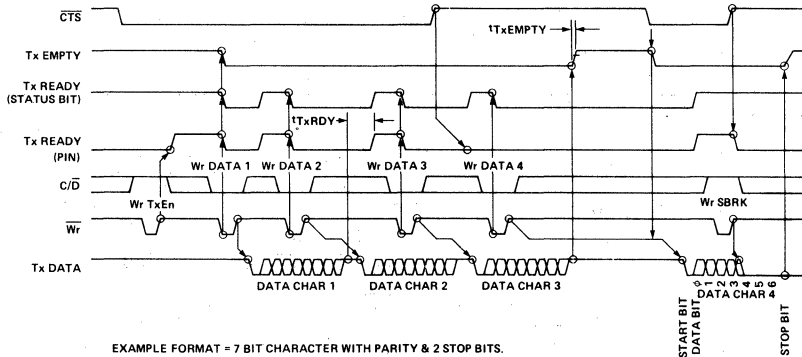


READ CONTROL OR INPUT PORT (CPU ← USART)



NOTE #1:  $T_{wc}$  INCLUDES THE RESPONSE TIMING OF A CONTROL BYTE.  
 NOTE #2:  $T_{CR}$  INCLUDES THE EFFECT OF CTS ON THE  $T_{xENBL}$  CIRCUITRY.

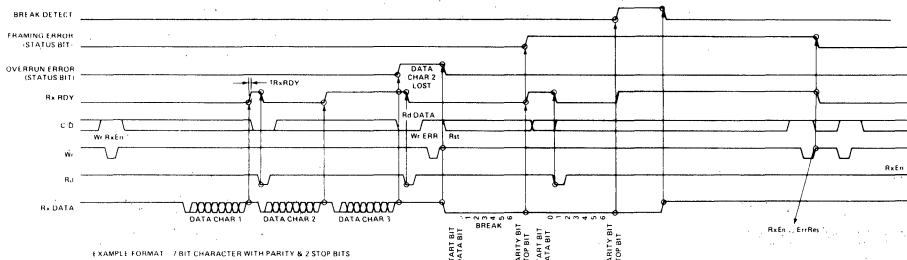
TRANSMITTER CONTROL AND FLAG TIMING (ASYNC MODE)



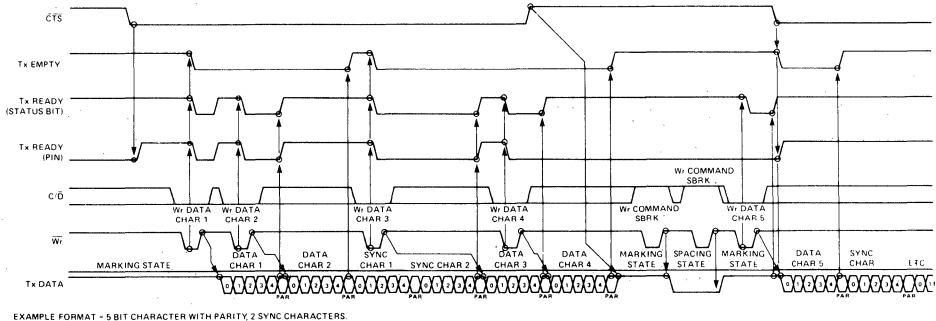
EXAMPLE FORMAT = 7 BIT CHARACTER WITH PARITY & 2 STOP BITS.

WAVEFORMS (Continued)

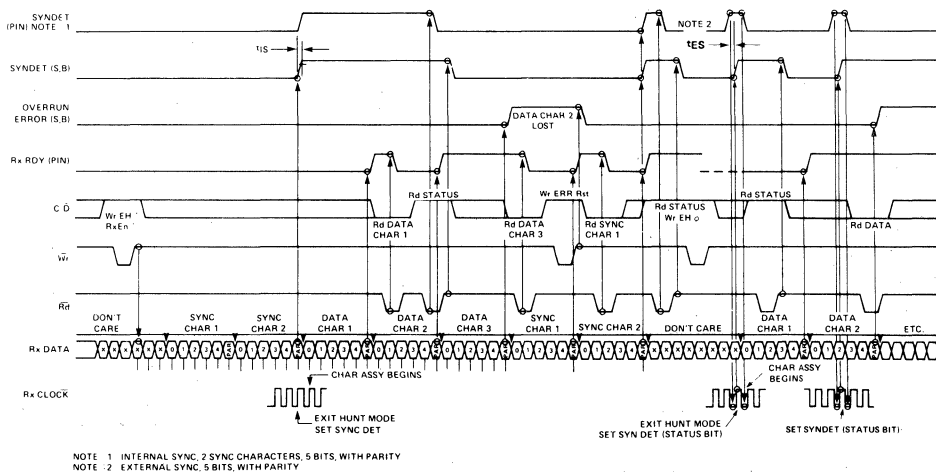
RECEIVER CONTROL AND FLAG TIMING (ASYNC MODE)



TRANSMITTER CONTROL AND FLAG TIMING (SYNC MODE)



RECEIVER CONTROL AND FLAG TIMING (SYNC MODE)



# 8256 MULTIFUNCTION UNIVERSAL ASYNCHRONOUS RECEIVER-TRANSMITTER (MUART)

- Programmable Serial Asynchronous Communications Interface for 5-, 6-, 7-, or 8-Bit Characters, 1, 1½, or 2 Stop Bits, and Parity Generation
  - On-Board Baud Rate Generator Programmable for 13 Common Baud Rates up to 19.2K Bits/second, or an External Baud Clock Maximum of 1M Bit/second
  - Five 8-Bit Programmable Timer/Counters; Four Can Be Cascaded to Two 16-Bit Timer/Counters
- Two 8-Bit Programmable Parallel I/O Ports; Port 1 Can Be Programmed for Port 2 Handshake Controls and Event Counter Inputs
  - Eight-Level Priority Interrupt Controller Programmable for 8085 or iAPX 86, iAPX 88 Systems and for Fully Nested Interrupt Capability
  - Programmable System Clock to 1 x, 2 x, 3 x, or 5 x 1.024 MHz

The Intel® 8256 Multifunction Universal Asynchronous Receiver-Transmitter (MUART) combines five commonly used functions into a single 40-pin device. It is designed to interface to the 8048, 8085A, iAPX 86, and iAPX 88 to perform serial communications, parallel I/O, timing, event counting, and priority interrupt functions. All of these functions are fully programmable through nine internal registers. In addition, the five timer/counters and two parallel I/O ports can be accessed directly by the microprocessor.

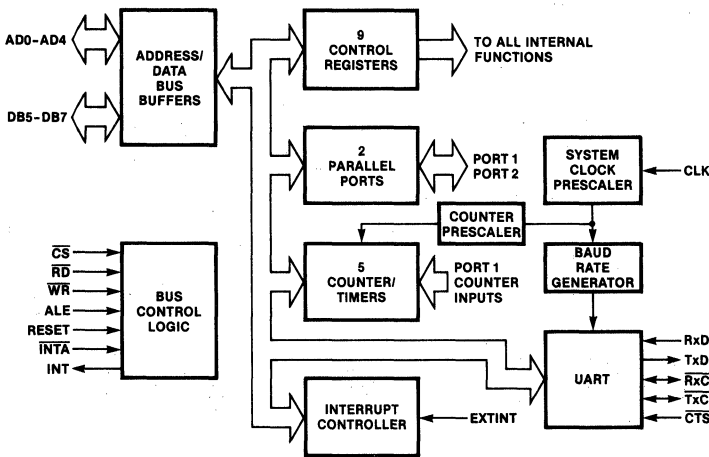


Figure 1. MUART Block Diagram

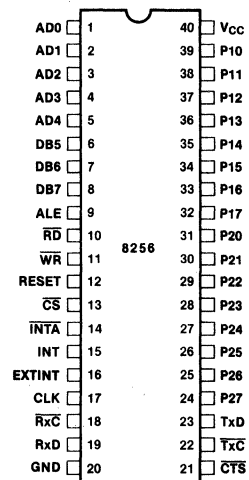


Figure 2. MUART Pin Configuration

Table 1. Pin Description

Symbol	Pin No.	Type	Name and Function
AD0-AD4 DB5-DB7	1-5 6-8	I/O	<b>Address/Data:</b> Three-State Address/Data lines which interface with the CPU lower 8-bit address/data bus. The 5-bit address is latched on the falling edge of ALE. In 8048 and 8085 mode, AD0-AD3 are used to select the proper register, while AD1-AD4 are used in 8086 and 8088 mode. The 8-bit bidirectional data bus is either written into or read from the chip depending on the latched $\overline{CS}$ and $\overline{RD}$ or $\overline{WR}$ .
ALE	9	I	<b>Address Latch Enable:</b> Latches the 5 address lines on AD0-AD4 and $\overline{CS}$ on the falling edge.
$\overline{RD}$	10	I	<b>Read Control:</b> When this signal is low, the previously selected register is enabled onto the data bus.
$\overline{WR}$	11	I	<b>Write Control:</b> When this signal is low, the value on the data bus is placed into the previously selected register.
RESET	12	I	Pulse provided by the CPU to initialize the system. The UART remains "idle" until it is reprogrammed by the CPU.
$\overline{CS}$	13	I	<b>Chip Select:</b> A low on this signal enables the UART. It is latched with the address on the falling edge of ALE, and $\overline{RD}$ and $\overline{WR}$ have no effect unless $\overline{CS}$ was latched low during the ALE cycle.
$\overline{INTA}$	14	I	<b>Interrupt Acknowledge:</b> If the MUART has been enabled to respond to interrupts, it puts an RST on the bus for the 8085 or a vector for the 8086. The bit in the interrupt register is reset when the interrupt is placed onto the bus.
INT	15	O	<b>Interrupt:</b> A high signals the CPU that the UART needs service.
EXTINT	16	I	<b>External Interrupt:</b> A high on this pin signals that an external device requests service. EXTINT must be held high until $\overline{INTA}$ or read interrupt occurs.
CLK	17	I	<b>System Clock:</b> This input provides an accurate timing source for the UART. It must be 1x, 2x, 3x, or 5x 1.024 MHz and is used by the baud rate generator and real time clocks.
$\overline{RxC}$	18	I/O	<b>Receive Clock:</b> If baud rate 0 is selected, this input clocks bits into $\overline{RxD}$ on the rising edge. If a baud rate from 1-0F <sub>16</sub> is selected, this output will provide a rising edge at the center of each received data bit. This output remains high during start, stop, and parity bits.
$\overline{RxD}$	19	I	<b>Receive Data:</b> Serial data input from the modem or terminal to the UART.
GND	20	PS	<b>Ground:</b> Power supply and logic ground reference.

Symbol	Pin No.	Type	Name and Function
V <sub>CC</sub>	40	PS	<b>Power:</b> +5V POWER supply.
P17-P10	32-39	I/O	<b>Parallel I/O Port 1:</b> Each pin can be programmed as an input or an output to perform general purpose I/O functions for the CPU under software control. In addition to general I/O, I/O Port 1 can be programmed to a variety of special functions for handshake control, counter inputs, and special communications functions.
P27-P20	24-31	I/O	<b>Parallel I/O Port 2:</b> Each nibble (4 bits) of this port can be either an input or an output. Also, this port can be used as a bidirectional 8-bit port using handshake lines in Port 1.
TxD	23	O	<b>Transmit Data:</b> This output carries the serial data to the terminal or modem from the MUART.
$\overline{TxC}$	22	I/O	<b>Transmit Clock:</b> If the baud rate is 0, this input clocks data out of the transmitter on the falling edge. If a baud rate of 1 or 2 is selected, this input permits the user to provide a 32x or 64x clock which is used for the receiver and transmitter. If the baud rate is 3-0F <sub>16</sub> , the internal transmitter clock is output. If 1½ stop bits are selected and characters are continuously transmitted, the internal baud rate generator will be reset at the end of the stop bits and the clock will have a small positive spike instead of a half clock. A high-to-low transition occurs at the beginning of each bit and a low-to-high transition at the center of each bit.
$\overline{CTS}$	21	I	<b>Clear to Send:</b> This input enables the serial transmitter. If $\overline{CTS}$ is low, any character in the transmitter buffer will be sent. A single negative-going pulse causes the transmission of a single previously loaded character out of the transmitter buffer. If this pulse occurs when the buffer is empty or during the transmission of a character up to 0.5 of the first stop bit, it will be ignored. If a baud rate from 1-0F <sub>16</sub> is selected, $\overline{CTS}$ must be low for at least 1/32 of a bit, or it will be ignored.



## FUNCTIONAL DESCRIPTION

The 8256 Multi-Function Universal Asynchronous Receiver-Transmitter (MUART) combines five commonly used functions onto a single 40-pin device. The MUART performs asynchronous serial communications, parallel I/O, timing, event counting, and interrupt control.

### Serial Communications

The serial communications portion of the MUART contains a full-duplex asynchronous receiver-transmitter (UART). A programmable baud rate generator is included on the MUART to permit a variety of operating speeds without external components. The UART can be programmed by the CPU for a variety of character sizes, parity generation and detection, error detection, and start/stop bit handling. The receiver checks the start and stop bits in the center of the bit, and a break halts the reception of data. The transmitter can send breaks and can be controlled by an external enable pin.

### Parallel I/O

The MUART includes 16 bits of general purpose parallel I/O. Eight bits (Port 1) can be individually changed from input to output or used for special I/O functions. The other eight bits (Port 2) can be used as nibbles (4 bits) or as bytes. These eight bits also include a handshaking capability using two pins on Port 1.

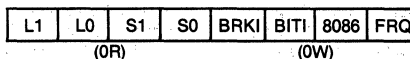
### Counter/Timers

There are five 8-bit counter/timers on the MUART. The timers can be programmed to use either a 1 kHz or 16 kHz clock generated from the system clock. Four of the 8-bit counter/timers can be cascaded to two 16-bit counter/timers, and one of the 8-bit counter/timers can be reset to its initial value by an external signal.

### Interrupts

An eight-level priority interrupt controller can be configured for fully nested or normal interrupt priority. Seven of the eight interrupts service functions on the MUART (counter/timers, UART), and one external interrupt is provided which can be used for a particular function or for chaining interrupt controllers or more MUARTs. The MUART will support 8085 and 8086/88 systems with direct interrupt vectoring, or the MUART can be polled to determine the cause of the interrupt.

## Command Register 1



### FRQ — Timer Frequency Select

This bit selects between two frequencies for the five timers. If FRQ = 0, the timer input frequency is 16 kHz (62.5 μs). If FRQ = 1, the timer input frequency is 1 kHz (1 ms). The selected clock frequency is shared by all the counter/timers enabled for timing; thus, all timers must run with the same time base.

### 8086 — 8086 Mode Enable

This bit selects between 8048/8085 mode and 8086/8088 mode. In 8085 mode (8086 = 0), A0 to A3 are used to address the internal registers, and an RST instruction is generated in response to the first  $\overline{INTA}$ . In 8086 mode (8086 = 1), A1 to A4 are used to address the internal registers, and A0 is used as an extra chip select (A0 must equal zero to be enabled). The response to  $\overline{INTA}$  is for 8086 interrupts where the first  $\overline{INTA}$  is ignored, and an interrupt vector (40<sub>16</sub> to 47<sub>16</sub>) is placed on the bus in response to the second  $\overline{INTA}$ .

### BITI — Interrupt on Bit Change

This bit disables the Timer 2 interrupt and enables an interrupt when a low-to-high transition occurs on pin 7 of Port 1 (pin 32).

### BRKI — Break-in Detect Enable

This bit enables the break-in detect feature. A break-in is detected when pin 6 of Port 1 (pin 33) is low during the first stop bit of a transmitted character. This could be used to detect a break-in condition by connecting the serial transmission line to pin 33. A break-in detect is OR-ed with break detect in bit 3 of the Status Register. If  $\overline{RxC}$  and  $\overline{TxC}$  are used for the serial bit rates, break-in cannot be detected.

### S0, S1 — Stop Bit Length

S1	S0	Stop Bit Length
0	0	1
0	1	1.5
1	0	2
1	1	0.75

If 0.75 stop bits is selected,  $\overline{CTS}$  becomes edge sensitive rather than level sensitive. A high-to-low transition of  $\overline{CTS}$  *immediately* initiates the transmission of the next character. A high-to-low transition will be ignored if the transmit buffer is empty, or if it occurs before 0.75 of the first stop bit. It will shorten the stop

Table 2. MUART Registers

Read Registers												Write Registers											
								8085 Mode:	AD3	AD2	AD1	AD0											
								8086 Mode:	AD4	AD3	AD2	AD1											
L1	L0	S1	S0	BRKI	BITI	8086	FRQ	0	0	0	0	L1	L0	S1	S0	BRKI	BITI	8086	FRQ				
Command 1												Command 1											
PEN	EP	C1	C0	B3	B2	B1	B0	0	0	0	1	PEN	EP	C1	C0	B3	B2	B1	B0				
Command 2												Command 2											
0	RxE	IAE	NIE	0	SBRK	TBRK	0	0	0	1	0	SET	RxE	IAE	NIE	END	SBRK	TBRK	RST				
Command 3												Command 3											
T35	T24	T5C	CT3	CT2	P2C2	P2C1	P2C0	0	0	1	1	T35	T24	T5C	CT3	CT2	P2C2	P2C1	P2C0				
Mode												Mode											
P17	P16	P15	P14	P13	P12	P11	P10	0	1	0	0	P17	P16	P15	P14	P13	P12	P11	P10				
Port 1 Control												Port 1 Control											
L7	L6	L5	L4	L3	L2	L1	L0	0	1	0	1	L7	L6	L5	L4	L3	L2	L1	L0				
Interrupt Enable												Set Interrupts											
D7	D6	D5	D4	D3	D2	D1	D0	0	1	1	0	L7	L6	L5	L4	L3	L2	L1	L0				
Interrupt Address												Reset Interrupts											
D7	D6	D5	D4	D3	D2	D1	D0	0	1	1	1	D7	D6	D5	D4	D3	D2	D1	D0				
Receiver Buffer												Transmitter Buffer											
D7	D6	D5	D4	D3	D2	D1	D0	1	0	0	0	D7	D6	D5	D4	D3	D2	D1	D0				
Port 1												Port 1											
D7	D6	D5	D4	D3	D2	D1	D0	1	0	0	1	D7	D6	D5	D4	D3	D2	D1	D0				
Port 2												Port 2											
D7	D6	D5	D4	D3	D2	D1	D0	1	0	1	0	D7	D6	D5	D4	D3	D2	D1	D0				
Timer 1												Timer 1											
D7	D6	D5	D4	D3	D2	D1	D0	1	0	1	1	D7	D6	D5	D4	D3	D2	D1	D0				
Timer 2												Timer 2											
D7	D6	D5	D4	D3	D2	D1	D0	1	1	0	0	D7	D6	D5	D4	D3	D2	D1	D0				
Timer 3												Timer 3											
D7	D6	D5	D4	D3	D2	D1	D0	1	1	0	1	D7	D6	D5	D4	D3	D2	D1	D0				
Timer 4												Timer 4											
D7	D6	D5	D4	D3	D2	D1	D0	1	1	1	0	D7	D6	D5	D4	D3	D2	D1	D0				
Timer 5												Timer 5											
INT	RBF	TBE	TRE	BD	PE	OE	FE	1	1	1	1	0	RS4	RS3	RS2	RS1	RS0	TME	DSC				
Status												Modification											

bit if it occurs after  $\frac{3}{4}$  of the stop bit has been sent. If CTS is high or low or a low-to-high transition occurs, the transmitter remains idle.

### L0, L1 — Character Length

L1	L0	Character Length
0	0	8
0	1	7
1	0	6
1	1	5

### Command Register 2

PEN	EP	C1	C0	B3	B2	B1	B0
(1R)				(1W)			

### B0, B1, B2, B3 — Baud Rate Select

B3	B2	B1	B0	Baud Rate	Sampling Rate
0	0	0	0	$\overline{\text{TxC}}, \overline{\text{RxC}}$	1
0	0	0	1	$\overline{\text{TxC}}/64$	64
0	0	1	0	$\overline{\text{TxC}}/32$	32
0	0	1	1	19200	32
0	1	0	0	9600	64
0	1	0	1	4800	64
0	1	1	0	2400	64
0	1	1	1	1200	64
1	0	0	0	600	64
1	0	0	1	300	64
1	0	1	0	200	64
1	0	1	1	150	64
1	1	0	0	110	64
1	1	0	1	100	64
1	1	1	0	75	64
1	1	1	1	50	64

If the baud rate is 0, then both the transmitter and receiver operate from separate external clocks. If the baud rate is 1 or 2, then both the transmitter and receiver divide the  $\overline{\text{TxC}}$  by 64 or 32, respectively.

### C0, C1 — System Clock Divider

C1	C0	Divider Ratio	System Clock Frequency
0	0	5	5.120 MHz
0	1	3	3.072 MHz
1	0	2	2.048 MHz
1	1	1	1.024 MHz

### EP — Even Parity

If parity is enabled, then even parity is enabled by a 1 and odd parity is enabled by a 0.

### PEN — Parity Enable

This enables parity detection and generation. The type of parity is determined by the EP bit.

### Command Register 3

SET	RxE	IAE	NIE	END	SBRK	TBRK	RST
(2R)				(2W)			

Command Register 3 is different from the first two registers because it has a bit set/reset capability. Writing a byte with bit 7 high sets any bits which were also high. Writing a byte with bit 7 low resets any bits which were high. If any bit 0–6 is low, no change occurs to that bit. When Command Register 3 is read, bits 0, 3, and 7 will always be zero.

### RST — Reset

If RST is set, the following events occur:

1. All bits in the Status Register except bits 4 and 5 are cleared, and bits 4 and 5 are set.
2. The Interrupt Enable, Interrupt Request, and Interrupt Service Registers are cleared.
3. The receiver and transmitter are reset. The transmitter goes idle (TxD is high), and the receiver enters start bit search mode.
4. If Port 2 is programmed for handshake mode,  $\overline{\text{IBF}}$  and  $\overline{\text{OBF}}$  are reset high.

RST does *not* alter ports, data registers or command registers, but it halts any operation in progress. RST is automatically cleared.

### TBRK — Transmit Break

This causes the transmitter data to be set low, and it stays low until TBRK is cleared. As long as break is active, data transfer from the Transmitter Buffer to the Transmitter Register will be inhibited.

### SBRK — Single Character Break

This causes the transmitter data to be set low for one character including start bit, data bits, parity bit, and stop bits. SBRK is automatically cleared when time for the last data bit has passed. It will start after the character in progress completes and will delay the next data transfer from the Transmitter Buffer to the Transmitter Register until TxD returns to an idle (marking) state. If both TBRK and SBRK are set, break will be sent as long as TBRK is set, but SBRK will be cleared after one character time of break. If SBRK is set again, it remains set for another character. The user can send a definite number of break characters in this manner by clearing TBRK after setting SBRK for the last character time.

### END — End of Interrupt

If fully nested interrupt mode is selected, this bit resets the currently served interrupt level in the Interrupt Service Register. *This command must occur at the end of each interrupt service routine during fully nested*

**interrupt mode.** END is automatically cleared when the Interrupt Service Register (internal) is cleared. See the NIE description for more information on nested interrupt servicing. END is ignored if nested interrupts are not enabled.

**NIE — Nested Interrupt Enable**

This bit enables fully nested interrupts. In this mode, the service routine for a lower priority interrupt can be interrupted by a request from a higher priority task.

In fully nested interrupt mode,  $\overline{INTA}$  or reading the Interrupt Address Register resets the highest priority interrupt bit in the Interrupt Register (internal), sets the corresponding bit in the Interrupt Service Register (internal), and resets INT. If an interrupt of higher priority than the currently served interrupt is requested or the END bit is set while another interrupt request is pending, the INT line will go high again. If an interrupt service routine is interrupted by an interrupt of higher priority, two or more bits in the Interrupt Service Register will be set.

If NIE is low, interrupt priority is used only when two interrupts occur at the same time. INT will be high as long as the CPU has not responded to all the interrupts in the Interrupt Register.

**IAE — Interrupt Acknowledge Enable**

This bit enables an automatic response to  $\overline{INTA}$ . The particular response is determined by the 8086 bit in Command Register 1.

**RxE — Receiver Enable**

This bit enables the serial receiver. The Receiver Buffer and all receiver status information will be disabled except for the break detect status.

**SET — Bit Set/Reset**

If this bit is high during a write to Command Register 3, then any bit marked by a high will be set. If this bit is low, then any bit marked by a high will be cleared.

**Mode Register**

T35	T24	T5C	CT3	CT2	P2C2	P2C1	P2C0
(3R)			(3W)				

**P2C2, P2C1, P2C0 — Port 2 Control**

P2C2	P2C1	P2C0	Mode	Direction	
				Upper	Lower
0	0	0	nibble	input	input
0	0	1	nibble	input	output
0	1	0	nibble	output	input
0	1	1	nibble	output	output
1	0	0	byte handshake	input	
1	0	1	byte handshake	output	
1	1	0	DO NOT USE		
1	1	1	test		

If test mode is selected and BRG of Port 1 Control Register is set, then the output from the internal baud rate generator is placed on pin 4 of Port 1 (pin 35).

**CT2, CT3 — Counter/Timer Mode**

If CT2 or CT3 are high, then counter/timer 2 or 3 respectively is configured as an event counter on pin 2 or 3 respectively of Port 1 (pins 37 or 36). The event counter decrements the count by one on each low-to-high transition of the external input. If CT2 or CT3 is low, then the respective counter/timer is configured as a timer and the Port 1 pins are used for parallel I/O.

**T5C — Timer 5 Control**

If T5C is set, then Timer 5 can be preset and started by an external signal. Writing to the Timer 5 Register loads the Timer 5 Save Register and stops the timer. A high-to-low transition on pin 5 of Port 1 (pin 34) loads the timer with the saved value and starts the timer. The next high-to-low transition on pin 5 retriggers the timer by reloading it with the initial value and continues timing.

When the timer reaches zero it issues an interrupt request, disables its interrupt level and continues counting. A subsequent high-to-low transition on pin 5 resets Timer 5 to its initial value. For another timer interrupt, the Timer 5 interrupt enable bit must be set again.

**T35, T24 — Cascade Timers**

These two bits cascade Timers 3 and 5 or 2 and 4. Timers 2 and 3 are the lower bytes, while Timers 4 and 5 are the upper bytes. If T5C is set, then both Timers 3 and 5 can be preset and started by an external pulse. When a high-to-low transition occurs, Timer 5 is preset to its saved value, but Timer 3 is always preset to all ones. If either CT2 or CT3 is set, then the corresponding timer pair is a 16-bit event counter.

**Port 1 Control Register**

P17	P16	P15	P14	P13	P12	P11	P10
(4R)				(4W)			

Each bit in the Port 1 Control Register configures the direction of the corresponding pin. If the bit is high, the pin is an output, and if it is low the pin is an input. Every Port 1 pin has another function which is controlled by other registers. If that special function is disabled, the pin functions as a general I/O pin as specified by this register. The special functions for each pin are described below.

**Port 10, 11 — Handshake Control**

If byte handshake control is enabled for Port 2 by the Mode Register, then Port 10 is programmed as STB/ACK handshake control input and Port 11 is programmed as IBF/OBF handshake control output.

If byte handshake mode is enabled for output on Port 2, OBF indicates that a character has been loaded into the

Port 2 output buffer. When an external device reads the data, it acknowledges this operation by driving ACK low. OBF is set low by writing to Port 2 and is reset high by ACK.

If byte handshake mode is enabled for input on Port 2, STB is an input to the MUART to latch the data into Port 2. After the data is latched, IBF is driven low. IBF is reset high when Port 2 is read.

**Port 12, 13 — Counter 2, 3 Input**

If Timer 2 or Timer 3 is programmed as an event counter by the mode register, then Port 12 or 13 is the counter input for Event Counter 2 or 3, respectively.

**Port 14 — Baud Rate Generator Output Clock**

If test mode is enabled by the Mode Register and Command Register 2 baud rate select is greater than 2, then Port 14 is an output from the internal baud rate generator.

**Port 15 — Timer 5 Trigger**

If T5C is set in the Mode Register enabling a re-triggerable timer, then Port 15 is the input which starts and reloads Timer 5.

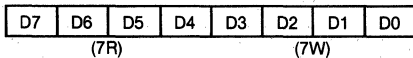
**Port 16 — Break-in Detect**

If break-in detect is enabled by BRKI in Command Register 1, then this input is used to sense a break-in. If Port 16 is low while the serial transmitter is sending the last stop bit, then a break-in condition is signaled.

**Port 17 — Port Interrupt Source**

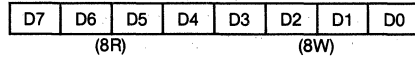
If BITI in Command Register 1 is set, then a low-to-high transition on Port 17 generates an interrupt request on priority level 1.

**Receiver and Transmitter Buffer**



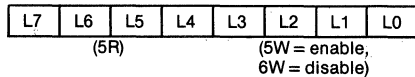
Both the transmitter and the receiver in the MUART are fully double buffered. The Receiver Buffer full flag is cleared when the character is read. If the character is not read before the next character's first stop bit, then an overrun error is generated. Bytes written to the Transmitter Buffer are held until the Transmitter Register (internal) is empty. If the Transmitter Register is empty, the byte is transferred immediately and the Transmitter Buffer empty flag is set. If a serial character length is less than 8 bits, then the unused most significant bits are set to zero on a read and are ignored on a write.

**Port 1**



Writing to Port 1 sets the data in the Port 1 output latch. Writing to an input pin does not affect the pin, but the data is stored and will be output if the direction of the pin is changed later. If the pin is used as a control signal, the pin will not be affected, but the data is stored. Reading Port 1 transfers the data in Port 1 onto the data bus. Reading an output pin or a control pin puts the data in the output latch (not the control signal) onto the data bus.

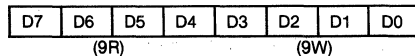
**Interrupt Enable Register**



Interrupts are enabled by writing to the Set Interrupts Register (5W). Interrupts are disabled by writing to the Reset Interrupts Register (6W). Each bit set by the Set Interrupts Register (5W) will enable that level interrupt, and each bit set in the Reset Interrupts Register (6W) will disable that level interrupt. The user can determine which interrupts are enabled by reading the Interrupt Enable Register (5R).

Priority	Source
Highest L0	Timer 1
L1	Timer 2 or Port Interrupt
L2	External Interrupt (EXTINT)
L3	Timer 3 or Timers 3 & 5
L4	Receiver Interrupt
L5	Transmitter Interrupt
L6	Timer 4 or Timers 2 & 4
Lowest L7	Timer 5 or Port 2 Handshaking

**Port 2**



Writing to Port 2 sets the data in the Port 2 output latch. Writing to an input pin does not affect the pin, but it does store the data in the latch. Reading Port 2 puts the input pins onto the bus or the contents of the output latch for output pins.

**Timer 1-5**

D7	D6	D5	D4	D3	D2	D1	D0
(0A <sub>16</sub> -0E <sub>16</sub> R)				(0A <sub>16</sub> -0E <sub>16</sub> W)			

Reading Timer N puts the contents of the timer onto the data bus. If the counter changes while  $\overline{RD}$  is low, the value on the data bus will not change. If two timers are cascaded, reading the high order byte will cause the low order byte to be latched. Reading the low order byte will unlatch them both. Writing to either timer or de-cascading them also clears the latch condition. Writing to a timer sets the starting value of that timer. If two timers are cascaded, writing to the high order byte presets the low order byte to all ones. Loading only the high order byte with a value of X leads to a count of  $X \cdot 256 + 255$ . Timers count down continuously. If the interrupt is enabled, it occurs when the counter changes from 1 to 0. When the interrupt is set in the Interrupt Register, interrupts are disabled in the Interrupt Mask Register.

**Status Register**

INT	RBF	TBE	TRE	BD	PE	OE	FE
(0F <sub>16</sub> R)							

**FE — Framing Error, Transmission Mode**

If transmission mode is disabled (in Modification Register), then FE indicates a framing error. A framing error is detected during the *first* stop bit. The error is reset by reading the Status Register or by a chip reset. A framing error does not inhibit the loading of the Receiver Buffer. If RxD remains low, the receiver will assemble the next character. The false stop bit is treated as the next start bit, and no high-to-low transition on RxD is required to synchronize the receiver.

If transmission mode is enabled, then this bit is used to suggest the transmitter was sending. FE will be high if the transmitter is active during the reception of the parity bit (or last data bit for no-parity). It is reset if the transmitter is not active or by a chip reset. The bit is intended to imply that the received character is from the transmitter in half-duplex systems.

**OE — Overrun Error**

If the user does not read the character in the Receiver Buffer before the next character is received and transferred to this register, then the OE bit is set. The OE flag is set during the reception of the first stop bit and is cleared when the Status Register is read or when a chip reset occurs.

**PE — Parity Error**

A parity error is set during the first stop bit and is reset by reading the Status Register or by a chip reset.

**BD — Break Detect, Break-in Detect**

If BRKI in Command Register 1 is set to enable break-in detect, then BD indicates a break-in condition. If Port 16 is low during the transmission of the last stop bit, then BD will be set near the end of the last stop bit. Break-in detect can only be detected if the internal baud rate generator is used. Break-in remains set until the Status Register is read or the chip is reset.

If BRKI is low, then BD indicates a break condition on the receiver. BD is set when the first stop bit of a break is sampled and will remain set until the Status Register is read or the chip is reset. The receiver will remain idle until the next high-to-low transition on RxD. A detected break inhibits the loading of the Receiver Buffer.

**TRE — Transmitter Register Empty**

This status bit indicates that the Transmitter Register is busy. It is set by a chip reset and when the last stop bit has left the transmitter. It is reset when a character is loaded into the Transmitter Register. If  $\overline{CTS}$  is low, the Transmitter Register will be loaded during the transmission of the start bit. If  $\overline{CTS}$  is high at the end of a character, TRE will remain high and no character will be loaded into the Transmitter Register until  $\overline{CTS}$  goes low. If the transmitter was inactive before a character is loaded into the Transmitter Buffer, the Transmitter Register will be empty temporarily while the buffer is full. However, the data in the buffer will be transferred to the transmitter register immediately and TRE will be cleared while TBE is set.

**TBE — Transmitter Buffer Empty**

TBE indicates the Transmitter Buffer is empty and is ready to accept a character. TBE is set by a chip reset or the transfer of data to the Transmitter Register and is cleared when a character is written to the transmitter buffer.

**RBF — Receiver Buffer Full**

RBF is set when the Receiver Buffer has been loaded with a new character during the sampling of the first stop bit. RBF is cleared by reading the receiver buffer or by a chip reset.

**INT — Interrupt Pending**

The INT bit reflects the state of the INT pin (pin 15) and indicates an interrupt is pending in the Interrupt Register. It is reset by  $\overline{INTA}$  or by reading the Interrupt Address Register if only one interrupt is pending and by a chip reset.

FE, OE, PE, RBF, and break detect all generate a level 4 interrupt when the receiver samples the first stop bit. TRE, TBE, and break-in detect generate a level 5 interrupt. TRE generates an interrupt when TBE is set and the Transmitter Register finishes transmitting. The

break-in detect interrupt is issued at the same time as TBE or TRE.

of the bit (sample time = 16). The receiver sample time can be modified only if the receiver is *not* clocked by RxC.

**Modification Register**

0	RS4	RS3	RS2	RS1	RS0	TME	DSC
(OF <sub>16</sub> W)							

**DSC — Disable Start Bit Check**

DSC disables the receiver's start bit check. In this state the receiver will not be reset if RxD is not low at the center of the start bit. This function is disabled by a chip reset.

**TME — Transmission Mode Enable**

TME enables transmission mode and disables framing error detection. A chip reset disables transmission mode and enables framing error detection.

**RS0, RS1, RS2, RS3, RS4 — Receiver Sample Time**

The number in RS<sub>n</sub> alters when the receiver samples RxD. A chip reset sets this value to 0 which is the center

RS4	RS3	RS2	RS1	Sample Time	
				RS0 = 0	RS0 = 1
0	1	1	1	2	1
0	1	1	0	4	3
0	1	0	1	6	5
0	1	0	0	8	7
0	0	1	1	10	9
0	0	1	0	12	11
0	0	0	1	14	13
0	0	0	0	16	15
1	1	1	1	18	17
1	1	1	0	20	19
1	1	0	1	22	21
1	1	0	0	24	23
1	0	1	1	26	25
1	0	1	0	28	27
1	0	0	1	30	29
1	0	0	0	32	31



# 8273, 8273-4, 8273-8

## PROGRAMMABLE HDLC/SDLC PROTOCOL CONTROLLER

- CCITT X.25 Compatible
- HDLC/SDLC Compatible
- Full Duplex, Half Duplex, or Loop SDLC Operation
- Up to 64K Baud Synchronous Transfers
- Automatic FCS (CRC) Generation and Checking
- Up to 9.6K Baud with On-Board Phase Locked Loop
- Programmable NRZI Encode/Decode
- Two User Programmable Modem Control Ports
- Digital Phase Locked Loop Clock Recovery
- Minimum CPU Overhead
- Fully Compatible with 8048/8080/8085/8088/8086 CPUs
- Single +5V Supply

The Intel® 8273 Programmable HDLC/SDLC Protocol Controller is a dedicated device designed to support the ISO/CCITT's HDLC and IBM's SDLC communication line protocols. It is fully compatible with Intel's new high performance microcomputer systems such as the MCS-88/86™. A frame level command set is achieved by a unique microprogrammed dual processor chip architecture. The processing capability supported by the 8273 relieves the system CPU of the low level real-time tasks normally associated with controllers.

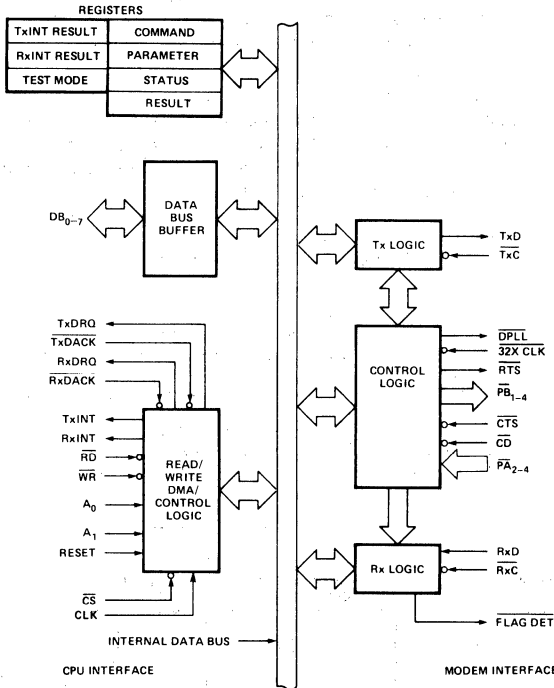


Figure 1. Block Diagram

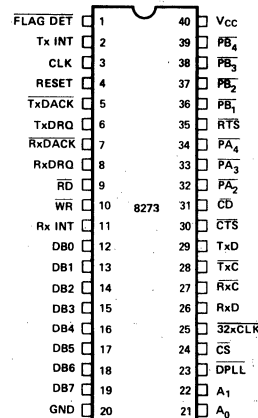


Figure 2. Pin Configuration



## A BRIEF DESCRIPTION OF HDLC/SDLC PROTOCOLS

### General

The High Level Data Link Control (HDLC) is a standard communication link protocol established by International Standards Organization (ISO). HDLC is the discipline used to implement ISO X.25 packet switching systems.

The Synchronous Data Link Control (SDLC) is an IBM communication link protocol used to implement the System Network Architecture (SNA). Both the protocols are bit oriented, code independent, and ideal for full duplex communication. Some common applications include terminal to terminal, terminal to CPU, CPU to CPU, satellite communication, packet switching and other high speed data links. In systems which require expensive cabling and interconnect hardware, any of the two protocols could be used to simplify interfacing (by going serial), thereby reducing interconnect hardware costs. Since both the protocols are speed independent, reducing interconnect hardware could become an important application.

### Network

In both the HDLC and SDLC line protocols, according to a pre-assigned hierarchy, a PRIMARY (Control) STATION controls the overall network (data link) and issues commands to the SECONDARY (Slave) STATIONS. The latter comply with instructions and respond by sending appropriate RESPONSES. Whenever a transmitting station must end transmission prematurely it sends an ABORT character. Upon detecting an abort character, a receiving station ignores the transmission block called a FRAME. Time fill between frames can be accomplished by transmitting either continuous frame preambles called FLAGS or an abort character. A time fill within a frame is not permitted. Whenever a station receives a string of more than fifteen consecutive ones, the station goes into an IDLE state.

### Frames

A single communication element is called a FRAME which can be used for both Link Control and data transfer purposes. The elements of a frame are the beginning eight bit FLAG (F) consisting of one zero, six ones, and a zero, an eight bit ADDRESS FIELD (A), an eight bit CONTROL FIELD (C), a variable (N-bit) INFORMATION FIELD (I), a sixteen bit FRAME CHECK SEQUENCE (FCS), and an eight bit end FLAG (F), having the same bit pattern as the beginning flag. In HDLC the Address (A) and Control (C) bytes are extendable. The HDLC and the SDLC use three

types of frames; an Information Frame is used to transfer data, a Supervisory Frame is used for control purposes, and a Non-sequenced Frame is used for initialization and control of the secondary stations.

### Frame Characteristics

An important characteristic of a frame is that its contents are made code transparent by use of a zero bit insertion and deletion technique. Thus, the user can adopt any format or code suitable for his system — it may even be a computer word length or a "memory dump". The frame is bit oriented that is, bits, not characters in each field, have specific meanings. The Frame Check Sequence (FCS) is an error detection scheme similar to the Cyclic Redundancy Checkword (CRC) widely used in magnetic disk storage devices. The Command and Response information frames contain sequence numbers in the control fields identifying the sent and received frames. The sequence numbers are used in Error Recovery Procedures (ERP) and as implicit acknowledgement of frame communication, enhancing the true full-duplex nature of the HDLC/SDLC protocols.

In contrast, BISYNC is basically half-duplex (two way alternate) because of necessity to transmit immediate acknowledgement frames. HDLC/SDLC therefore saves propagation delay times and have a potential of twice the throughput rate of BISYNC.

It is possible to use HDLC or SDLC over half duplex lines but there is a corresponding loss in throughput because both are primarily designed for full-duplex communication. As in any synchronous system, the bit rate is determined by the clock bits supplied by the modem, protocols themselves are speed independent.

A byproduct of the use of zero-bit insertion-deletion technique is the non-return-to-zero invert (NRZI) data transmission/reception compatibility. The latter allows HDLC/SDLC protocols to be used with asynchronous data communication hardware in which the clocks are derived from the NRZI encoded data.

### References

- IBM Synchronous Data Link Control General Information*, IBM, GA 27-3093-1.
- Standard Network Access Protocol Specification*, DATAPAC, Trans-Canada Telephone System CCG111 Recommendation X.25, ISO/CCITT March 2, 1976.
- IBM 3650 Retail Store System Loop Interface OEM Information*, IBM, GA 27-3098-0
- Guidebook to Data Communications*, Training Manual, Hewlett-Packard 5955-1715
- IBM Introduction to Teleprocessing*, IBM, GC 20-8095-02
- System Network Architecture, Technical Overview*, IBM, GA 27-3102
- System Network Architecture Format and Protocol*, IBM GA 27-3112

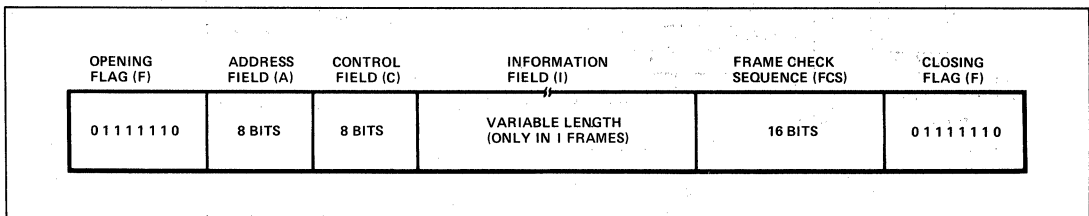


Figure 3. Frame Format

**Table 1. Pin Description**

Symbol	Pin No.	Type	Name and Function
V <sub>CC</sub>	40		<b>Power Supply:</b> +5V Supply.
GND	20		<b>Ground:</b> Ground.
RESET	4	I	<b>Reset:</b> A high signal on this pin will force the 8273 to an idle state. The 8273 will remain idle until a command is issued by the CPU. The modem interface output signals are forced high. Reset must be true for a minimum of 10 TCY.
$\overline{\text{CS}}$	24	I	<b>Chip Select:</b> The RD and WR inputs are enabled by the chip select input.
DB <sub>7</sub> -DB <sub>0</sub>	19-12	I/O	<b>Data Bus:</b> The Data Bus lines are bi-directional three-state lines which interface with the system Data Bus.
$\overline{\text{WR}}$	10	I	<b>Write Input:</b> The Write signal is used to control the transfer of either a command or data from CPU to the 8273.
RD	9	I	<b>Read Input:</b> The Read signal is used to control the transfer of either a data byte or a status word from the 8273 to the CPU.
TxINT	2	O	<b>Transmitter Interrupt:</b> The Transmitter interrupt signal indicates that the transmitter logic requires service.
RxINT	11	O	<b>Receiver Interrupt:</b> The Receiver interrupt signal indicates that the Receiver logic requires service.
TxDRQ	6	O	<b>Transmitter Data Request:</b> Requests a transfer of data between memory and the 8273 for a transmit operation.
RxRDQ	8	O	<b>Receiver DMA Request:</b> Requests a transfer of data between the 8273 and memory for a receive operation.
$\overline{\text{TxDACK}}$	5	I	<b>Transmitter DMA Acknowledge:</b> The Transmitter DMA acknowledge signal notifies the 8273 that the TxDMA cycle has been granted.
$\overline{\text{RxDACK}}$	7	I	<b>Receiver DMA Acknowledge:</b> The Receiver DMA acknowledge signal notifies the 8273 that the RxDMA cycle has been granted.
A <sub>1</sub> -A <sub>0</sub>	22-21	I	<b>Address:</b> These two lines are CPU Interface Register Select lines.
TxD	29	O	<b>Transmitter Data:</b> This line transmits the serial data to the communication channel.
$\overline{\text{TxC}}$	28	I	<b>Transmitter Clock:</b> The transmitter clock is used to synchronize the transmit data.
RxD	26	I	<b>Receiver Data:</b> This line receives serial data from the communication channel.
$\overline{\text{RxC}}$	27	I	<b>Receiver Clock:</b> The Receiver Clock is used to synchronize the receive data.

Symbol	Pin No.	Type	Name and Function
32X CLK	25	I	<b>32X Clock:</b> The 32X clock is used to provide clock recovery when an asynchronous modem is used. In loop configuration the loop station can run without an accurate 1X clock by using the 32X CLK in conjunction with the DPLL output. (This pin must be grounded when not used.)
$\overline{\text{DPLL}}$	23	O	<b>Digital Phase Locked Loop:</b> Digital Phase Locked Loop output can be tied to RxCan and/or TxCan when 1X clock is not available. DPLL is used with 32X CLK.
$\overline{\text{FLAG DET}}$	1	O	<b>Flag Detect:</b> Flag Detect signals that a flag (01111110) has been received by an active receiver.
$\overline{\text{RTS}}$	35	O	<b>Request to Send:</b> Request to Send signals that the 8273 is ready to transmit data.
$\overline{\text{CTS}}$	30	I	<b>Clear to Send:</b> Clear to Send signals that the modem is ready to accept data from the 8273.
$\overline{\text{CD}}$	31	I	<b>Carrier Detect:</b> Carrier Detect signals that the line transmission has started and the 8273 may begin to sample data on RxD line.
$\overline{\text{PA}}_{2-4}$	32-34	I	<b>General purpose input ports:</b> The logic levels on these lines can be Read by the CPU through the Data Bus Buffer.
$\overline{\text{PB}}_{1-4}$	36-39	O	<b>General purpose output ports:</b> The CPU can write these output lines through Data Bus Buffer.
CLK	3	I	<b>Clock:</b> A square wave TTL clock.

## FUNCTIONAL DESCRIPTION

### General

The Intel® 8273 HDLC/SDLC controller is a microcomputer peripheral device which supports the International Standards Organization (ISO) High Level Data Link Control (HDLC), and IBM Synchronous Data Link Control (SDLC) communications protocols. This controller minimizes CPU software by supporting a comprehensive frame-level instruction set and by hardware implementation of the low level tasks associated with frame assembly/disassembly and data integrity. The 8273 can be used in either synchronous or asynchronous applications.

In asynchronous applications the data can be programmed to be encoded/decoded in NRZI code. The clock is derived from the NRZI data using a digital phase locked loop. The data transparency is achieved by using a zero-bit insertion/deletion technique. The frames are automatically checked for errors during reception by verifying the Frame Check Sequence (FCS); the FCS is automatically generated and appended before the final flag in transmit.

The 8273 recognizes and can generate flags (01111110), Abort, Idle, and GA (EOP) characters.

The 8273 can assume either a primary (control) or a secondary (slave) role. It can therefore be readily implemented in an SDLC loop configuration as typified by the IBM 3650 Retail Store System by programming the 8273 into a one-bit delay mode. In such a configuration, a two wire pair can be effectively used for data transfer between controllers and loop stations. The digital phase locked loop output pin can be used by the loop station without the presence of an accurate Tx clock.

**CPU Interface**

The CPU interface is optimized for the MCS-80/85™ bus with an 8257 DMA controller. However, the interface is flexible, and allows either DMA or non-DMA data transfers, interrupt or non-interrupt driven. It further allows maximum line utilization by providing early interrupt mechanism for buffered (only the information field can be transferred to memory) Tx command overlapping. It also provides separate Rx and Tx interrupt output channels for efficient operation. The 8273 keeps the interrupt request active until all the associated interrupt results have been read.

The CPU utilizes the CPU interface to specify commands and transfer data. It consists of seven registers addressed via CS, A1, A0, RD and WR signals and two independent data registers for receive data and transmit data. A1, A0 are generally derived from two low order bits of the address bus. If an 8080 based CPU is utilized, the RD and WR signals may be driven by the 8228 I/OR and I/OW. The table shows the seven register select decoding:

A1	A0	TxDACK	RxDACK	CS	RD	WR	Register
0	0	1	1	0	1	0	Command
0	0	1	1	0	0	1	Status
0	1	1	1	0	1	0	Parameter
0	1	1	1	0	0	1	Result
1	0	1	1	0	1	0	Reset
1	0	1	1	0	0	1	TxINT Result
1	1	1	1	0	1	0	—
1	1	1	1	0	0	1	RxINT Result
X	X	0	1	1	1	0	Transmit Data
X	X	1	0	1	0	1	Receive Data

**Register Description**

**Command**

Operations are initiated by writing an appropriate command in the Command Register.

**Parameter**

Parameters of commands that require additional information are written to this register.

**Result**

Contains an immediate result describing an outcome of an executed command.

**Transmit Interrupt Result**

Contains the outcome of 8273 transmit operation (good/bad completion).

**Receive Interrupt Result**

Contains the outcome of 8273 receive operation (good/bad completion), followed by additional results which detail the reason for interrupt.

**Status**

The status register reflects the state of the 8273 CPU Interface.

**DMA Data Transfers**

The 8273 CPU interface supports two independent data interfaces: receive data and transmit data. At high data transmission speeds the data transfer rate of the 8273 is great enough to justify the use of direct memory access (DMA) for the data transfers. When the 8273 is configured in DMA mode, the elements of the DMA interfaces are:

**TxDQ: Transmit DMA Request**

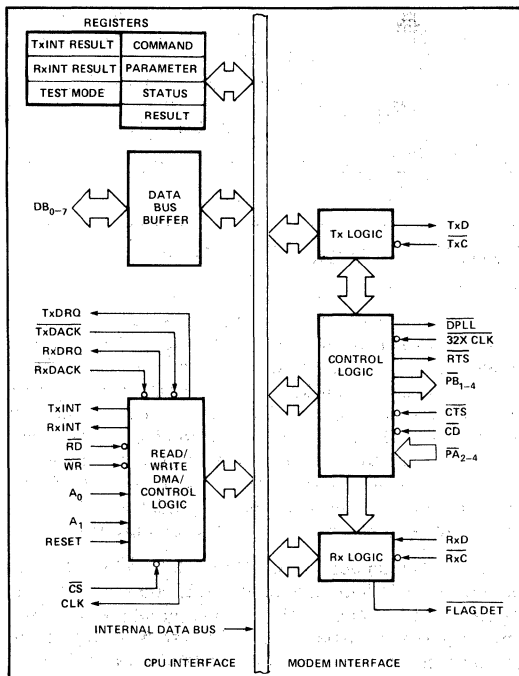
Requests a transfer of data between memory and the 8273 for a transmit operation.

**TxDACK: Transmit DMA Acknowledge**

The TxDACK signal notifies the 8273 that a transmit DMA cycle has been granted. It is also used with WR to transfer data to the 8273 in non-DMA mode. Note: RD must not be asserted while TxDACK is active.

**RxDQ: Receive DMA Request**

Requests a transfer of data between the 8273 and memory for a receive operation.



**Figure 4. 8273 Block Diagram Showing CPU Interface Functions**

**RxDACK: Receive DMA Acknowledge**

The RxDACK signal notifies the 8273 that a receive DMA cycle has been granted. It is also used with RD to read data from the 8273 in non-DMA mode. Note: WR must not be asserted while RxDACK is active.

**RD, WR: Read, Write**

The RD and WR signals are used to specify the direction of the data transfer.

DMA transfers require the use of a DMA controller such as the Intel 8257. The function of the DMA controller is to provide sequential addresses and timing for the transfer, at a starting address determined by the CPU. Counting of data block lengths is performed by the 8273.

To request a DMA transfer the 8273 raises the appropriate DMA REQUEST. DMA ACKNOWLEDGE and READ enables DMA data onto the bus (independently of CHIP SELECT). DMA ACKNOWLEDGE and WRITE transfers DMA data to the 8273 (independent of CHIP SELECT).

It is also possible to configure the 8273 in the non-DMA data transfer mode. In this mode the CPU module must pass data to the 8273 in response to non-DMA data requests indicated by the status word.

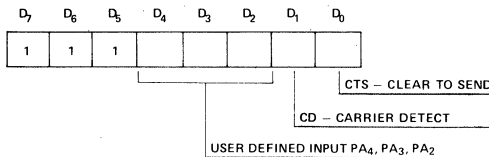
**Modem Interface**

The 8273 Modem interface provides both dedicated and user defined modem control functions. All the control signals are active low so that EIA RS-232C inverting drivers (MC 1488) and inverting receivers (MC 1489) may be used to interface to standard modems. For asynchronous operation, this interface supports programmable NRZI data encode/decode, a digital phase locked loop for efficient clock extraction from NRZI data, and modem control ports with automatic CTS, CD monitoring and RTS generation. This interface also allows the 8273 to operate in PRE-FRAME SYNC mode in which the 8273 prefixes 16 transitions to a frame to synchronize idle lines before transmission of the first flag.

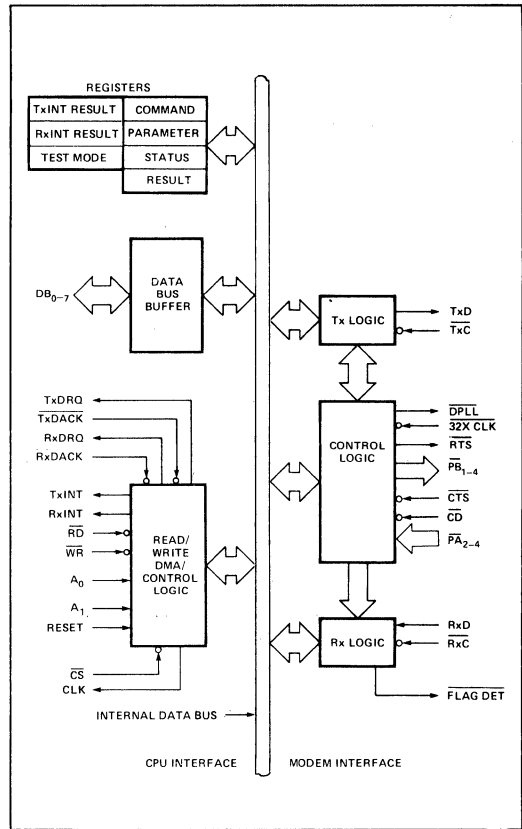
It should be noted that all the 8273 port operations deal with logical values, for instance, bit D0 of Port A will be a one when CTS (Pin 30) is a physical zero (logical one).

**Port A — Input Port**

During operation, the 8273 interrogates input pins CTS (Clear to Send) and CD (Carrier Detect). CTS is used to condition the start of a transmission. If during transmission CTS is lost the 8273 generates an interrupt. During reception, if CD is lost, the 8273 generates an interrupt.



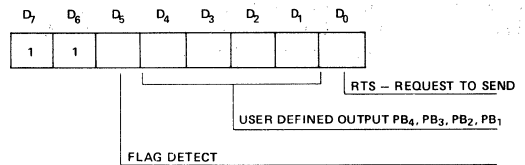
The user defined input bits correspond to the 8273 PA<sub>4</sub>, PA<sub>3</sub> and PA<sub>2</sub> pins. The 8273 does not interrogate or manipulate these bits.



**Figure 5. 8273 Block Diagram Showing Control Logic Functions**

**Port B - Output Port**

During normal operation, if the CPU sets RTS active, the 8273 will not change this pin; however, if the CPU sets RTS inactive, the 8273 will activate it before each transmission and deactivate it one byte time after transmission. While the receiver is active the flag detect pin is pulsed each time a flag sequence is detected in the receive data stream. Following an 8273 reset, all pins of Port B are set to a high, inactive level.



The user defined output bits correspond to the state of PB<sub>4</sub>-PB<sub>1</sub> pins. The 8273 does not interrogate or manipulate these bits.

## Serial Data Logic

The Serial data is synchronized by the user transmit ( $\overline{\text{TxC}}$ ) and receive ( $\overline{\text{RxC}}$ ) clocks. The leading edge of  $\overline{\text{TxC}}$  generates new transmit data and the trailing edge of  $\overline{\text{RxC}}$  is used to capture receive data. The NRZI encoding/decoding of the receive and transmit data is programmable.

The diagnostic features included in the Serial Data logic are programmable loop back of data and selectable clock for the receiver. In the loop-back mode, the data presented to the TxD pin is internally routed to the receive data input

circuitry in place of the RxD pin, thus allowing a CPU to send a message to itself to verify operation of the 8273.

In the selectable clock diagnostic feature, when the data is looped back, the receiver may be presented incorrect sample timing by the external circuitry. The user may select to substitute the  $\overline{\text{TxC}}$  pin for the  $\overline{\text{RxC}}$  input on-chip so that the clock used to generate the loop back data is used to sample it. Since TxD is generated off the leading edge of  $\overline{\text{TxC}}$  and RxD is sampled on the trailing edge, the selected clock allows bit synchronism.

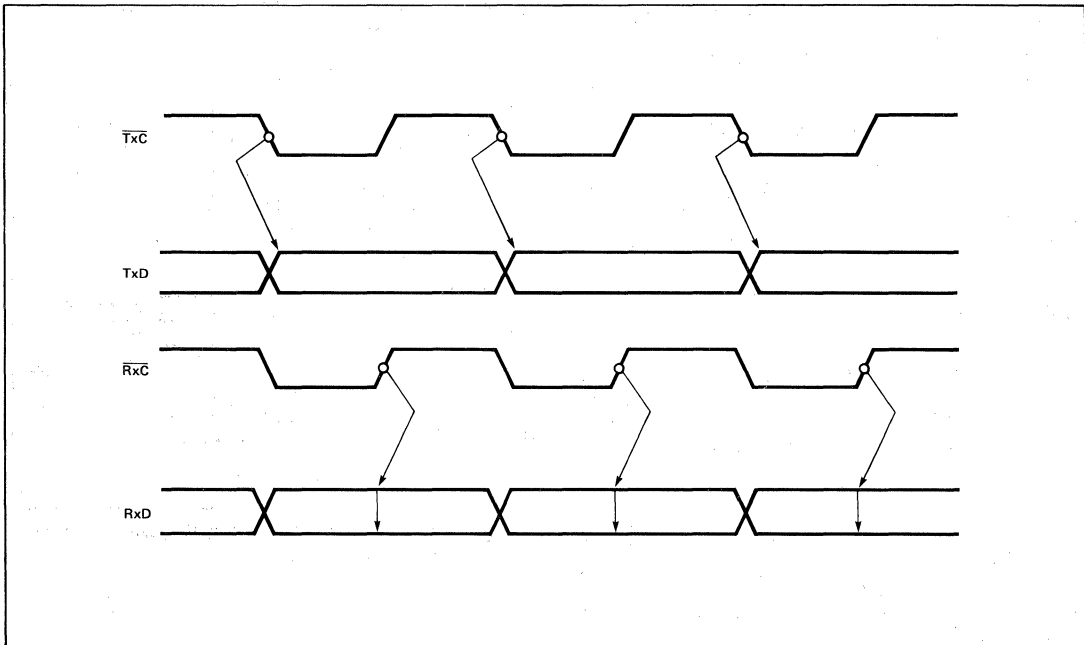


Figure 6. Transmit/Receive Timing

## Asynchronous Mode Interface

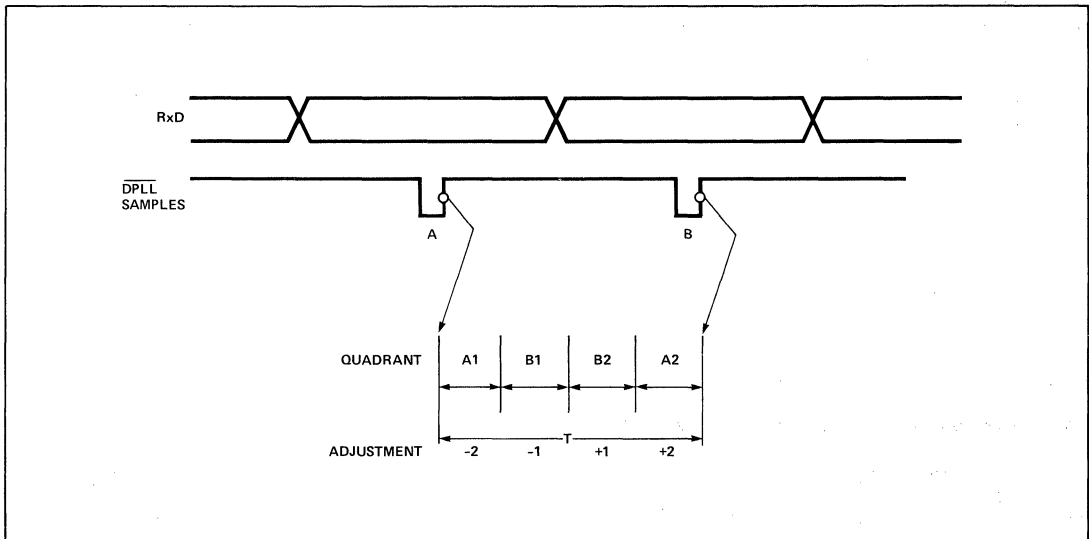
Although the 8273 is fully compatible with the HDLC/SDLC communication line protocols, which are primarily designed for synchronous communication, the 8273 can also be used in asynchronous applications by using this interface. The interface employs a digital phase locked loop (DPLL) for clock recovery from a receive data stream and programmable NRZI encoding and decoding of data. The use of NRZI coding with SDLC transmission

guarantees that within a frame, data transitions will occur at least every five bit times — the longest sequence of ones which may be transmitted without zero-bit insertion. The DPLL should be used only when NRZI coding is used since the NRZI coding will transmit zero sequence as line transitions. The digital phase locked loop also facilitates full-duplex and half-duplex asynchronous implementation with, or without modems.

**Digital Phase Locked Loop**

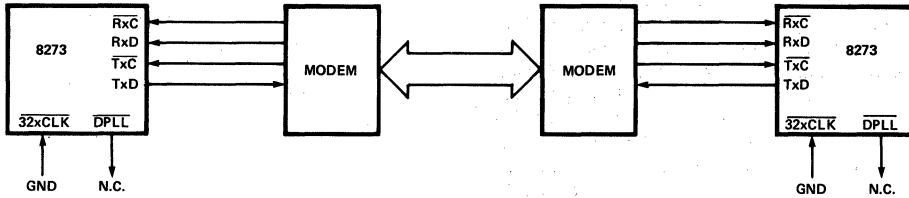
In asynchronous applications, the clock is derived from the receiver data stream by the use of the digital phase locked loop (DPLL). The DPLL requires a clock input at 32 times the required baud rate. The receive data (RxD) is sampled with this 32X CLK and the 8273 DPLL supplies a sample pulse nominally centered on the RxD bit cells. The DPLL has a built-in "stiffness" which reduces sensitivity to line noise and bit distortion. This is accomplished by making phase error adjustments in discrete increments. Since the nominal pulse is made to occur at 32 counts of the 32X CLK, these counts are subtracted or added to the nominal, depending upon which quadrant of the four error quadrants the data edge occurs in. For example if an RxD edge is detected in quadrant A1, it is apparent that the DPLL sample "A" was placed too close to the trailing edge of the data cell; sample "B" will then be placed at  $T = (T_{nominal} - 2 \text{ counts}) = 30$  counts of the 32X CLK to move the sample pulse "B" toward the nominal center of the next bit cell. A data edge occurring in quadrant B1 would cause a smaller adjustment of phase with  $T = 31$  counts of the 32X CLK. Using this technique the DPLL pulse will converge to nominal bit center within 12 data bit times, worst case, with constant incoming RxD edges.

A method of attaining bit synchronism following a line idle is to use PRE-FRAME SYNC mode of transmission.

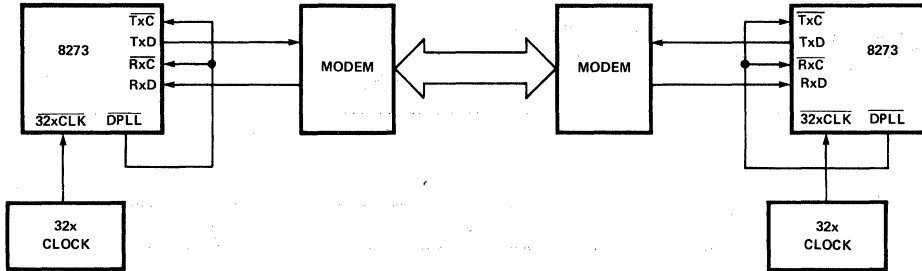


**Figure 7. DPLL Sample Timing**

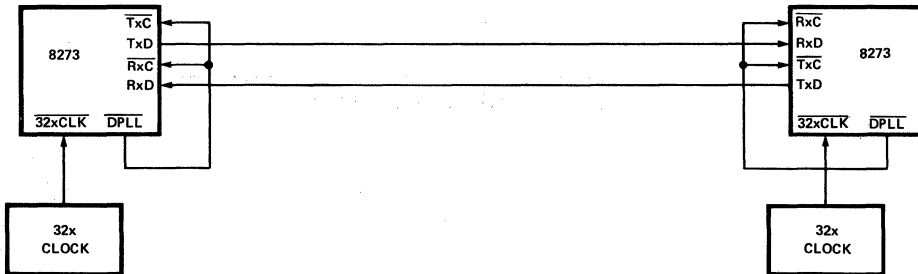
**Synchronous Modem — Duplex or Half Duplex Operation**



**Asynchronous Modems — Duplex or Half Duplex Operation**



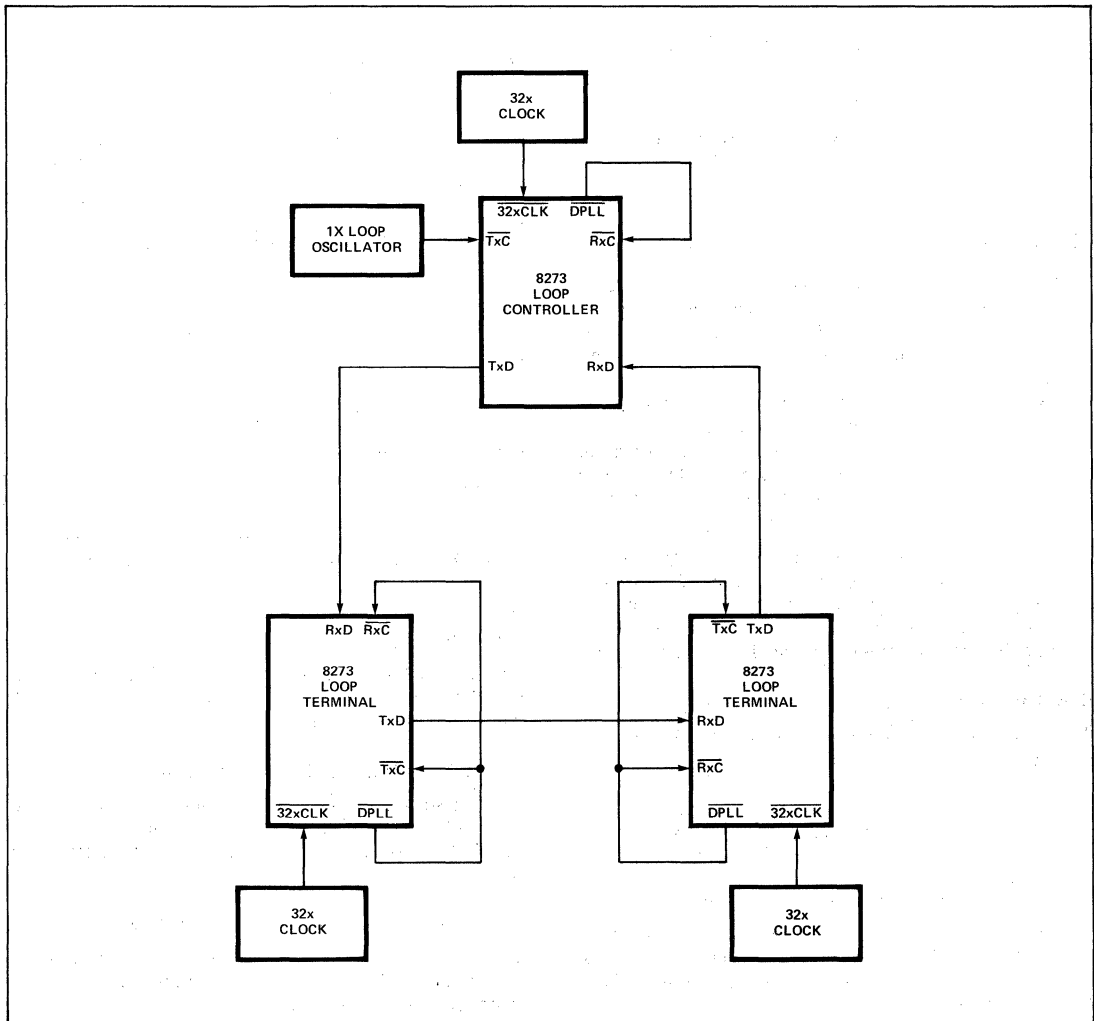
**Asynchronous — No Modems — Duplex or Half Duplex**



**SDLC Loop**

The DPLL simplifies the SDLC loop station implementation. In this application, each secondary station on a loop data link is a repeater set in one-bit delay mode. The signals sent out on the loop by the loop controller (primary station) are relayed from station to station then, back to the controller. Any secondary station finding its address in the A field captures the frame for action at that station. All received frames are relayed to the next station on the loop.

Loop stations are required to derive bit timing from the incoming NRZI data stream. The DPLL generates sample Rx clock timing for reception and uses the same clock to implement Tx clock timing.



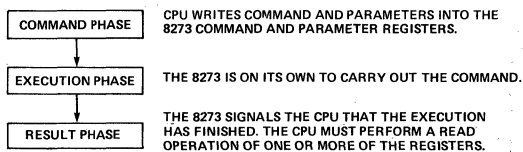
**Figure 8. SDLC Loop Application**



## PRINCIPLES OF OPERATION

The 8273 is an intelligent peripheral controller which relieves the CPU of many of the rote tasks associated with constructing and receiving frames. It is fully compatible with the MCS-80/85™ system bus. As a peripheral device, it accepts commands from a CPU, executes these commands and provides an Interrupt and Result back to the CPU at the end of the execution. The communication with the CPU is done by activation of  $\overline{CS}$ ,  $\overline{RD}$ ,  $\overline{WR}$  pins, while the  $A_1$ ,  $A_0$  select the appropriate registers on the chip as described in the Hardware Description Section.

The 8273 operation is composed of the following sequence of events:



### The Command Phase

During the command phase, the software writes a command to the command register. The command bytes provide a general description of the type of operation requested. Many commands require more detailed information about the command. In such a case up to four parameters are written into the parameter register. The flowchart of the command phase indicates that a command may not be issued if the Status Register indicates that the device is busy. Similarly if a parameter is issued when the Parameter Buffer shows full, incorrect operation will occur.

The 8273 is a duplex device and both transmitter and receiver may each be executing a command or passing results at any given time. For this reason separate interrupt pins are provided. However, the command register must be used for one command sequence at a time.

### Status Register

The status register contains the status of the 8273 activity. The description is as follows.

D <sub>7</sub>	D <sub>6</sub>	D <sub>5</sub>	D <sub>4</sub>	D <sub>3</sub>	D <sub>2</sub>	D <sub>1</sub>	D <sub>0</sub>
CBSY	CBF	CPBF	CRBF	RxINT	TxINT	RxIRA	TxIRA

### Bit 7 CBSY (Command Busy)

Indicates in-progress command, set for CPU poll when Command Register is full, reset upon command phase completion. It is improper to write a command when CBSY is set; it results in incorrect operation.

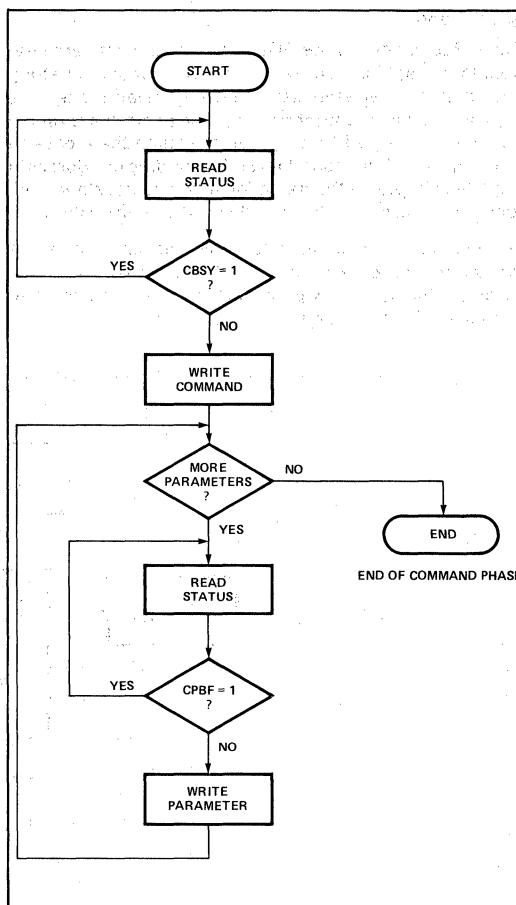


Figure 9. Command Phase Flowchart

### Bit 6 CBF (Command Buffer Full)

Indicates that the command register is full, it is reset when the 8273 accepts the command byte but does not imply that execution has begun.

### Bit 5 CPBF (Command Parameter Buffer Full)

CPBF is set when the parameter buffer is full, and is reset by the 8273 when it accepts the parameter. The CPU may poll CPBF to determine when additional parameters may be written.

### Bit 4 CRBF (Command Result Buffer Full)

Indicates that an executed command immediate result is present in the Result Register. It is set by 8273 and reset when CPU reads the result.

**Bit 3 RxINT (Receiver Interrupt)**

RxINT indicates that the receiver requires CPU attention. It is identical to RxINT (pin 11) and is set by the 8273 either upon good/bad completion of a specified command or by Non-DMA data transfer. It is reset only after the CPU has read the result byte or has received a data byte from the 8273 in a Non-DMA data transfer.

**Bit 2 TxINT (Transmitter Interrupt)**

The TxINT indicates that the transmitter requires CPU attention. It is identical to TxINT (pin 2). It is set by 8273 either upon good/bad completion of a specified command or by Non-DMA data transfer. It is reset only after the CPU has read the result byte or has transferred transmit data byte to the 8273 in a Non-DMA transfer.

**Bit 1 RxIRA (Receiver Interrupt Result Available)**

The RxIRA is set by the 8273 when an interrupt result byte is placed in the RxINT register. It is reset after the CPU has read the RxINT register.

**Bit 0 TxIRA (Transmitter Interrupt Result Available)**

The TxIRA is set by the 8273 when an interrupt result byte is placed in the TxINT register. It is reset when the CPU has read the TxINT register.

**The Execution Phase**

Upon accepting the last parameter, the 8273 enters into the Execution Phase. The execution phase may consist of a DMA or other activity, and may or may not require CPU intervention. The CPU intervention is eliminated in this phase if the system utilizes DMA for the data transfers, otherwise, for non-DMA data transfers, the CPU is interrupted by the 8273 via TxINT and RxINT pins, for each data byte request.

**The Result Phase**

During the result phase, the 8273 notifies the CPU of the execution outcome of a command. This phase is initiated by:

1. The successful completion of an operation
2. An error detected during an operation.

To facilitate quick network software decisions, two types of execution results are provided:

1. An Immediate Result
2. A Non-Immediate Result

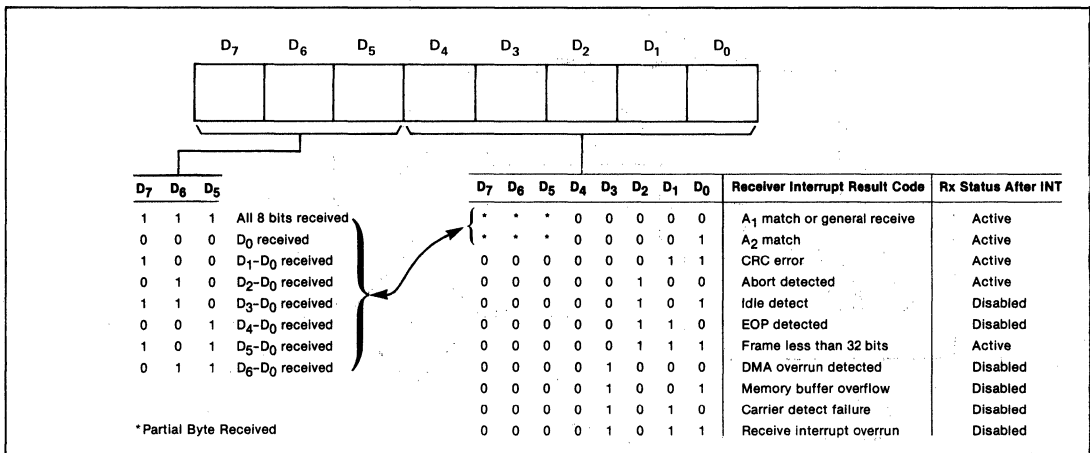


Figure 10. Rx Interrupt Result Byte Format

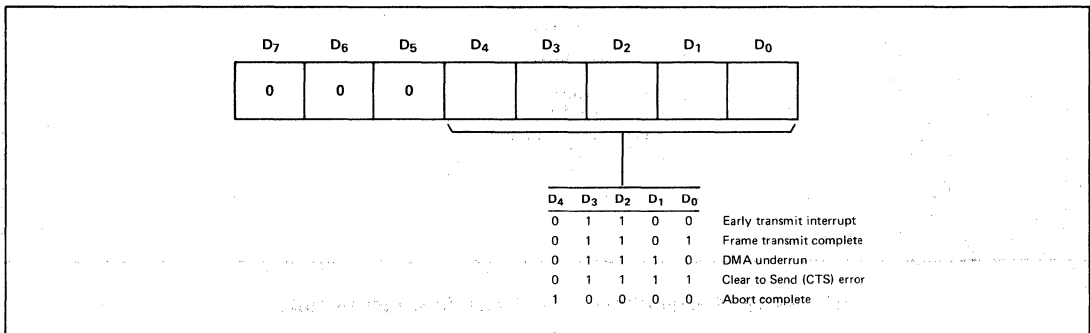


Figure 11. Tx Interrupt Result Byte Format

Immediate result is provided by the 8273 for commands such as Read Port A and Read Port B which have information (CTS, CD, RTS, etc.) that the network software needs to make quick operational decisions.

A command which cannot provide an immediate result will generate an interrupt to signal the beginning of the Result phase. The immediate results are provided in the Result Register; all non-immediate results are available upon device interrupt, through TX Interrupt Result Register TxI/R or Rx Interrupt Result Register RxI/R. The result may consist of a one-byte interrupt code indicating the

condition for the interrupt and, if required, one or more bytes which detail the condition.

**Tx and Rx Interrupt Result Registers**

The Result Registers have a result code, the three high order bits D7-D5 of which are set to zero for all but the receive command. This command result contains a count that indicates the number of bits received in the last byte. If a partial byte is received, the high order bits of the last data byte are indeterminate.

All results indicated in the command summary must be read during the result phase.

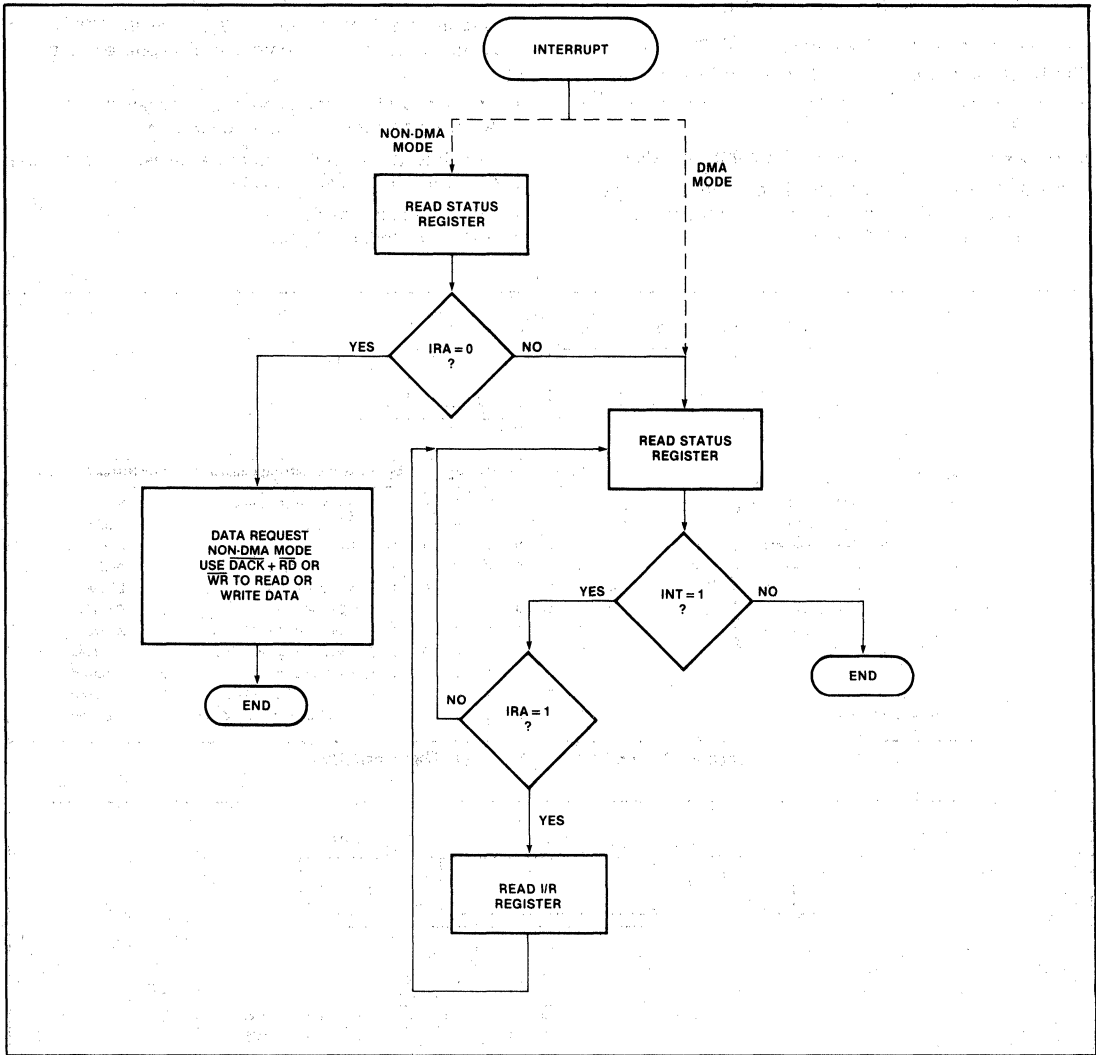


Figure 12. Result Phase Flowchart—Interrupt Results

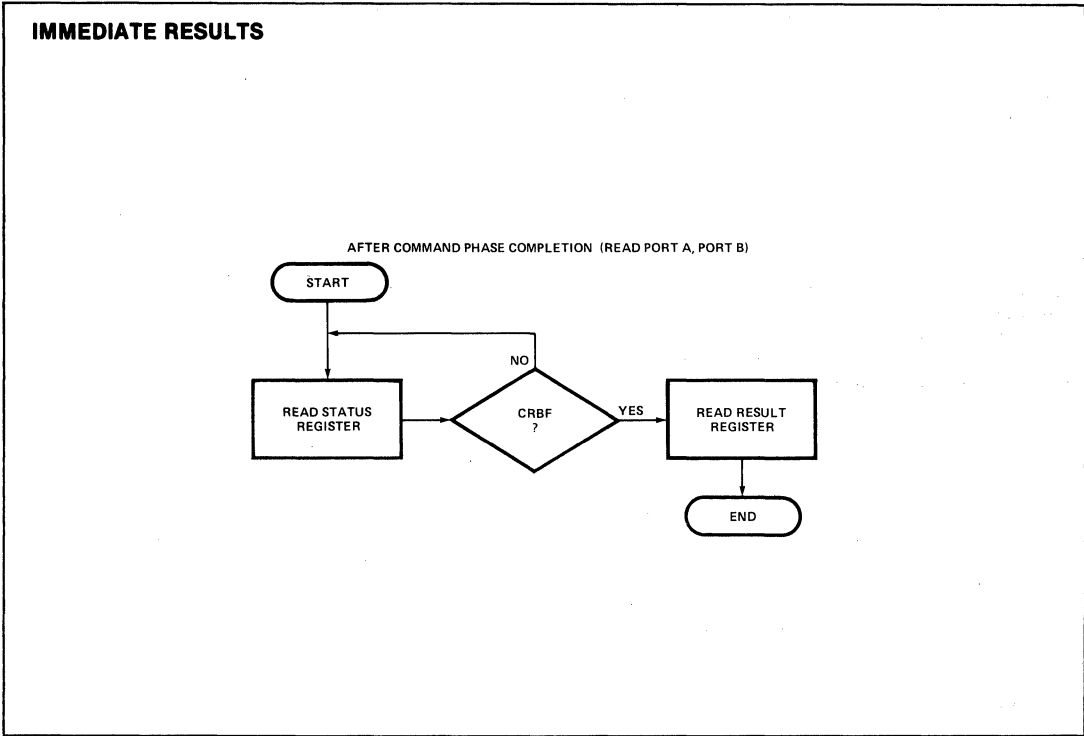


Figure 13. (Rx Interrupt Service)

## DETAILED COMMAND DESCRIPTION

### General

The 8273 HDLC/SDLC controller supports a comprehensive set of high level commands which allows the 8273 to be readily used in full-duplex, half-duplex, synchronous, asynchronous and SDLC loop configuration, with or without modems. These frame-level commands minimize CPU and software overhead. The 8273 has address and control byte buffers which allow the receive and transmit commands to be used in buffered or non-buffered modes.

In buffered transmit mode, the 8273 transmits a flag automatically, reads the Address and Control buffer registers and transmits the fields, then via DMA, it fetches the information field. The 8273, having transmitted the information field, automatically appends the Frame Check Sequence (FCS) and the end flag. Correspondingly, in buffered read mode, the Address and Control fields are stored in their respective buffer registers and only Information Field is transferred to memory.

In non-buffered transmit mode, the 8273 transmits the beginning flag automatically, then fetches and transmits the Address, Control and Information fields from the memory, appends the FCS character and an end flag. In the non-buffered receive mode the entire contents of a frame are sent to memory with the exception of the flags and FCS.

### HDLC Implementation

HDLC Address and Control field are extendable. The extension is selected by setting the low order bit of the field to be extended to a one, a zero in the low order bit indicates the last byte of the respective field.

Since Address/Control field extension is normally done with software to maximize extension flexibility, the 8273 does not create or operate upon contents of the extended HDLC Address/Control fields. Extended fields are transparently passed by the 8273 to user as either interrupt results or data transfer requests. Software must assemble the fields for transmission and interrogate them upon reception.

However, the user can take advantage of the powerful 8273 commands to minimize CPU/Software overhead and simplify buffer management in handling extended fields. For instance buffered mode can be used to separate the first two bytes, then interrogate the others from buffer. Buffered mode is perfect for a two byte address field.

The 8273 when programmed, recognizes protocol characters unique to HDLC such as Abort, which is a string of seven or more ones (01111111). Since Abort character is the same as the GA (EOP) character used in SDLC Loop applications, Loop Transmit and Receive commands are not recommended to be used in HDLC. HDLC does not support Loop mode.

### Initialization Set/Reset Commands

These commands are used to manipulate data within the 8273 registers. The Set commands have a single parameter which is a mask that corresponds to the bits to be set. (They perform a logical-OR of the specified register with the mask provided as a parameter). The Register commands have a single parameter which is a mask that has a zero in the bit positions that are to be reset. (They perform a logical-AND of the specified register with the mask).

#### Set One-Bit Delay (CMD Code A4)

	A <sub>1</sub>	A <sub>0</sub>	D <sub>7</sub>	D <sub>6</sub>	D <sub>5</sub>	D <sub>4</sub>	D <sub>3</sub>	D <sub>2</sub>	D <sub>1</sub>	D <sub>0</sub>
CMD:	0	0	1	0	1	0	0	1	0	0
PAR:	0	1	1	0	0	0	0	0	0	0

When one bit delay is set, 8273 retransmits the received data stream one bit delayed. This mode is entered at a receiver character boundary, and should only be used by Loop Stations.

#### Reset One-Bit Delay (CMD Code 64)

	A <sub>1</sub>	A <sub>0</sub>	D <sub>7</sub>	D <sub>6</sub>	D <sub>5</sub>	D <sub>4</sub>	D <sub>3</sub>	D <sub>2</sub>	D <sub>1</sub>	D <sub>0</sub>
CMD:	0	0	0	1	1	0	0	1	0	0
PAR:	0	1	0	1	1	1	1	1	1	1

The 8273 stops the one bit delayed retransmission mode.

#### Set Data Transfer Mode (CMD Code 97)

	A <sub>1</sub>	A <sub>0</sub>	D <sub>7</sub>	D <sub>6</sub>	D <sub>5</sub>	D <sub>4</sub>	D <sub>3</sub>	D <sub>2</sub>	D <sub>1</sub>	D <sub>0</sub>
CMD:	0	0	1	0	0	1	0	1	1	1
PAR:	0	1	0	0	0	0	0	0	0	1

When the data transfer mode is set, the 8273 will interrupt when data bytes are required for transmission or are available from a receive. If a transmit interrupt occurs and the status indicates that there is no Transmit Result (TxIRA = 0), the interrupt is a transmit data request. If a receive interrupt occurs and the status indicates that there is no receive result (RxIRA = 0), the interrupt is a receive data request.

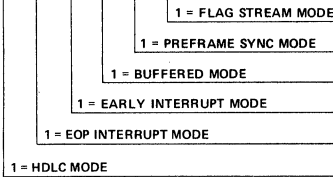
#### Reset Data Transfer Mode (CMD Code 57)

	A <sub>1</sub>	A <sub>0</sub>	D <sub>7</sub>	D <sub>6</sub>	D <sub>5</sub>	D <sub>4</sub>	D <sub>3</sub>	D <sub>2</sub>	D <sub>1</sub>	D <sub>0</sub>
CMD:	0	0	0	1	0	1	0	1	1	1
PAR:	0	1	1	1	1	1	1	1	1	0

If the Data Transfer Mode is reset, the 8273 data transfers are performed through the DMA requests without interrupting the CPU.

**Set Operating Mode (CMD Code 91)**

	A <sub>1</sub>	A <sub>0</sub>	D <sub>7</sub>	D <sub>6</sub>	D <sub>5</sub>	D <sub>4</sub>	D <sub>3</sub>	D <sub>2</sub>	D <sub>1</sub>	D <sub>0</sub>
CMD:	0	0	1	0	0	1	0	0	0	1
PAR:	0	1	0	0						



**Reset Operating Mode (CMD Code 51)**

	A <sub>1</sub>	A <sub>0</sub>	D <sub>7</sub>	D <sub>6</sub>	D <sub>5</sub>	D <sub>4</sub>	D <sub>3</sub>	D <sub>2</sub>	D <sub>1</sub>	D <sub>0</sub>
CMD:	0	0	0	1	0	1	0	0	0	1
PAR:	0	1	1	1						

Any mode switches set in CMD code 91 can be reset using this command by placing zeros in the appropriate positions.

**(D5) HDLC Mode**

In HDLC mode, a bit sequence of seven ones (0111111) is interpreted as an abort character. Otherwise, eight ones (01111111) signal an abort.

**(D4) EOP Interrupt Mode**

In EOP interrupt mode, an interrupt is generated whenever an EOP character (01111111) is detected by an active receiver. This mode is useful for the implementation of an SDLC loop controller in detecting the end of a message stream after a loop poll.

**(D3) Transmitter Early Interrupt Mode (Tx)**

The early interrupt mode is specified to indicate when the 8273 should generate an end of frame interrupt. When set, an early interrupt is generated when the last data character has been passed to the 8273. If the user software responds with another transmit command before the final flag is sent, the final flag interrupt will not be generated and a new frame will immediately begin when the current frame is complete. This permits frames to be separated by a single flag. If no additional Tx commands are provided, a final interrupt will follow.

Note: In buffered mode, if a supervisory frame (no Information) Transmit command is sent in response to an early Transmit Interrupt, the 8273 will repeatedly transmit the same supervisory frame with one flag in between, until a non-supervisory transmit is issued.

Early transmitter interrupt can be used in buffered mode by waiting for a transmit complete interrupt instead of early Transmit Interrupt before issuing a transmit frame command for a supervisory frame. See Figure 14.

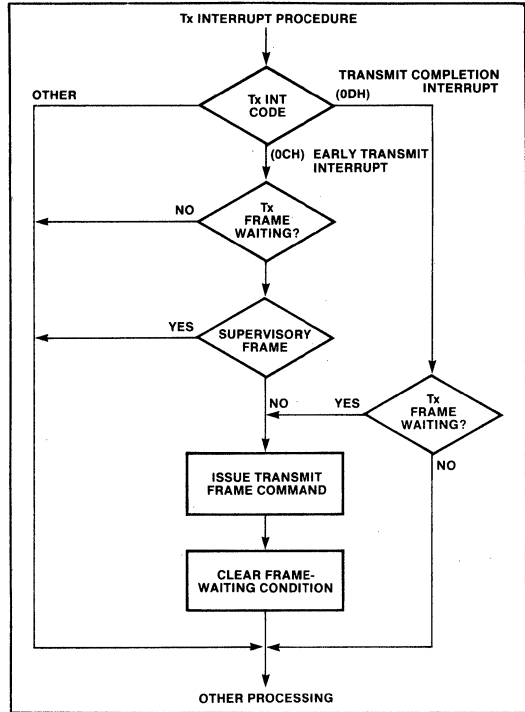


Figure 14.

If this bit is zero, the interrupt will be generated only after the final flag has been transmitted.

**(D2) Buffered Mode**

If the buffered mode bit is set to a one, the first two bytes (normally the address (A) and control (C) fields) of a frame are buffered by the 8273. If this bit is a zero the address and control fields are passed to and from memory.

**(D1) Preframe Sync Mode**

If this bit is set to a one the 8273 will transmit two characters before the first flag of a frame. To guarantee sixteen line transitions, the 8273 sends two bytes of data (00)<sub>H</sub> if NRZI is set or data (55)<sub>H</sub> if NRZI is not set.

**(D0) Flag Stream Mode**

If this bit is set to a one, the following table outlines the operation of the transmitter.

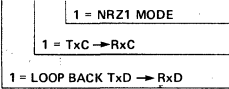
TRANSMITTER STATE	ACTION
Idle	Send Flags immediately.
Transmit or Transmit-Transparent Active	Send Flags after the transmission complete
Loop Transmit Active	Ignore command.
1 Bit Delay Active	Ignore command.

If this bit is reset to zero the following table outlines the operation of the transmitter.

TRANSMITTER STATE	ACTION
IDLE	Send Idles on next character boundary.
Transmit or Transparent Active	Send Idles after the transmission is complete.
Loop Transmit Active	
1 Bit Delay Active	Ignore command.

**Set Serial I/O Mode (CMD Code A0)**

	A <sub>1</sub>	A <sub>0</sub>	D <sub>7</sub>	D <sub>6</sub>	D <sub>5</sub>	D <sub>4</sub>	D <sub>3</sub>	D <sub>2</sub>	D <sub>1</sub>	D <sub>0</sub>
CMD:	0	0	1	0	1	0	0	0	0	0
PAR:	0	1	0	0	0	0	0			



**Reset Serial I/O Mode (CMD Code 60)**

This command allows bits set in CMD code A0 to be reset by placing zeros in the appropriate positions.

	A <sub>1</sub>	A <sub>0</sub>	D <sub>7</sub>	D <sub>6</sub>	D <sub>5</sub>	D <sub>4</sub>	D <sub>3</sub>	D <sub>2</sub>	D <sub>1</sub>	D <sub>0</sub>
CMD:	0	0	0	1	1	0	0	0	0	0
PAR:	0	1	1	1	1	1	1			

**(D2) Loop Back**

If this bit is set to a one, the transmit data is internally routed to the receive data circuitry.

**(D1) Tx C → Rx C**

If this bit is set to a one, the transmit clock is internally routed to the receive clock circuitry. It is normally used with the loop back bit (D2).

**(D0) NRZI Mode**

If this bit is set to a one, NRZI encoding and decoding of transmit and receive data is provided. If this bit is a zero, the transmit and receive data is treated as a normal positive logic bit stream.

NRZI encoding specifies that a zero causes a change in the polarity of the transmitted signal and a one causes no polarity change. NRZI is used in all asynchronous operations. Refer to IBM document GA27-3093 for details.

**Reset Device Command**

	A <sub>1</sub>	A <sub>0</sub>	D <sub>7</sub>	D <sub>6</sub>	D <sub>5</sub>	D <sub>4</sub>	D <sub>3</sub>	D <sub>2</sub>	D <sub>1</sub>	D <sub>0</sub>
TMR:	1	0	0	0	0	0	0	0	0	1
TMR:	1	0	0	0	0	0	0	0	0	0

An 8273 reset command is executed by outputting a (01)<sub>H</sub> followed by (00)<sub>H</sub> to the reset register (TMR). See 8273 AC timing characteristics for Reset pulse specifications.

The reset command emulates the action of the reset pin.

1. The modem control signals are forced high (inactive level).
2. The 8273 status register flags are cleared.
3. Any commands in progress are terminated immediately.
4. The 8273 enters an idle state until the next command is issued.
5. The Serial I/O and Operating Mode registers are set to zero and DMA data register transfer mode is selected.
6. The device assumes a non-loop SDLC terminal role.

**Receive Commands**

The 8273 supports three receive commands: General Receive, Selective Receive, and Selective Loop Receive.

**General Receive (CMD Code C0)**

General receive is a receive mode in which frames are received regardless of the contents of the address field.

	A <sub>1</sub>	A <sub>0</sub>	D <sub>7</sub>	D <sub>6</sub>	D <sub>5</sub>	D <sub>4</sub>	D <sub>3</sub>	D <sub>2</sub>	D <sub>1</sub>	D <sub>0</sub>
CMD:	0	0	1	1	0	0	0	0	0	0
PAR:	0	1	LEAST SIGNIFICANT BYTE OF THE RECEIVE BUFFER LENGTH (B0)							
PAR:	0	1	MOST SIGNIFICANT BYTE OF RECEIVE BUFFER LENGTH (B1)							

**NOTES:**

1. If buffered mode is specified, the R0, R1 receive frame length (result) is the number of data bytes received.
2. If non-buffered mode is specified, the R0, R1 receive frame length (result) is the number of data bytes received plus two (the count includes the address and control bytes).
3. The frame check sequence (FCS) is not transferred to memory.
4. Frames with less than 32 bits between flags are ignored (no interrupt generated) if the buffered mode is specified.
5. In the non-buffered mode an interrupt is generated when a less than 32 bit frame is received, since data transfer requests have occurred.
6. The 8273 receiver is always disabled when an Idle is received after a valid frame. The CPU module must issue a receive command to re-enable the receiver.
7. The intervening ABORT character between a final flag and an IDLE does not generate an interrupt.
8. If an ABORT Character is not preceded by a flag and is followed by an IDLE, an interrupt will be generated for the ABORT followed by an IDLE interrupt one character time later. The reception of an ABORT will disable the receiver.

**Selective Receive (CMD Code C1)**

	A <sub>1</sub>	A <sub>0</sub>	D <sub>7</sub>	D <sub>6</sub>	D <sub>5</sub>	D <sub>4</sub>	D <sub>3</sub>	D <sub>2</sub>	D <sub>1</sub>	D <sub>0</sub>
CMD:	0	0	1	1	0	0	0	0	0	1
PAR:	0	1	LEAST SIGNIFICANT BYTE OF THE RECEIVE BUFFER LENGTH (B0)							
PAR:	0	1	MOST SIGNIFICANT BYTE OF RECEIVE BUFFER LENGTH (B1)							
PAR:	0	1	RECEIVE FRAME ADDRESS MATCH FIELD ONE (A1)							
PAR:	0	1	RECEIVE FRAME ADDRESS MATCH FIELD TWO (A2)							

Selective receive is a receive mode in which frames are ignored unless the address field matches any one of two address fields given to the 8273 as parameters.

When selective receive is used in HDLC the 8273 looks at the first character, if extended, software must then decide if the message is for this unit.

### Selective Loop Receive (CMD Code C2)

	A <sub>1</sub>	A <sub>0</sub>	D <sub>7</sub>	D <sub>6</sub>	D <sub>5</sub>	D <sub>4</sub>	D <sub>3</sub>	D <sub>2</sub>	D <sub>1</sub>	D <sub>0</sub>
CMD:	0	0	1	1	0	0	0	0	1	0
PAR:	0	0	LEAST SIGNIFICANT BYTE OF THE RECEIVE BUFFER LENGTH (B0)							
PAR:	0	1	MOST SIGNIFICANT BYTE OF RECEIVE BUFFER LENGTH (B1)							
PAR:	0	1	RECEIVE FRAME ADDRESS MATCH FIELD ONE (A1)							
PAR:	0	1	RECEIVE FRAME ADDRESS MATCH FIELD TWO (A2)							

Selective loop receive operates like selective receive except that the transmitter is placed in flag stream mode automatically after detecting an EOP (01111111) following a valid received frame. The one bit delay mode is also reset at the end of a selective loop receive.

### Receive Disable (CMD Code C5)

Terminates an active receive command immediately.

	A <sub>1</sub>	A <sub>0</sub>	D <sub>7</sub>	D <sub>6</sub>	D <sub>5</sub>	D <sub>4</sub>	D <sub>3</sub>	D <sub>2</sub>	D <sub>1</sub>	D <sub>0</sub>
CMD:	0	0	1	1	0	0	0	1	0	1
PAR:	NONE									

### Transmit Commands

The 8273 supports three transmit commands: Transmit Frame, Loop Transmit, Transmit Transparent.

### Transmit Frame (CMD Code C8)

	A <sub>1</sub>	A <sub>0</sub>	D <sub>7</sub>	D <sub>6</sub>	D <sub>5</sub>	D <sub>4</sub>	D <sub>3</sub>	D <sub>2</sub>	D <sub>1</sub>	D <sub>0</sub>
CMD:	0	0	1	1	0	0	1	0	0	0
PAR:	0	1	LEAST SIGNIFICANT BYTE OF FRAME LENGTH (L0)							
PAR:	0	1	MOST SIGNIFICANT BYTE OF FRAME LENGTH (L1)							
PAR:	0	1	ADDRESS FIELD OF TRANSMIT FRAME (A)							
PAR:	0	1	CONTROL FIELD OF TRANSMIT FRAME (C)							

Transmits one frame including: initial flag, frame check sequence, and the final flag.

If the buffered mode is specified, the L0, L1, frame length provided as a parameter is the length of the information field and the address and control fields must be input.

In unbuffered mode the frame length provided must be the length of the information field plus two and the address and control fields must be the first two bytes of data. Thus only the frame length bytes are required as parameters.

### Loop Transmit (CMD Code CA)

	A <sub>1</sub>	A <sub>0</sub>	D <sub>7</sub>	D <sub>6</sub>	D <sub>5</sub>	D <sub>4</sub>	D <sub>3</sub>	D <sub>2</sub>	D <sub>1</sub>	D <sub>0</sub>
CMD:	0	0	1	1	0	0	1	0	1	0
PAR:	0	1	LEAST SIGNIFICANT BYTE OF FRAME LENGTH (L0)							
PAR:	0	1	MOST SIGNIFICANT BYTE OF FRAME LENGTH (L1)							
PAR:	0	1	ADDRESS FIELD OF TRANSMIT FRAME (A)							
PAR:	0	1	CONTROL FIELD OF TRANSMIT FRAME (C)							

Transmits one frame in the same manner as the transmit frame command except:

1. If the flag stream mode is not active transmission will begin after a received EOP has been converted to a flag.
2. If the flag stream mode is active transmission will begin at the next flag boundary for buffered mode or at the third flag boundary for non-buffered mode.
3. At the end of a loop transmit the one-bit delay mode is entered and the flag stream mode is reset.

### Transmit Transparent (CMD Coded C9)

	A <sub>1</sub>	A <sub>0</sub>	D <sub>7</sub>	D <sub>6</sub>	D <sub>5</sub>	D <sub>4</sub>	D <sub>3</sub>	D <sub>2</sub>	D <sub>1</sub>	D <sub>0</sub>
CMD:	0	0	1	1	0	0	1	0	0	1
PAR:	0	1	LEAST SIGNIFICANT BYTE OF FRAME LENGTH (L0)							
PAR:	0	1	MOST SIGNIFICANT BYTE OF FRAME LENGTH (L1)							

The 8273 will transmit a block of raw data without protocol, i.e., no zero bit insertion, flags, or frame check sequences.

### Abort Transmit Commands

An abort command is supported for each type of transmit command. The abort commands are ignored if a transmit command is not in progress.

### Abort Transmit Frame (CMD Code CC)

	A <sub>1</sub>	A <sub>0</sub>	D <sub>7</sub>	D <sub>6</sub>	D <sub>5</sub>	D <sub>4</sub>	D <sub>3</sub>	D <sub>2</sub>	D <sub>1</sub>	D <sub>0</sub>
CMD:	0	0	1	1	0	0	1	1	0	0
PAR:	NONE									

After an abort character (eight contiguous ones) is transmitted, the transmitter reverts to sending flags or idles as a function of the flag stream mode specified.

### Abort Loop Transmit (CMD Code CE)

	A <sub>1</sub>	A <sub>0</sub>	D <sub>7</sub>	D <sub>6</sub>	D <sub>5</sub>	D <sub>4</sub>	D <sub>3</sub>	D <sub>2</sub>	D <sub>1</sub>	D <sub>0</sub>
CMD:	0	0	1	1	0	0	1	1	1	0
PAR:	NONE									

After a flag is transmitted the transmitter reverts to one bit delay mode.

### Abort Transmit Transparent (CMD Code CD)

	A <sub>1</sub>	A <sub>0</sub>	D <sub>7</sub>	D <sub>6</sub>	D <sub>5</sub>	D <sub>4</sub>	D <sub>3</sub>	D <sub>2</sub>	D <sub>1</sub>	D <sub>0</sub>
CMD:	0	0	1	1	0	0	1	1	0	1
PAR:	NONE									

The transmitter reverts to sending flags or idles as a function of the flag stream mode specified.



**Modem Control Commands**

The modem control commands are used to manipulate the modem control ports.

When read Port A or Port B commands are executed the result of the command is returned in the result register. The Bit Set Port B command requires a parameter that is a mask that corresponds to the bits to be set. The Bit Reset Port B command requires a mask that has a zero in the bit positions that are to be reset.

**Read Port A (CMD Code 22)**

	A <sub>1</sub>	A <sub>0</sub>	D <sub>7</sub>	D <sub>6</sub>	D <sub>5</sub>	D <sub>4</sub>	D <sub>3</sub>	D <sub>2</sub>	D <sub>1</sub>	D <sub>0</sub>
CMD:	0	0	0	0	1	0	0	0	1	0
PAR:	NONE									

**Read Port B (CMD Code 23)**

	A <sub>1</sub>	A <sub>0</sub>	D <sub>7</sub>	D <sub>6</sub>	D <sub>5</sub>	D <sub>4</sub>	D <sub>3</sub>	D <sub>2</sub>	D <sub>1</sub>	D <sub>0</sub>
CMD:	0	0	0	0	1	0	0	0	1	1
PAR:	NONE									

**Set Port B Bits (CMD Code A3)**

This command allows user defined Port B pins to be set.

	A <sub>1</sub>	A <sub>0</sub>	D <sub>7</sub>	D <sub>6</sub>	D <sub>5</sub>	D <sub>4</sub>	D <sub>3</sub>	D <sub>2</sub>	D <sub>1</sub>	D <sub>0</sub>
CMD:	0	0	1	0	1	0	0	0	1	1
PAR:	0	1	0	0						

RTS – REQUEST TO SEND

USER DEFINED

FLAG DETECT

**(D5) Flag Detect**

This bit can be used to set the flag detect pin. However, it will be reset when the next flag is detected.

**(D4-D1) User Defined Outputs**

These bits correspond to the state of the PB<sub>4</sub>-PB<sub>1</sub> output pins.

**(D0) Request to Send**

This is a dedicated 8273 modem control signal, and reflects the same logical state of RTS pin.

**Reset Port B Bits (CMD Code 63)**

This command allows Port B user defined bits to be reset.

	A <sub>1</sub>	A <sub>0</sub>	D <sub>7</sub>	D <sub>6</sub>	D <sub>5</sub>	D <sub>4</sub>	D <sub>3</sub>	D <sub>2</sub>	D <sub>1</sub>	D <sub>0</sub>
CMD:	0	0	0	1	1	0	0	0	1	1
PAR:	0	1	1	1						

RTS – REQUEST TO SEND

USER DEFINED

FLAG DETECT

This command allows Port B (D<sub>4</sub>-D<sub>1</sub>) user defined bits to be reset. These bits correspond to Output Port pins (PB<sub>4</sub>-PB<sub>1</sub>).

**8273 Command Summary**

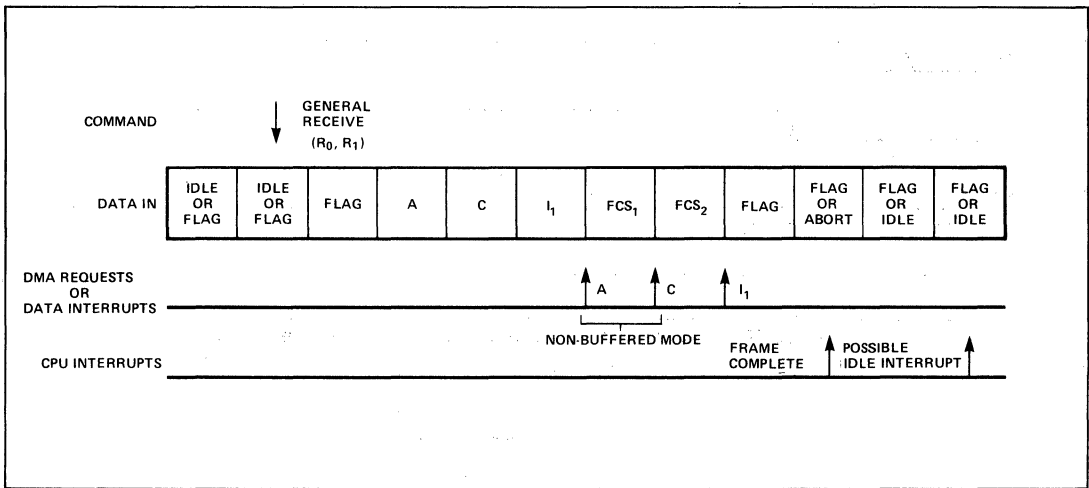
Command Description	Command (HEX)	Parameter	Results	Result Port	Completion Interrupt
Set One Bit Delay	A4	Set Mask	None	—	No
Reset One Bit Delay	64	Reset Mask	None	—	No
Set Data Transfer Mode	97	Set Mask	None	—	No
Reset Data Transfer Mode	57	Reset Mask	None	—	No
Set Operating Mode	91	Set Mask	None	—	No
Reset Operating Mode	51	Reset Mask	None	—	No
Set Serial I/O Mode	A0	Set Mask	None	—	No
Reset Serial I/O Mode	60	Reset Mask	None	—	No
General Receive	C0	B0,B1	RIC,R0,R1,(A,C) <sup>(2)</sup>	RXI/R	Yes
Selective Receive	C1	B0,B1,A1,A2	RIC,R0,R1,(A,C) <sup>(2)</sup>	RXI/R	Yes
Selective Loop Receive	C2	B0,B1,A1,A2	RIC,R0,R1,(A,C) <sup>(2)</sup>	RXI/R	Yes
Receive Disable	C5	None	None	—	No
Transmit Frame	C8	L0,L1,(A,C) <sup>(1)</sup>	TIC	TXI/R	Yes
Loop Transmit	CA	L0,L1,(A,C) <sup>(1)</sup>	TIC	TXI/R	Yes
Transmit Transparent	C9	L0,L1	TIC	TXI/R	Yes
Abort Transmit Frame	CC	None	TIC	TXI/R	Yes
Abort Loop Transmit	CE	None	TIC	TXI/R	Yes
Abort Transmit Transparent	CD	None	TIC	TXI/R	Yes
Read Port A	22	None	Port Value	Result	No
Read Port B	23	None	Port Value	Result	No
Set Port B Bit	A3	Set Mask	None	—	No
Reset Port B Bit	63	Reset Mask	None	—	No

**NOTES:**

1. Issued only when in buffered mode.
2. Read as results only in buffered mode.

**8273 Command Summary Key**

- B0** — Least significant byte of the receive buffer length.
- B1** — Most significant byte of the receive buffer length.
- L0** — Least significant byte of the Tx frame length.
- L1** — Most significant byte of the Tx frame length.
- A1** — Receive frame address match field one.
- A2** — Receive frame address match field two.
- A** — Address field of received frame. If non-buffered mode is specified, this result is not provided.
- C** — Control field of received frame. If non-buffered mode is specified this result is not provided.
- RX1/R** — Receive interrupt result register.
- TX1/R** — Transmit interrupt result register.
- R0** — Least significant byte of the length of the frame received.
- R1** — Most significant byte of the length of the frame received.
- RIC** — Receiver interrupt result code.
- TIC** — Transmitter interrupt result code.



**Figure 15. Typical Frame Reception**

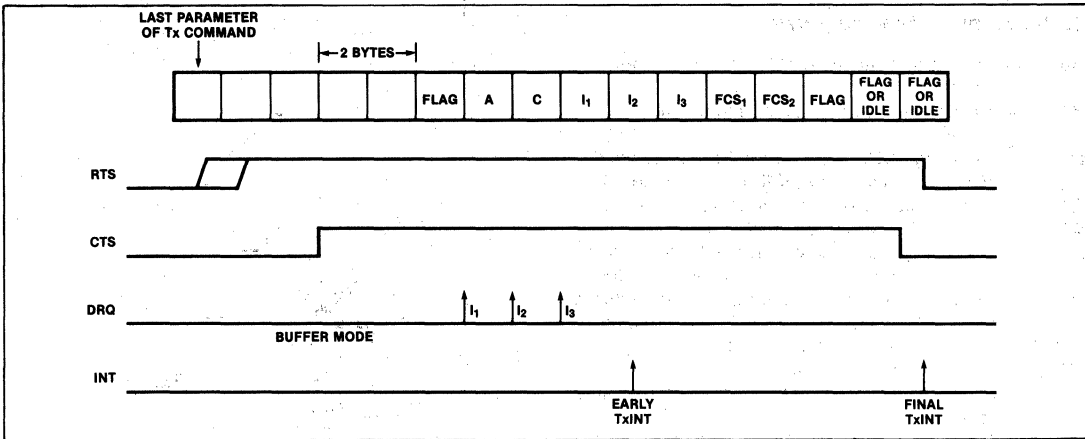


Figure 16a. Typical Frame Transmission, Buffered Mode

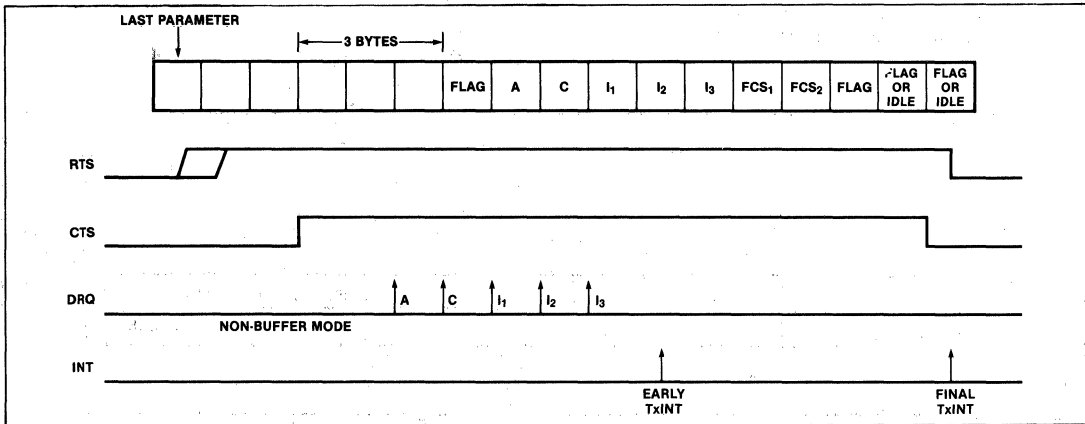


Figure 16b. Typical Frame Transmission, Non-Buffered Mode

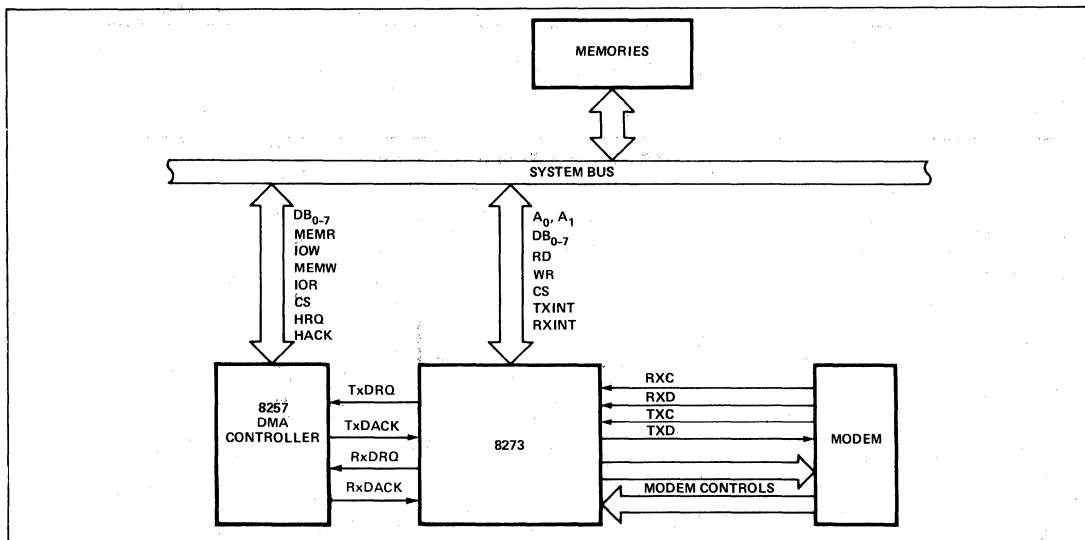


Figure 17. 8273 System Diagram

**ABSOLUTE MAXIMUM RATINGS\***

Ambient Temperature Under Bias	0°C to 70°C
Storage Temperature	-65°C to +150°C
Voltage on Any Pin With Respect to Ground	-0.5V to +7V
Power Dissipation	1 Watt

*\*NOTICE: Stresses above those listed under "Absolute Maximum Ratings" may cause permanent damage to the device. This is a stress rating only and functional operation of the device at these or any other conditions above those indicated in the operational sections of this specification is not implied. Exposure to absolute maximum rating conditions for extended periods may affect device reliability.*

**D.C. CHARACTERISTICS (8273, 8273-4, 8273-8) (T<sub>A</sub> = 0°C to 70°C, V<sub>CC</sub> = +5.0V ± 5%)**

Symbol	Parameter	Min.	Max.	Unit	Test Conditions
V <sub>IL</sub>	Input Low Voltage	-0.5	0.8	V	
V <sub>IH</sub>	Input High Voltage	2.0	V <sub>CC</sub> + 0.5	V	
V <sub>OL</sub>	Output Low Voltage		0.45	V	I <sub>OL</sub> = 2.0 mA for Data Bus Pins I <sub>OL</sub> = 1.0 mA for Output Port Pins I <sub>OL</sub> = 1.6 mA for All Other Pins
V <sub>OH</sub>	Output High Voltage	2.4		V	I <sub>OH</sub> = -200 μA for Data Bus Pins I <sub>OH</sub> = -100 μA for All Other Pins
I <sub>IL</sub>	Input Load Current		± 10	μA	V <sub>IN</sub> = V <sub>CC</sub> to 0V
I <sub>OZ</sub>	Off-State Output Current		± 10	μA	V <sub>OUT</sub> = V <sub>CC</sub> to 0V
I <sub>CC</sub>	V <sub>CC</sub> Supply Current		180	mA	

**CAPACITANCE (8273, 8273-4, 8273-8) (T<sub>A</sub> = 25°C, V<sub>CC</sub> = GND = 0V)**

Symbol	Parameter	Min.	Typ.	Max.	Unit	Test Conditions
C <sub>IN</sub>	Input Capacitance			10	pF	t <sub>c</sub> = 1 MHz
C <sub>I/O</sub>	I/O Capacitance			20	pF	Unmeasured Pins Returned to GND

**A.C. CHARACTERISTICS (T<sub>A</sub> = 0°C to 70°C, V<sub>CC</sub> = +5.0V ± 5%)**
**CLOCK TIMING (8273)**

Symbol	Parameter	Min.	Typ.	Max.	Unit	Test Conditions
t <sub>CY</sub>	Clock	250		2000	ns	64K Baud Max Operating Rate
t <sub>CL</sub>	Clock Low	120			ns	
t <sub>CH</sub>	Clock High	120			ns	

**CLOCK TIMING (8273-4)**

Symbol	Parameter	Min.	Typ.	Max.	Unit	Test Conditions
t <sub>CY</sub>	Clock	286		2000	ns	56K Baud Max Operating Rate
t <sub>CL</sub>	Clock Low	135			ns	
t <sub>CH</sub>	Clock High	135			ns	

**CLOCK TIMING (8273-8)**

Symbol	Parameter	Min.	Typ.	Max.	Unit	Test Conditions
t <sub>CY</sub>	Clock	330		2000	ns	48K Baud Max Operating Rate
t <sub>CL</sub>	Clock Low	150			ns	
t <sub>CH</sub>	Clock High	150			ns	

**A.C. CHARACTERISTICS (8273, 8273-4, 8273-8) ( $T_A = 0^\circ\text{C}$  to  $70^\circ\text{C}$ ,  $V_{CC} = +5.0\text{V} \pm 5\%$ )**
**READ CYCLE**

Symbol	Parameter	Min.	Max.	Unit	Test Conditions
$t_{AC}$	Select Setup to $\overline{RD}$	0		ns	Note 2
$t_{CA}$	Select Hold from $\overline{RD}$	0		ns	Note 2
$t_{RR}$	$\overline{RD}$ Pulse Width	250		ns	
$t_{AD}$	Data Delay from Address		300	ns	Note 2
$t_{RD}$	Data Delay from $\overline{RD}$		200	ns	$C_L = 150\text{pF}$ , Note 2
$t_{DF}$	Output Float Delay	20	100	ns	$C_L = 20\text{pF}$ for Minimum; $150\text{pF}$ for Maximum
$t_{DC}$	DACK Setup to $\overline{RD}$	25		ns	
$t_{CD}$	DACK Hold from $\overline{RD}$	25		ns	
$t_{KD}$	Data Delay from DACK		300	ns	

**WRITE CYCLE**

Symbol	Parameter	Min.	Max.	Unit	Test Conditions
$t_{AC}$	Select Setup to $\overline{WR}$	0		ns	
$t_{CA}$	Select Hold from $\overline{WR}$	0		ns	
$t_{WW}$	$\overline{WR}$ Pulse Width	250		ns	
$t_{DW}$	Data Setup to $\overline{WR}$	150		ns	
$t_{WD}$	Data Hold from $\overline{WR}$	0		ns	
$t_{DC}$	DACK Setup to $\overline{WR}$	25		ns	
$t_{CD}$	DACK Hold from $\overline{WR}$	25		ns	

**DMA**

Symbol	Parameter	Min.	Max.	Unit	Test Conditions
$t_{CQ}$	Request Hold from $\overline{WR}$ or $\overline{RD}$ (for Non-Burst Mode)		200	ns	

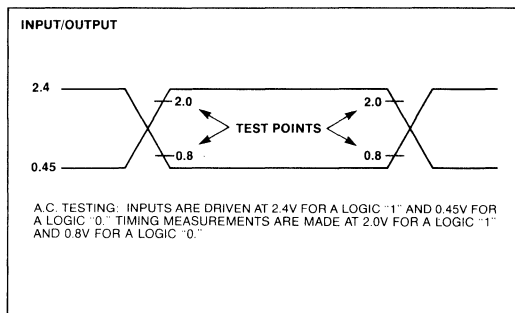
**OTHER TIMING**

Symbol	Parameter	Min.	Max.	Unit	Test Conditions
$t_{RSTW}$	Reset Pulse Width	10		$t_{CY}$	
$t_r$	Input Signal Rise Time		20	ns	
$t_f$	Input Signal Fall Time		20	ns	
$t_{RSTS}$	Reset to First $\overline{IOWR}$	2		$t_{CY}$	
$t_{CY32}$	32X Clock Cycle Time	$9.7 \cdot t_{CY}$		ns	
$t_{CL32}$	32X Clock Low Time	$4 \cdot t_{CY}$		ns	
$t_{CH32}$	32X Clock High Time	$4 \cdot t_{CY}$		ns	
$t_{DPLL}$	DPLL Output Low	$1 \cdot t_{CY} - 50$		ns	
$t_{DCL}$	Data Clock Low	$1 \cdot t_{CY} - 50$		ns	
$t_{DCH}$	Data Clock High	$2 \cdot t_{CY}$		ns	
$t_{DCY}$	Data Clock	$62.5 \cdot t_{CY}$		ns	
$t_{TD}$	Transmit Data Delay		200	ns	
$t_{DS}$	Data Setup Time	200		ns	
$t_{DH}$	Data Hold Time	100		ns	
$t_{FLD}$	FLAG DET Output Low	$8 \cdot t_{CY} \pm 50$		ns	

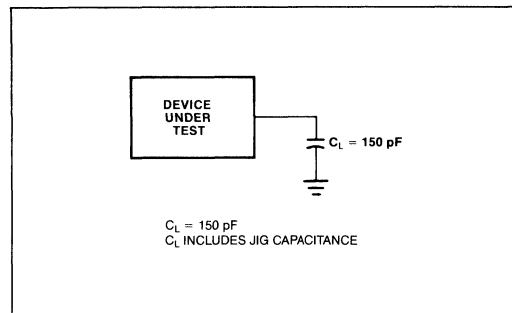
**NOTES:**

- All timing measurements are made at the reference voltages unless otherwise specified: Input "1" at 2.0V, "0" at 0.8V; Output "1" at 2.0V, "0" at 0.8V.
- $t_{AD}$ ,  $t_{RD}$ ,  $t_{AC}$ , and  $t_{CA}$  are not concurrent specs.

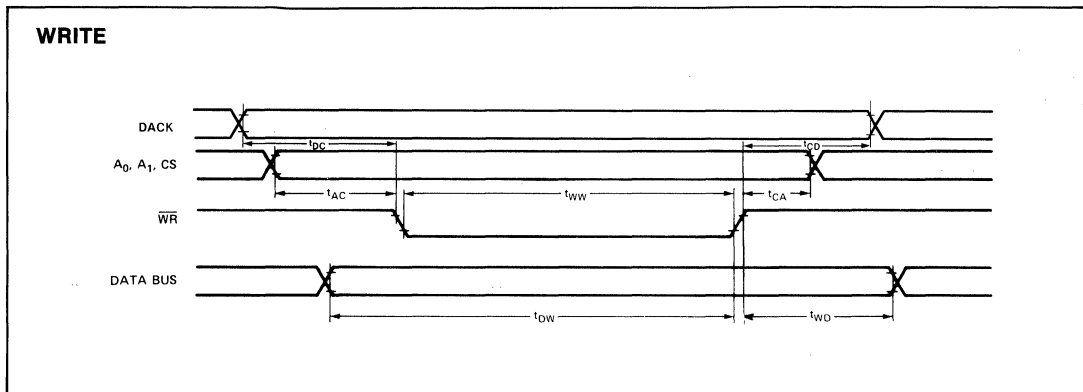
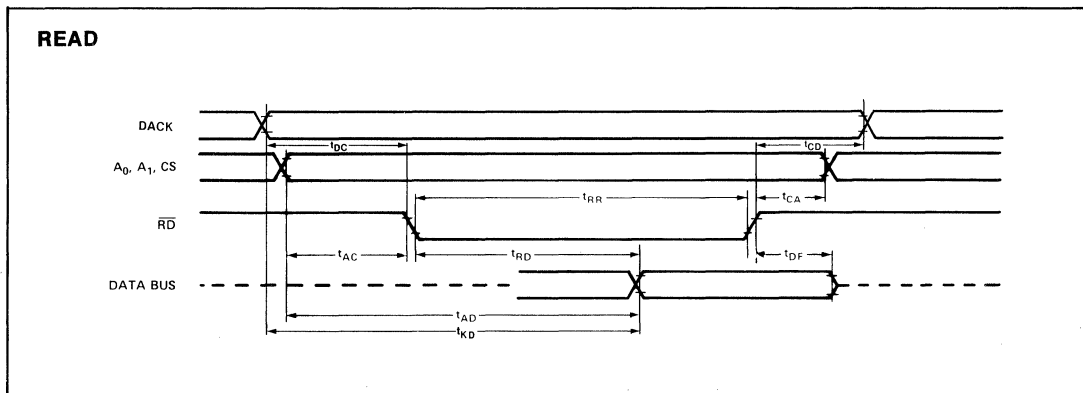
**A.C. TESTING INPUT, OUTPUT WAVEFORM**



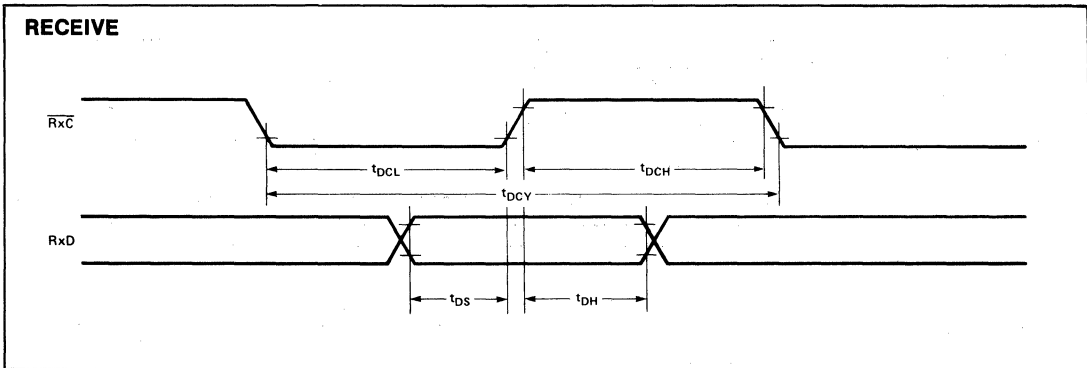
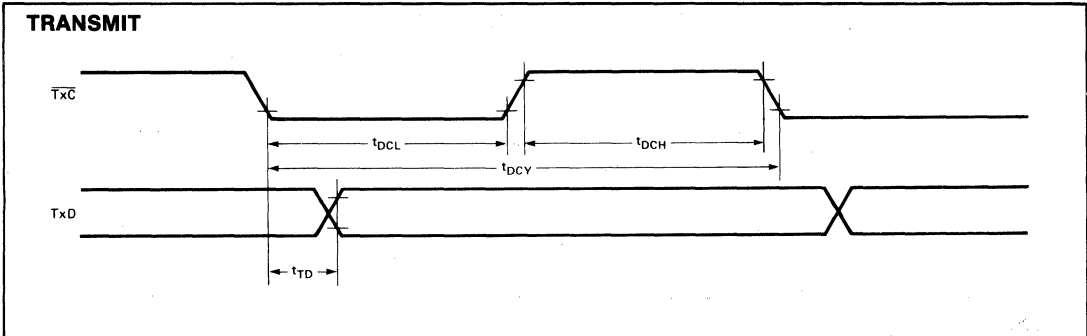
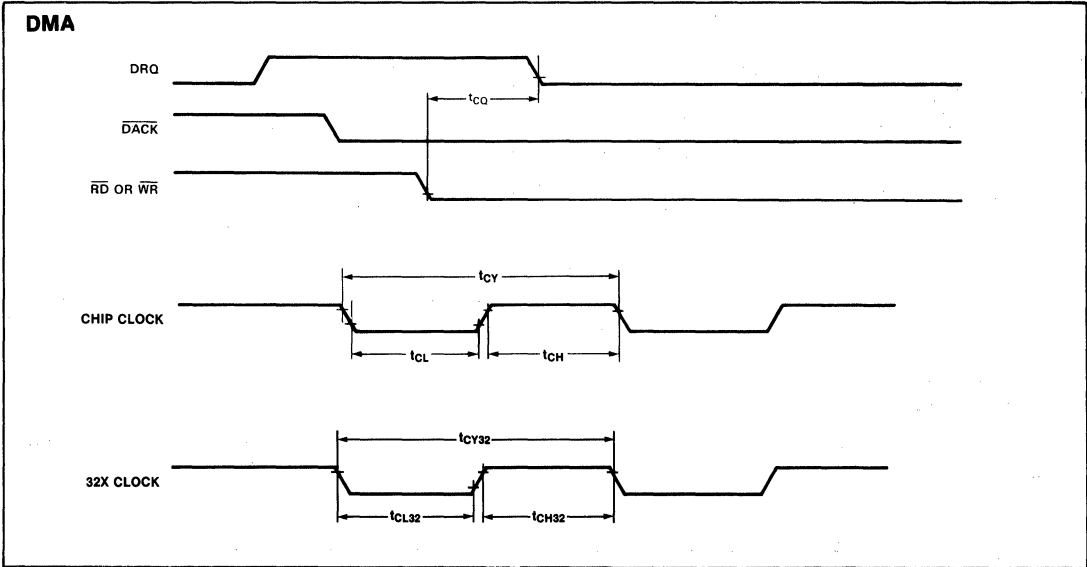
**A.C. TESTING LOAD CIRCUIT**



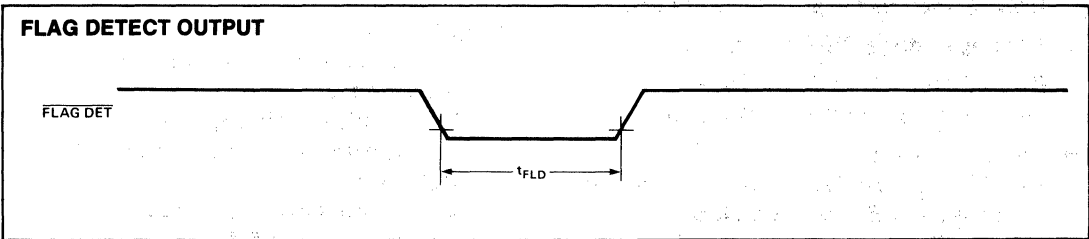
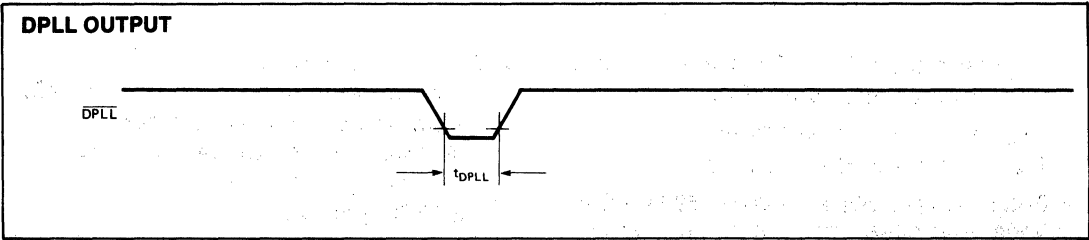
**WAVEFORMS**



**WAVEFORMS (Continued)**



**WAVEFORMS (Continued)**





# 8274 MULTI-PROTOCOL SERIAL CONTROLLER (MPSC)

- **Asynchronous, Byte Synchronous and Bit Synchronous Operation**
- **Two Independent Full Duplex Transmitters and Receivers**
- **Fully Compatible with 8048, 8051, 8085, 8088, and 8086 CPU's; 8257 and 8237 DMA Controllers; and 8089 I/O Proc.**
- **4 Independent DMA Channels**
- **Baud Rate: DC to 880K Baud**  
—Future Selections to 1M Baud
- **Asynchronous:**  
—5-8 Bit Character; Odd, Even, or No Parity; 1, 1.5 or 2 Stop Bits  
—Error Detection: Framing, Overrun, and Parity
- **Byte Synchronous:**  
— Character Synchronization, Int. or Ext.  
— One or Two Sync Characters  
— Automatic CRC Generation and Checking (CRC-16)  
— IBM Bisync Compatible
- **Bit Synchronous:**  
— SDLC/HDLC Flag Generation and Recognition  
— 8 Bit Address Recognition  
— Automatic Zero Bit Insertion and Deletion  
— Automatic CRC Generation and Checking (CCITT-16)  
— CCITT X.25 Compatible

The Intel® 8274 Multi-Protocol Series Controller (MPSC) is designed to interface High Speed Communications Lines using Asynchronous, IBM Bisync, and SDLC/HDLC protocol to Intel microcomputer systems. It can be interfaced with Intel's MCS-48, -85, -51; iAPX-86, and -88 families, the 8237 DMA Controller, or the 8089 I/O Processor in polled, interrupt driven, or DMA driven modes of operation.

The MPSC is a 40 pin device fabricated using Intel's High Performance HMOS Technology.

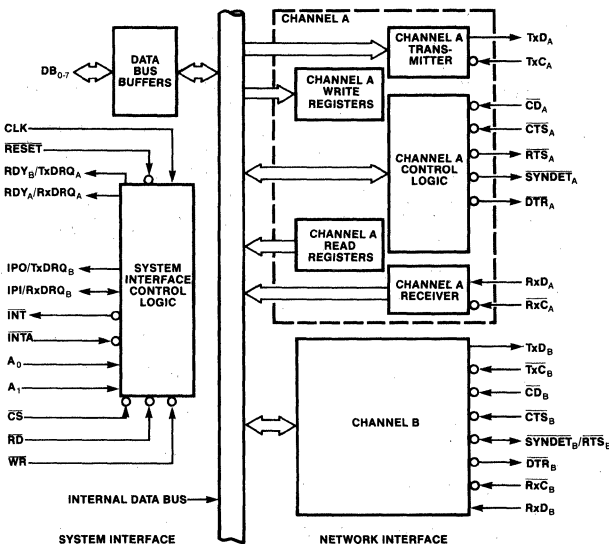


Figure 1. Block Diagram

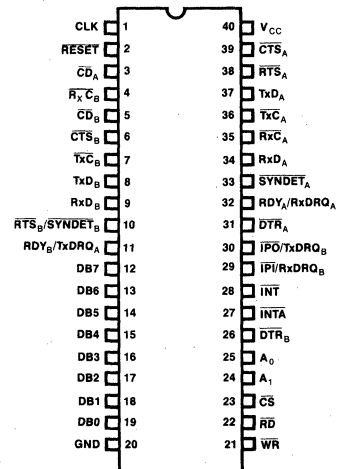


Figure 2. Pin Configuration

Table 1. Pin Description

Symbol	Pin No.	Type	Name and Function
CLK	1	I	<b>Clock:</b> System clock, TTL compatible.
RESET	2	I	<b>Reset:</b> A low signal on this pin will force the MPSC to an idle state. TxDA and TxD <sub>B</sub> are forced high. The modem interface output signals are forced high. The MPSC will remain idle until the control registers are initialized. Reset must be true for one complete CLK cycle.
CD <sub>A</sub>	3	I	<b>Carrier Detect (Channel A):</b> Carrier Detect (Channel A) signals that the line transmission has started. The MPSC will begin to sample data on the RxDA line if modem enables are selected.
RxC <sub>B</sub>	4	I	<b>Receiver Clock:</b> The Receiver Clock (Channel B) clocks in data on the RxDB pin.
CD <sub>B</sub>	5	I	<b>Carrier Detect (Channel B):</b> Carrier Detect (Channel B) signals that the line transmission has started. The MPSC will begin to sample data on the RxDB line if modem enables are selected.
CTS <sub>B</sub>	6	I	<b>Clear To Send (Channel B):</b> Clear To Send (Channel B) signals that the modem is ready to accept data from the MPSC. Clear To Send will enable Channel B transmitter if modem enables are selected, otherwise this pin may be used as a general purpose input.
TxC <sub>B</sub>	7	I	<b>Transmit Clock (Channel B):</b> Transmit Clock (Channel B) for TxDB pin.
TxD <sub>B</sub>	8	O	<b>Transmit Data (Channel B):</b> This line transmits the serial data to the communications channel (Channel B).
RxD <sub>B</sub>	9	I	<b>Receive Data (Channel B):</b> This line receives serial data from the communications channel (Channel B).
SYNDET <sub>B</sub> / RTS <sub>B</sub>	10	I/O	<b>Synchronous Detection (Channel B):</b> This pin is used in byte synchronous mode as either an internal sync detect (output) or as a means to force external synchronization (input). In SDLC mode, this pin is an output indicating Flag detection. In asynchronous mode it is a general purpose input (Channel B). Request To Send (Channel B) is a general purpose output, generally used to signal that Channel B is ready to send data.

Symbol	Pin No.	Type	Name and Function
RDY <sub>B</sub> / TxDRQ <sub>A</sub>	11	O	<b>Ready Transmit Data:</b> In mode 0 this pin is used to synchronize data transfers for both Receive and Transmit of Channel B to the controlling processor's READY line (open collector). In modes 1 and 2 this pin requests a DMA transfer of data for a transmit operation (Channel A).
DB7	12	I/O	<b>Data Bus:</b> The Data Bus lines are bi-directional three state lines which interface with the system's Data Bus.
DB6	13		
DB5	14		
DB4	15		
DB3	16		
DB2	17		
DB1	18		
DB0	19		
GND	20		<b>Ground.</b>
V <sub>cc</sub>	40		<b>Power:</b> +5V Supply.
CTS <sub>A</sub>	39	I	<b>Clear To Send (Channel A):</b> This signals that the modem is ready to accept data from the MPSC. Clear To Send will enable Channel B transmitter if modem enables are selected, otherwise this pin may be used as a general purpose input.
RTS <sub>A</sub>	38	O	<b>Request To Send (Channel A):</b> Request To Send (Channel A) is a general purpose output generally used to signal that Channel A is ready to send data.
TxD <sub>A</sub>	37	O	<b>Transmit Data (Channel A):</b> This line transmits the serial data to the communications channel (Channel A).
TxC <sub>A</sub>	36	I	<b>Transmitter Clock (Channel A):</b> The transmitter clock (Channel A) clocks out data on the TxDA pin.
RxC <sub>A</sub>	35	I	<b>Receiver Clock (Channel A):</b> The receiver clock (Channel A) clocks in data on the RxDA pin.
RxD <sub>A</sub>	34	I	<b>Receive Data (Channel A):</b> This line receives serial data from the communications channel (Channel A).
SYNDET <sub>A</sub>	33	I/O	<b>Synchronous Detection (Channel A):</b> This pin is used in byte synchronous mode as either an internal sync detect (output) or as a means to force external synchronization (input). In SDLC mode, this pin is an output indicating flag detection. In asynchronous mode it is a general purpose input (Channel A).

Table 1. Pin Description (Continued)

Symbol	Pin No.	Type	Name and Function
$\overline{\text{RDY}}_A$ / RxDRQ <sub>A</sub>	32	O	<b>Ready:</b> In mode 0 this pin is used to synchronize data transfers for both receive and transmit of Channel A to the controlling processor's READY line (open collector). In modes 1 and 2 RxDRQ <sub>A</sub> requests a DMA transfer of data for a receive operation for Channel A.
DTR <sub>A</sub>	31	O	<b>Data Terminal Ready:</b> This pin is Data Terminal Ready (Channel A) which is a general purpose output.
IPO/ TxDRQ <sub>B</sub>	30	O	<b>Interrupt Priority Out:</b> In modes 0 and 1, IPO is Interrupt Priority Out. It is used to establish a hardware interrupt priority scheme with IPI. It is low only if IPI is low and the controlling processor is not servicing an interrupt from this MPSC. In mode 2, TxDRQ <sub>B</sub> requests a DMA transfer of data for a transmit operation for Channel B.
IPI/ RxDRQ <sub>B</sub>	29	I/O	<b>Interrupt Priority In:</b> In modes 0 and 1, IPI is Interrupt Priority In. A low on IPI means that no higher priority device is being serviced by the controlling processor's interrupt service routine. In mode 2, RxDRQ <sub>B</sub> requests a DMA transfer of data for a receive operation for Channel B.

Symbol	Pin No.	Type	Name and Function
$\overline{\text{INT}}$	28	O	<b>Interrupt:</b> The interrupt signal indicates that the highest priority internal interrupt requires service (open collector). Priority can be resolved via an external interrupt controller or a daisy-chain scheme.
$\overline{\text{INTA}}$	27	I	<b>Interrupt Acknowledge:</b> This Interrupt Acknowledge allows the highest priority interrupting device to generate an interrupt vector.
$\overline{\text{DTR}}_B$	26	O	<b>Data Terminal Ready (Channel B):</b> This is a general purpose output.
A <sub>0</sub>	25	I	<b>Address:</b> This line selects Channel A or B during data or command transfers. A low selects Channel A.
A <sub>1</sub>	24	I	<b>Address:</b> This line selects between data or command information transfer. A low means data.
$\overline{\text{CS}}$	23	I	<b>Chip Select:</b> Chip Select enables RD or WR.
$\overline{\text{RD}}$	22	I	<b>Read:</b> Read controls a data byte or status byte transfer from the MPSC to CPU.
WR	21	I	<b>Write:</b> Write controls transfer of data or commands to the MPSC.

## GENERAL DESCRIPTION

The Intel® 8274 Multi-Protocol Serial Controller is a microcomputer peripheral device which supports Asynchronous (Start/Stop), Byte Synchronous (Monosync, IBM Bisync), and Bit Synchronous (ISO's HDLC, IBM's SDLC) protocols. This controller's flexible architecture allows easy implementation of many variations of these three protocols with low software and hardware overhead.

The Multi-Protocol Serial Controller (MPSC) implements two independent serial receiver/transmitter channels.

The MPSC supports several microprocessor interface options; Polled, Wait, Interrupt driven and DMA driven. The MPSC is designed to support Intel's® MCS-85 and iAPX 86, 88 families.

This data sheet will describe the serial protocol functions, the microprocessor interface, a detailed register and command description, general system operations, specifications, and waveforms.

## FUNCTIONAL DESCRIPTION

This section of the data sheet describes how the Asynchronous and Synchronous protocols are implemented in the MPSC. It describes general considerations, transmit operation, and receive operation for Asynchronous, Byte Synchronous, and Bit Synchronous protocols.

## ASYNCHRONOUS OPERATIONS

### General

For operation in the asynchronous mode, the MPSC must be initialized with the following information: character length (WR3; D7, D6 and WR5; D6, D5), clock rate (WR4; D7, D6), number of stop bits (WR4; D3, D2), odd, even or no parity (WR4; D1, D0), interrupt mode (WR1, WR2), and receiver (WR3; D0) or transmitter (WR5; D3) enable. When loading these parameters into the MPSC, WR4 information must

be written before the WR1, WR3, WR5 parameters/commands. (See Detailed Command Description Section).

For transmission via a modem or RS232C interface, the Request To Send (RTS) (WR5; D1) and Data Terminal Ready (DTR) (WR5; D7) bits must be set along with the Transmit Enable bit (WR5; D3). Setting the Auto Enables (WR3; D5) bit allows the programmer to send the first character of the message without waiting for a clear to send (CTS).

Both the Framing Error and Receive Overrun Error flags are latched and cause an interrupt, i.e., if status affects vector (WR1B; D2) is selected, the interrupt vector indicates a special Receive condition.

If the External/Status Interrupt bit (WR1; D0) is enabled, Break Detect (RR0; D7) and Carrier Detect (RR0; D3) will cause an interrupt. Reset External/Status Interrupts (WR0; D5, D4, D3) will clear Break Detect and Carrier Detect bits if they are set.

A status read after a data read will include error status for the next word in the buffer. If the Interrupt on First Character (WR1; D4, D3) is selected, then data and error status are held until an Error Reset command (WR0; D5, D4, D3) is given.

If the Interrupt on Every Character Mode bit (WR1; D4, D3) is selected, the interrupt vector is different if there is an error status in RR1. When the character is read, the error status bit is set and the Special Receive Condition vector is returned if Status Affects vector (WR1B; D2) is selected.

In a polled environment, the Receive Character Available bit (RR0; D0) must be monitored so that the CPU can determine when data is available. The bit is reset automatically when the data is read.

If the X1 clock mode is selected, the bit synchronization must be accomplished externally.

### Transmit

The transmit function begins when the Transmit Enable bit (WR5; D3) is set. The MPSC automatically adds the start bit, the programmed parity bit (odd, even or no parity) and the programmed number of stop bits (1, 1.5 or 2 bits) to the data character being transmitted.

The Serial data is shifted out from the Transmit Data (TxD) output on the falling edge of the Transmit Clock (TxCl) input, at a rate programmable to 1, 1/16, 1/32nd, or 1/64th of the clock rate supplied to the TxCl input.

The TxD output is held high when the transmitter has no data to send, unless, under program control, the Send Break (WR5; D4) command is issued to hold the TxD low.

If the External/STATUS Interrupt bit (WR1; D0) is set, the status of CD, CTS and SYNDET are monitored, and, if any changes occur for a period of time greater than the minimum specified pulse width, an interrupt is generated. CTS is usually monitored using this interrupt feature.

If the Auto Enables (WR; D5) option is selected the programmer need not wait for the CTS before sending the first character. The MPSC will automatically wait for the CTS pin to go active before the transmission begins.

The Transmit Buffer Empty bit (RR0; D2) is set by the MPSC when the data byte from the buffer is loaded in the transmit shift register. The data is written to the MPSC only when the Tx buffer becomes empty to prevent overwriting.

**Asynchronous Mode Register Setup**

	D7	D6	D5	D4	D3	D2	D1	D0
<b>WR3</b>	00 Rx 5 b/char 01 Rx 7 b/char 10 Rx 6 b/char 11 Rx 8 b/char		AUTO ENABLES	0	0	0	0	Rx ENABLE
<b>WR4</b>	00 X1 Clock 01 X16 Clock 10 X32 Clock 11 X64 Clock		0	0	00 ENABLE SYNC MODES 01 1 STOP BIT 10 1½ STOP BITS 11 2 STOP BITS		EVEN/ ODD PARITY	PARITY ENABLE
<b>WR5</b>	DTR	00 Tx 5 b/char 01 Tx 7 b/char 10 Tx 6 b/char 11 Tx 8 b/char		SEND BREAK	Tx ENABLE	0	RTS	0

## Receive

The receive function begins when the Receive Enable (WR3; D0) bit is set. If the Auto Enables (WR3; D5) option is selected, then Carrier Detect (CD) must also be low. A valid start bit is detected if a low persists for at least 1/2 bit time on the Receive Data (RxD) input.

The data is sampled at mid-bit time, on the rising edge of RxC, until the entire character is assembled. The receiver inserts 1's when a character is less than 8 bits. If parity (WR4; D1, D0) is enabled and the character is less than 8 bits the parity bit is not stripped from the character.

The receiver also stores error status for each of the 3 data characters in the data buffer. When a parity error is detected, the parity error flag (RR1; D4) is set and remains set until it is reset by the Error Reset command (WR0; D5, D4, D3).

When a character is assembled without a stop bit being detected, the Framing Error bit (RR1; D6) is set. The detection of a Framing Error adds an additional 1/2 bit time to the character time so the Framing Error is not interpreted as a new start bit.

If the CPU fails to read a data character while more than three characters have been received, the Receive Overrun bit (RR1; D5) is set. Only the overwritten character is flagged with the Receive Overrun bit. When this occurs, the fourth character assembled replaces the third character in the receive buffers. The Receive Overrun bit (RR1; D5) is reset by the Error Reset command (WR0; D5, D4, D3).

## SYNCHRONOUS OPERATION— MONO SYNC, BI SYNC

### General

The MPSC must be initialized with the following parameters: odd or even parity (WR4; D1,D0), X1 clock mode (WR4; D7, D6), 8- or 16-bit sync character (WR4; D5, D4), CRC polynomial (WR5; D2), Transmitter Enable (WR5; D3), interrupt modes (WR1, WR2), transmit character length (WR5; D6, D5) and receive character length (WR3; D7, D6). WR4 parameters must be written before WR1, WR3, WR5, WR6 and WR7.

The data is transmitted on the falling edge of the Transmit Clock, (TxC) and is received on the rising edge of Receive Clock (RxC). The X1 clock is used for both transmit and receive operations for all three sync modes: Mono, Bi and External.

### Transmit Set-Up—Monosync, Bisync

Transmit data is held high after channel reset, or if the transmitter is not enabled. A break may be programmed to generate a spacing line that begins as soon as the Send Break (WR5; D4) bit is set. With the transmitter fully initialized and enabled, the default condition is continuous transmission of the 8- or 16-bit sync character.

Using interrupts for data transfer requires that the Transmit Interrupt/DMA Enable bit (WR1; D1) be set. An interrupt is generated each time the transmit buffer becomes empty. The interrupt can be satisfied

**Synchronous Mode Register Setup—Monosync, Bisync**

	D7	D6	D5	D4	D3	D2	D1	D0
<b>WR3</b>	00 Rx 5 b/char 01 Rx 7 b/char 10 Rx 6 b/char 11 Rx 8 b/char		AUTO ENABLES	ENTER HUNT MODE	Rx CRC ENABLE	0	SYNC CHAR LOAD INHIBIT	Rx ENABLE
<b>WR4</b>	0	0	00 8 bit Sync 01 16 bit Sync 11 Ext Sync		0	0	EVEN/ ODD PARITY	PARITY ENABLE
<b>WR5</b>	DTR	00 Tx 5 b/char 01 Tx 7 b/char 10 Tx 6 b/char 11 Tx 8 b/char		SEND BREAK	Tx ENABLE	1 (SELECTS CRC-16)	RTS	Tx CRC ENABLE

either by writing another character into the transmitter or by resetting the Transmitter Interrupt/DMA Pending latch with a Reset Transmitter Interrupt/DMA Pending Command (WR0; D5, D4, D3). If nothing more is written into the transmitter, there can be no further Transmit Buffer Empty interrupt, but this situation does cause a Transmit Underrun condition (RR0; D6).

Data Transfers using the RDY signal are for software controlled data transfers such as block moves. RDY tells the CPU that the MPSC is not ready to accept/provide data and that the CPU must extend the output/input cycle. DMA data transfers use the TxDRQ A/B signals which indicate that the transmit buffer is empty, and that the MPSC is ready to accept the next data character. If the data character is not loaded into the MPSC by the time the transmit shift register is empty, the MPSC enters the Transmit Underrun condition.

The MPSC has two programmable options for solving the transmit underrun condition: it can insert sync characters, or it can send the CRC characters generated so far, followed by sync characters. Following a chip or channel reset, the Transmit Underrun/EOM status bit (RR0; D6) is in a set condition allowing the insertion of sync characters when there is no data to send. The CRC is not calculated on these automatically inserted sync characters. When the CPU detects the end of message, a Reset Transmit Underrun/EOM command can be issued. This allows CRC to be sent when the transmitter has no data to send.

In the case of sync insertion, an interrupt is generated only after the first automatically inserted sync character has been loaded in Transmit Shift Register. The status indicates the Transmit Underrun/EOM bit and the Transmit Buffer Empty bit are set.

In the case of CRC insertion, the Transmit Underrun/EOM bit is set and the Transmit Buffer Empty bit is reset while CRC is being sent. When CRC has been completely sent, the Transmit Buffer Empty status bit is set and an interrupt is generated to indicate to the CPU that another message can begin (this interrupt occurs because CRC has been sent and sync has been loaded into the Tx Shift Register). If no more messages are to be sent, the program can terminate transmission by resetting RTS, and disabling the transmitter (WR5; D3).

**Bisync CRC Generation.** Setting the Transmit CRC enable bit (WR5; D0) indicates CRC accumulation when the program sends the first data character to

the MPSC. Although the MPSC automatically transmits up to two sync characters (16 bit sync), it is wise to send a few more sync characters ahead of the message (before enabling Transmit CRC) to ensure synchronization at the receiving end.

The Transmit CRC Enable bit can be changed on the fly any time in the message to include or exclude a particular data character from CRC accumulation. The Transmit CRC Enable bit should be in the desired state when the data character is loaded from the transmit shift register. To ensure this bit in the proper state, the Transmit CRC Enable bit must be issued before sending the data character to the MPSC.

**Transmit Transparent Mode.** Transparent mode (Bisync protocol) operation is made possible by the ability to change Transmit CRC Enable on the fly and by the additional capability of inserting 16 bit sync characters. Exclusion of DLE characters from CRC calculation can be achieved by disabling CRC calculation immediately preceding the DLE character transfer to the MPSC.

In the transmit mode, the transmitter always sends the programmed number of sync bits (8 or 16) (WR4; D5, D4). When in the Monosync mode, the transmitter sends from WR6 and the receiver compares against WR7. One of two CRC polynomials, CRC 16 or SDLC, may be used with synchronous modes. In the transmit initialization process, the CRC generator is initialized by setting the Reset Transmit CRC Generator command (WR0; D7, D6).

The External/Status interrupt (WR1; D0) mode can be used to monitor the status of the CTS input as well as the Transmit Underrun/EOM latch. Optionally, the Auto Enable (WR3; D5) feature can be used to enable the transmitter when CTS is active. The first data transfer to the MPSC can begin when the External/Status interrupt occurs (CTS (RR0; D5) status bit set) following the Transmit Enable command (WR5; D3).

## Receive

After a channel reset, the receiver is in the Hunt phase, during which the MPSC looks for character synchronization. The Hunt begins only when the receiver is enabled and data transfer begins only when character synchronization has been achieved. If character synchronization is lost, the hunt phase can be re-entered by writing the Enter Hunt Phase (WR3; D4) bit. The assembly of received data continues until the MPSC is reset or until the receiver is

disabled (by command or by  $\overline{CD}$  while in the Auto Enables mode) or until the CPU sets the Enter Hunt Phase bit. Under program control, all the leading sync characters of the message can be inhibited from loading the receive buffers by setting the Sync Character Load Inhibit (WR3; D1) bit. After character synchronization is achieved the assembled characters are transferred to the receive data FIFO.

Data may be transferred with or without interrupts. Transferring data without interrupts is used for a purely polled operation or for off-line conditions. There are three interrupt modes available for data transfer: Interrupt on First Character Only, Interrupt on Every Character, and Special Receive Conditions Interrupt.

Interrupt on First Character Only mode is normally used to start a polling loop, a block transfer sequence using RDY to synchronize the CPU to the incoming data rate or a DMA transfer using the RxDRQ signal. The MPSC interrupts on the first character and thereafter only interrupts after a Special Receive Condition is detected. This mode can be reinitialized using the Enable Interrupt on Next Receive Character (WR0; D5, D4, D3) command which allows the next character received to generate an interrupt. Parity Errors do not cause interrupts, but End of Frame (SDLC operation) and Receive Overrun do cause interrupts in this mode. If the external status interrupts (WR1; D0) are enabled an interrupt may be generated any time the  $\overline{CD}$  changes state.

Interrupt On Every Character mode generates an interrupt whenever a character enters the receive buffer. Errors and Special Receive Conditions generate a special vector if the Status Affects Vector (WR1B; D2) is selected. Also the Parity Error may be

programmed (WR1; D4, D3) not to generate the special vector while in the Interrupt On Every Character mode.

The Special Receive Condition interrupt can only occur while in the Receive Interrupt On First Character Only or the Interrupt On Every Receive Character modes. The Special Receive Condition interrupt is caused by the Receive Overrun (RR1; D5) error condition. The error status reflects an error in the current word in the receive buffer, in addition to any Parity or Overrun errors since the last Error Reset (WR0; D5, D4, D3). The Receive Overrun and Parity error status bits are latched and can only be reset by the Error Reset (WR0; D5, D4, D3) command.

## SYNCHRONOUS OPERATION—SDLC

### General

Like the other synchronous operations the SDLC mode must be initialized with the following parameters: SDLC mode (WR4; D5, D4), SDLC polynomial (WR5; D2), Request to Send, Data Terminal Ready, transmit character length (WR5; D6, D5), interrupt modes (WR1; WR2), Transmit Enable (WR5; D3), Receive Enable (WR3; D0), Auto Enable (WR3; D5) and External/Status Interrupt (WR1; D0). WR4 parameters must be written before WR1, WR3, WR5, WR6 and WR7.

The Interrupt modes for SDLC operation are similar to those discussed previously in the synchronous operations section.

Synchronous Mode Register Setup—SDLC/HDLC

	D7	D6	D5	D4	D3	D2	D1	D0
<b>WR3</b>	00 Rx 5b/char 01 Rx 7b/char 10 Rx 6b/char 11 Rx 8b/char		AUTO ENABLES	ENTER HUNT MODE	Rx CRC ENABLE	ADDRESS SEARCH MODE	0	Rx ENABLE
<b>WR4</b>	0	0	1 0 (SELECTS SDLC/ HDLC MODE)		0	0	0	0
<b>WR5</b>	DTR	00 Tx ≤5b/char 01 Tx 7b/char 10 Tx 6b/char 11 Tx 8b/char		0	Tx ENABLE	0 (SELECTS SDLC/ HDLC CRC)	RTS	Tx CRC ENABLE

## Transmit

After a channel reset, the MPSC begins sending SDLC flags.

Following the flags in an SDLC operation the 8-bit address field, control field and information field may be sent to the MPSC by the microprocessor. The MPSC transmits the Frame Check Sequence using the Transmit Underrun feature. The MPSC automatically inserts a zero after every sequence of 5 consecutive 1's except when transmitting Flags or Aborts.

SDLC—like protocols do not have provision for fill characters within a message. The MPSC therefore automatically terminates an SDLC frame when the transmit data buffer and output shift register have no more bits to send. It does this by sending the two bytes of CRC and then one or more flags. This allows very high-speed transmissions under DMA or CPU control without requiring the CPU to respond quickly to the end-of-message situation.

After a reset, the Transmit Underrun/EOM status bit is in the set state and prevents the insertion of CRC characters during the time there is no data to send. Flag characters are sent. The MPSC begins to send the frame when data is written into the transmit buffer. Between the time the first data byte is written, and the end of the message, the Reset Transmit Underrun/EOM (WR0; D5, D4, D1) command must be issued. The Transmit Underrun/EOM status bit (RR0; D6) is in the reset state at the end of the message which automatically sends the CRC characters.

The MPSC may be programmed to issue a send Abort command (WR0; D5, D4, D3). This command causes at least eight 1's but less than fourteen 1's to be sent before the line reverts to continuous flags.

## Receive

After initialization, the MPSC enters the Hunt phase, and remains in the Hunt phase until the first Flag is received. The MPSC never again enters the Hunt phase unless the microprocessor writes the Enter Hunt command.

The MPSC can be programmed to receive all frames or it can be programmed to the Address Search Mode. In the Address Search Mode, only frames with addresses that match the value in WR6 or the global address (OFFH) are received by the MPSC. Extended address recognition must be done by the microprocessor software.

The control and information fields are received as data.

SDLC/HDLC CRC calculation does not have an 8-bit delay, since all characters are included in the calculation, unlike Byte Synchronous Protocols.

Reception of an abort sequence (7 or more 1's) will cause the Break/Abort bit (RR0; D7) to be set and will cause an External/Status interrupt, if enabled. After the Reset External/Status Interrupts Command has been issued, a second interrupt will occur at the end of the abort sequence.

## MPSC

### Detailed Command Description

#### GENERAL

The MPSC supports an extremely flexible set of serial and system interface modes.

The system interface to the CPU consists of 8 ports or buffers:

$\overline{CS}$	$A_0$	$A_1$	Read Operation	Write Operation
0	0	0	Ch. A Data Read	Ch. A Data Write
0	0	1	Ch. A Status Read	Ch. A Command/Parameter
0	1	0	Ch. B Data Read	Ch. B Data Write
0	1	1	Ch. B Status Read	Ch. B Command/Parameter
1	X	X	High Impedance	High Impedance

Data buffers are addressed by  $A_1 = 0$ , and Command ports are addressed by  $A_1 = 1$ .

Command, parameter, and status information is held in 22 registers within the MPSC (8 write registers and 3 read registers for each channel). They are all accessed via the command ports.

An internal pointer register selects which of the command or status registers will be read or written during a command/status access of an MPSC channel.



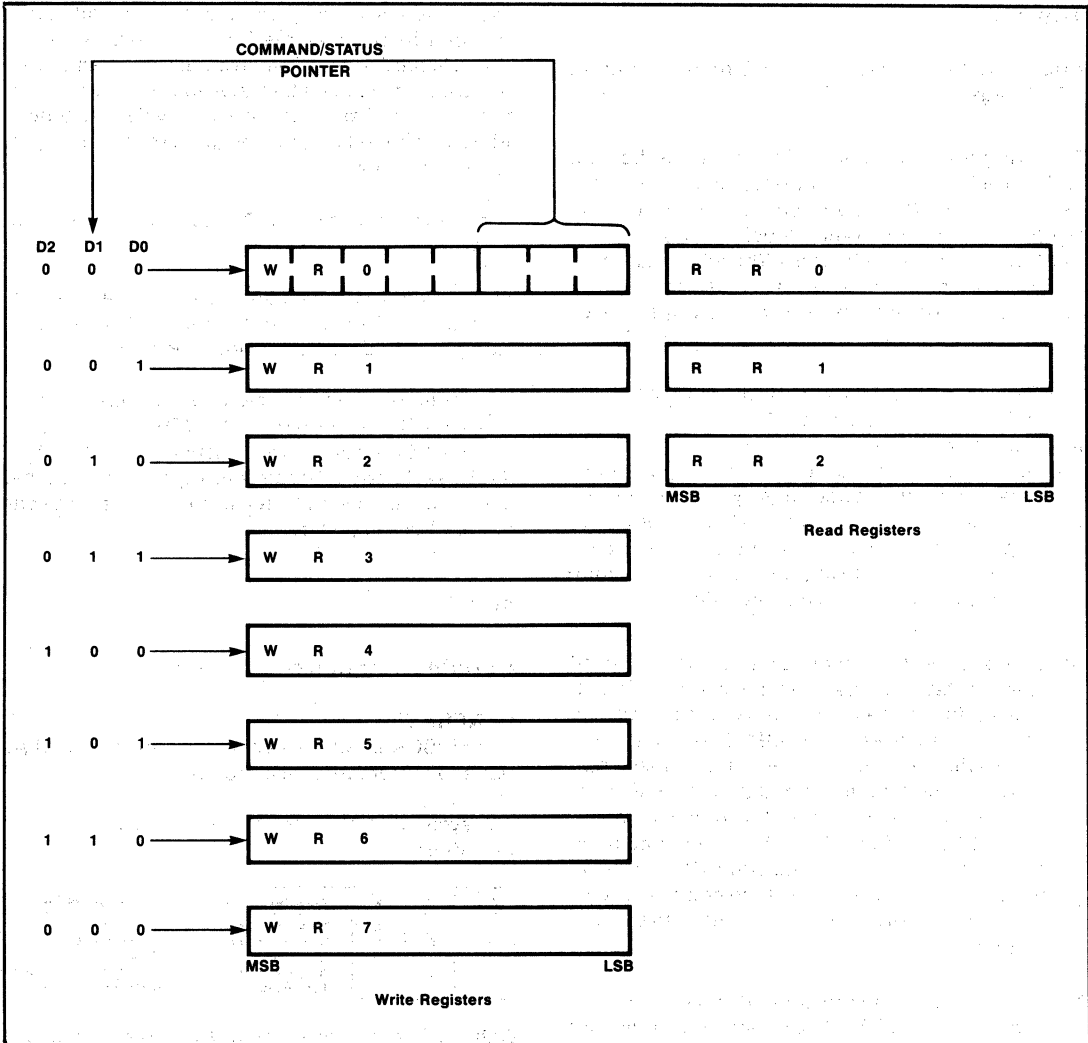


Figure 3. Command/Status Register Architecture (each serial channel)

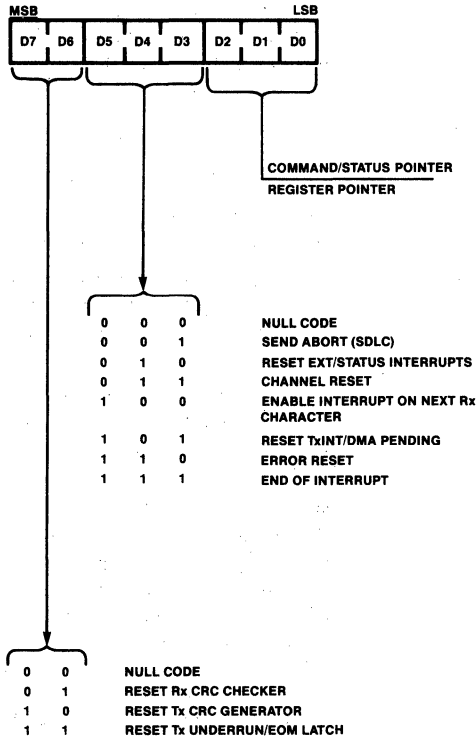
After reset, the contents of the pointer register are zero. The first write to a command register causes the data to be loaded into Write Register 0 (WR0). The three least significant bits of WR0 are loaded

into the Command/Status Pointer. The next read or write operation accesses the read or write register selected by the pointer. The pointer is reset after the read or write operation is completed.

**COMMAND/STATUS DESCRIPTION**

The following command and status bytes are used during initialization and execution phases of operation. All Command/Status operations on the two channels are identical, and independent, except where noted.

**Write Register 0 (WR0):**



**Detailed Register Description**

**WR0**

D2, D1, D0—Command/Status Register Pointer bits determine which write-register the next byte is to be written into, or which read-register the next byte is to be read from. After reset, the first byte written into either channel goes into WR0. Following a read or write to any register (except WR0) the pointer will point to WR0.

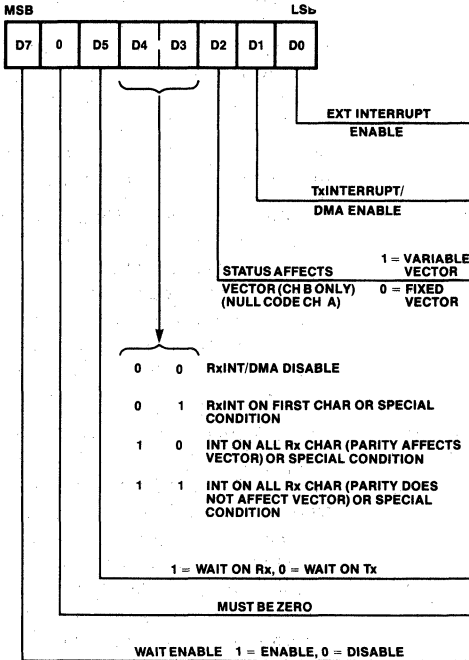
D5, D4, D3—Command bits determine which of the basic seven commands are to be performed.

- Command 0 Null—has no effect.
  - Command 1 Send Abort—causes the generation of eight to thirteen 1's when in the SDLC mode.
  - Command 2 Reset External/Status Interrupts—resets the latched status bits of RR0 and re-enables them, allowing interrupts to occur again.
  - Command 3 Channel Reset—resets the Latched Status bits of RR0, the interrupt prioritization logic and all control registers for the channel. Four extra system clock cycles should be allowed for MPSC reset time before any additional commands or controls are written into the channel.
  - Command 4 Enable Interrupt on Next Receive Character—if the Interrupt on First Receive Character mode is selected, this command reactivates that mode after each complete message is received to prepare the MPSC for the next message.
  - Command 5 Reset Transmitter Interrupt/DMA Pending—if The Transmit Interrupt/DMA Enable mode is selected, the MPSC automatically interrupts or requests DMA data transfer when the transmit buffer becomes empty. When there are no more characters to be sent, issuing this command prevents further transmitter interrupts or DMA requests until the next character has been completely sent.
  - Command 6 Error Reset—error latches, Parity and Overrun errors in RR1 are reset.
  - Command 7 End of Interrupt—resets the interrupt-in-service latch of the highest-priority internal device under service.
- | D7, D6 | Command   |
|--------|---|
| 00     | Null—has no effect.   |
| 01     | Reset Receive CRC Checker—resets the CRC checker to 0's. If in SDLC mode the CRC checker is initialized to all 1's. |

- 10 Reset Transmit CRC Generator —resets the CRC generator to 0's. If in SDLC mode the CRC generator's initialized to all 1's.
- 11 Reset Tx Underrun/End of Message Latch.

- D1 Transmitter Interrupt/DMA Enable —allows the MPSC to interrupt or request a DMA transfer when the transmitter buffer becomes empty.
- D2 Status Affects vector—(WR1, D2 active in channel B only.) If this bit is not set, then the fixed vector, programmed in WR2, is returned from an interrupt acknowledge sequence. If the bit is set then the vector returned from an interrupt acknowledge is variable as shown in the Interrupt Vector Table.

**Write Register 1 (WR1):**

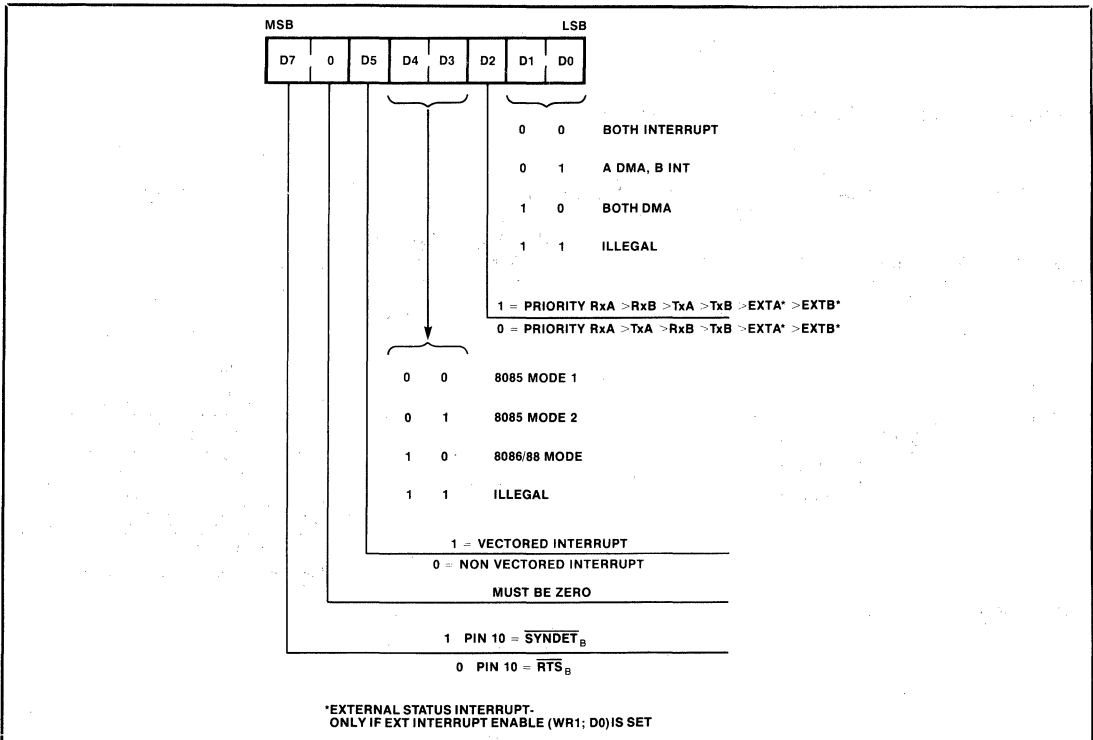


- D4, D3 Receive Interrupt Mode
  - 0 0 Receive Interrupts/DMA Disabled
  - 0 1 Receive Interrupt on First Character Only or Special Condition
  - 1 0 Interrupt on All Receive Characters or Special Condition (Parity Error is a Special Receive Condition)
  - 1 1 Interrupt on All Receive Characters or Special Condition (Parity Error is not a Special Receive Condition).
- D5 Wait on Receive/Transmit—when the following conditions are met the RDY pin is activated, otherwise it is held in the High-Z state. (Conditions: Interrupt Enabled Mode, Wait Enabled, CS = 0, A0 = 0/1, and A1 = 0). The RDY pin is pulled low when the transmitter buffer is full or the receiver buffer is empty and it is driven High when the transmitter buffer is empty or the receiver buffer is full. The RDY<sub>A</sub> and RDY<sub>B</sub> may be wired OR connected since only one signal is active at any one time while the other is in the High Z state.
- D6 Must be Zero
- D7 Wait Enable—enables the wait function.

- D0 External/Status Interrupt Enable —allows interrupt to occur as the result of transitions on the CD, CTS or SYNDET inputs. Also allows interrupts as the result of a Break/Abort detection and termination, or at the beginning of CRC, or sync character transmission when the Transmit Underrun/EOM latch becomes set.

<b>WR2</b>	<b>Channel A</b>	D5, D4, D3	Interrupt Code—specifies the behavior of the MPSC when it receives an interrupt acknowledge sequence from the CPU. (See Interrupt Vector Mode Table).
D1, D0	System Configuration—These specify the data transfer from MPSC channels to the CPU, either interrupt or DMA based.	0 X X	Non-vectored interrupts—intended for use with external DMA CONTROLLER. The Data Bus remains in a high impedance state during INTA sequences.
0 0	Channel A and Channel B both use interrupts	1 0 0	8085 Vector Mode 1—intended for use as the primary MPSC in a daisy chained priority structure. (See System Interface section)
0 1	Channel A uses DMA, Channel B uses interrupt	1 0 1	8085 Vector Mode 2—intended for use as any secondary MPSC in a daisy chained priority structure. (See System Interface section)
1 0	Channel A and Channel B both use DMA	1 1 0	8086/88 Vector Mode—intended for use as either a primary or secondary in a daisy chained priority structure. (See System Interface section)
1 1	Illegal Code	D7, D6	Must be zero.
D2	Priority—this bit specifies the relative priorities of the internal MPSC interrupt/DMA sources.		
0	(Highest) RxA, TxA, RxB, TxB ExTA, ExTB (Lowest)		
1	(Highest) RxA, RxB, TxA, TxB, ExTA, ExTB (Lowest)		

**Write Register 2 (WR2): Channel A**



The following table describes the MPSC's response to an interrupt acknowledge sequence:

D5	D4	D3	$\overline{IP\bar{I}}$	MODE	INTA	Data Bus
0	X	X	X	Non-vectored	Any INTA	D7 High Impedance D0
1	0	0	0	85 Mode 1	1st INTA 2nd INTA 3rd INTA	1 1 0 0 1 1 0 1 V7 V6 V5 V4* V3* V2* V1 V0 0 0 0 0 0 0 0 0
1	0	0	1	85 Mode 1	1st INTA 2nd INTA 3rd INTA	1 1 0 0 1 1 0 1 High Impedance High Impedance
1	0	1	0	85 Mode 2	1st INTA 2nd INTA 3rd INTA	High Impedance High Impedance High Impedance
1	1	0	0	86 Mode	1st INTA	High Impedance V7 V6 V5 V4 V3 V2* V1*V0*
1	1	0	1	86 Mode	1st INTA 2nd INTA	High Impedance High Impedance

\*These bits are variable if the "status affects vector" mode has been programmed, (WR1B, p.2).

**Interrupt/DMA Mode, Pin Functions, and Priority**

Ch. A WR2			Int/DMA Mode		RDY <sub>A</sub> / RxDRQ <sub>A</sub> Pin 32	Pin Functions			Priority	
D <sub>2</sub>	D <sub>1</sub>	D <sub>0</sub>	CH. A	CH. B		RDY <sub>B</sub> / TxDRQ <sub>A</sub> Pin 11	$\overline{IP\bar{I}}$ / RxDRQ <sub>B</sub> Pin 29	$\overline{IP\bar{O}}$ / TxDRQ <sub>B</sub> Pin 30	Highest	Lowest
0	0	0	INT	INT	RDY <sub>A</sub>	RDY <sub>B</sub>	$\overline{IP\bar{I}}$	$\overline{IP\bar{O}}$	Rx <sub>A</sub> , Tx <sub>A</sub> , Rx <sub>B</sub> , Tx <sub>B</sub> , EXT <sub>A</sub> , EXT <sub>B</sub>	
1	0	0	INT	INT					Rx <sub>A</sub> , Rx <sub>B</sub> , Tx <sub>A</sub> , Tx <sub>B</sub> , EXT <sub>A</sub> , EXT <sub>B</sub>	
0	0	1	DMA	INT	RxDRQ <sub>A</sub>	TxDRQ <sub>A</sub>	$\overline{IP\bar{I}}$	$\overline{IP\bar{O}}$	Rx <sub>A</sub> , Tx <sub>A</sub> (DMA)	
1	0	1	DMA	INT					Rx <sub>A</sub> <sup>1</sup> , Rx <sub>B</sub> , Tx <sub>B</sub> , EXT <sub>A</sub> , EXT <sub>B</sub> (INT)	
0	1	0	DMA	DMA	RxDRQ <sub>A</sub>	TxDRQ <sub>A</sub>	RxDRQ <sub>B</sub>	TxDRQ <sub>B</sub>	Rx <sub>A</sub> , Tx <sub>A</sub> , Rx <sub>B</sub> , Tx <sub>B</sub> (DMA)	
1	1	0	DMA	DMA					Rx <sub>A</sub> <sup>1</sup> , Rx <sub>B</sub> <sup>1</sup> , EXT <sub>A</sub> , EXT <sub>B</sub> (INT)	
0	1	0	DMA	DMA	RxDRQ <sub>A</sub>	TxDRQ <sub>A</sub>	RxDRQ <sub>B</sub>	TxDRQ <sub>B</sub>	Rx <sub>A</sub> , Tx <sub>A</sub> , Rx <sub>B</sub> , Tx <sub>B</sub> (DMA)	
1	1	0	DMA	DMA					Rx <sub>A</sub> <sup>1</sup> , Rx <sub>B</sub> <sup>1</sup> , EXT <sub>A</sub> , EXT <sub>B</sub> (INT)	

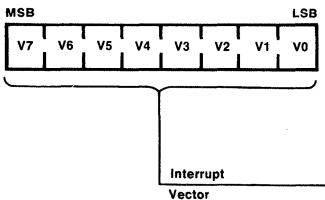
<sup>1</sup>Special Receive Condition

Interrupt Vector Mode Table

8085 Modes	V <sub>4</sub>	V <sub>3</sub>	V <sub>2</sub>	Channel	Condition
8086/88 Mode	V <sub>2</sub>	V <sub>1</sub>	V <sub>0</sub>		
Note 1: Special Receive Condition= Parity Error, Rx Overrun Error, Framing Error, End of Frame (SDLC)	0	0	0	B	Tx Buffer Empty Ext/Status Change Rx Char. Available Special Rx Condition (Note 1)
	0	0	1		
	0	1	0		
	0	1	1		
	1	0	0	A	Tx Buffer Empty Ext/Status Change Rx Char. Available Special Rx Condition (Note 1)
	1	0	1		
	1	1	0		
	1	1	1		

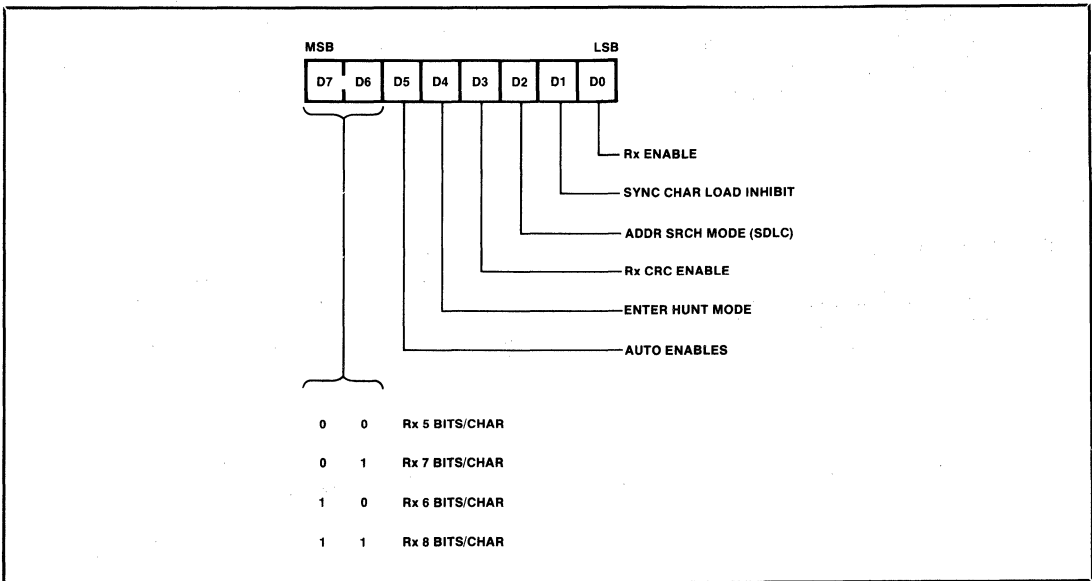
Write Register 2 (WR2): Channel B

WR2 CHANNEL B



D7-D0 Interrupt vector—This register contains the value of the interrupt vector placed on the data bus during interrupt acknowledge sequences.

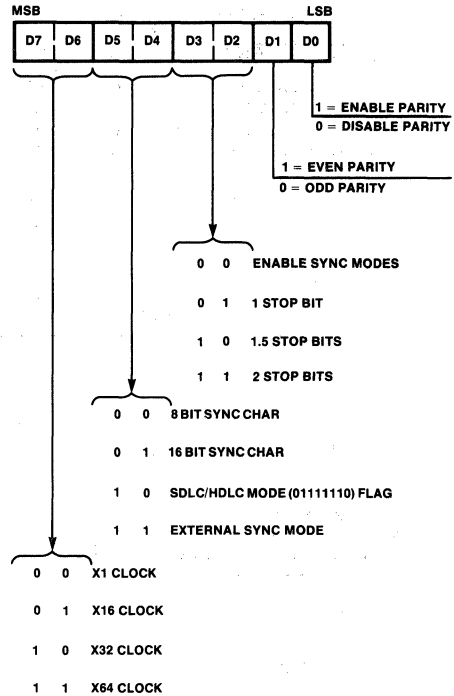
Write Register 3 (WR3):



**WR3**

- D0 Receiver Enable—A one enables the receiver to begin. This bit should be set only after the receiver has been initialized.
- D1 Sync Character Load Inhibit—A one prevents the receiver from loading sync characters into the receive buffers.
- D2 Address Search Mode—If the SDLC mode has been selected, the MPSC will receive all frames unless this bit is a 1. If this bit is a 1, the MPSC will receive only frames with address bytes that match the global address (OFFH) or the value loaded into WR6. This bit must be zero in non-SDLC modes.
- D3 Receive CRC Enable—A one in this bit enables (or re-enables) CRC calculation. CRC calculation starts with the last character placed in the Receiver FIFO. A zero in this bit disables, but does not reset, the Receiver CRC generator.
- D4 Enter Hunt Phase—After initialization, the MPSC automatically enters the Hunt mode. If synchronization is lost, the Hunt phase can be re-entered by writing a one to this bit.
- D5 Auto Enables—A one written to this bit causes  $\overline{CD}$  to be automatic enable signal for the receiver and  $\overline{CTS}$  to be an automatic enable signal for the transmitter. A zero written to this bit limits the effect of  $\overline{CD}$  and  $\overline{CTS}$  signals to setting/resetting their corresponding bits in the status register (RR0).
- D7, D6 Receive Character length
  - 0 0 Receive 5 Data bits/character
  - 0 1 Receive 7 Data bits/character
  - 1 0 Receive 6 Data bits/character
  - 1 1 Receive 8 Data bits/character

**Write Register 4 (WR4):**

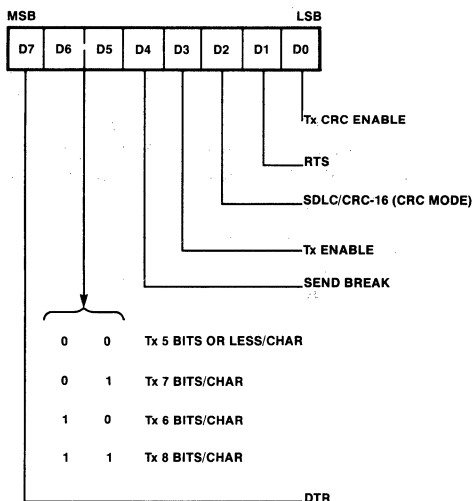


**WR4**

- D0 Parity—a one in this bit causes a parity bit to be added to the programmed number of data bits per character for both the transmitted and received character. If the MPSC is programmed to receive 8 bits per character, the parity bit is not transferred to the microprocessor. With other receiver character lengths, the parity bit is transferred to the microprocessor.
- D1 Even/Odd Parity—if parity is enabled, a one in this bit causes the MPSC to transmit and expect even parity, and a zero causes it to send and expect odd parity.
- D3, D2 Stop bits/sync mode

- 0 0 Selects synchronous modes.
- 0 1 Async mode, 1 stop bit/character
- 1 0 Async mode, 1-½ stop bits/character
- 1 1 Async mode, 2 stop bits/character
- D5, D4 Sync mode select
  - 0 0 8 bit sync character
  - 0 1 16 bit sync character
  - 1 0 SDLC mode (Flag sync)
  - 1 1 External sync mode
- D7, D6 Clock mode—selects the clock/data rate multiplier for both the receiver and the transmitter. 1x mode must be selected for synchronous modes. If the 1x mode is selected, bit synchronization must be done externally.
  - 0 0 Clock rate = Data rate x 1
  - 0 1 Clock rate = Data rate x 16
  - 1 0 Clock rate = Data rate x 32
  - 1 1 Clock rate = Data rate x 64

**Write Register 5 (WR5):**



**WR5**

- D0 Transmit CRC Enable—a one in this bit enables the transmitter CRC generator. The CRC calculation is done when a character is moved from the transmit buffer into the shift register. A zero in this bit disables CRC calculations. If this bit is not set when a transmitter underrun occurs, the CRC will not be sent.
  - D1 Request to Send—a one in this bit forces the RTS pin active (low) and zero in this bit forces the RTS pin inactive (high).
  - D2 CRC Select—a one in this bit selects the CRC - 16 polynomial ( $X^{16} + X^{15} + X^2 + 1$ ) and a zero in this bit selects the CCITT-CRC polynomial ( $X^{16} + X^{15} + X^5 + 1$ ).
  - D3 Transmitter Enable—a zero in this bit forces a marking state on the transmitter output. If this bit is set to zero during data or sync character transmission, the marking state is entered after the character has been sent. If this bit is set to zero during transmission of a CRC character, sync or flag bits are substituted for the remainder of the CRC bits.
  - D4 Send Break—a one in this bit forces the transmit data low. A one in this bit allows normal transmitter operation.
  - D6, D5 Transmit Character length
    - 0 0 Transmit 5 or less bits/character
    - 0 1 Transmit 7 bits/character
    - 1 0 Transmit 6 bits/character
    - 1 1 Transmit 8 bits/character
- Bits to be sent must be right justified least significant bit first, eg:
- |    |    |    |    |    |    |    |    |
|----|----|----|----|----|----|----|----|
| D7 | D6 | D5 | D4 | D3 | D2 | D1 | D0 |
| 0  | 0  | B5 | B4 | B3 | B2 | B1 | B0 |

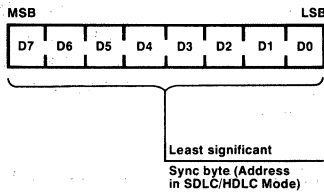


Five or less mode allows transmission of one to five bits per character. The microprocessor must format the data in the following way:

D7	D6	D5	D4	D3	D2	D1	D0	
1	1	1	1	1	1	1	B0	Sends one data bit
1	1	1	0	0	0	B1	B0	Sends two data bits
1	1	0	0	0	B2	B1	B0	Sends three data bits
1	0	0	0	B3	B2	B1	B0	Sends four data bits
0	0	0	B4	B3	B2	B1	B0	Sends five data bits

**D7** Data Terminal Ready—when set, this bit forces the  $\overline{\text{DTR}}$  pin active (low). When reset, this bit forces the DTR pin inactive (high).

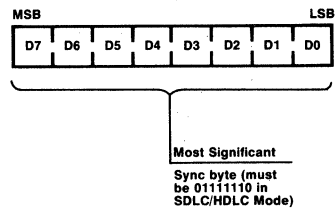
**Write Register 6 (WR6):**



**WR6**

**D7–D0** Sync/Address—this register contains the transmit sync character in Monosync mode, the low order 8 sync bits in Bisync mode, or the Address byte in SDLC mode.

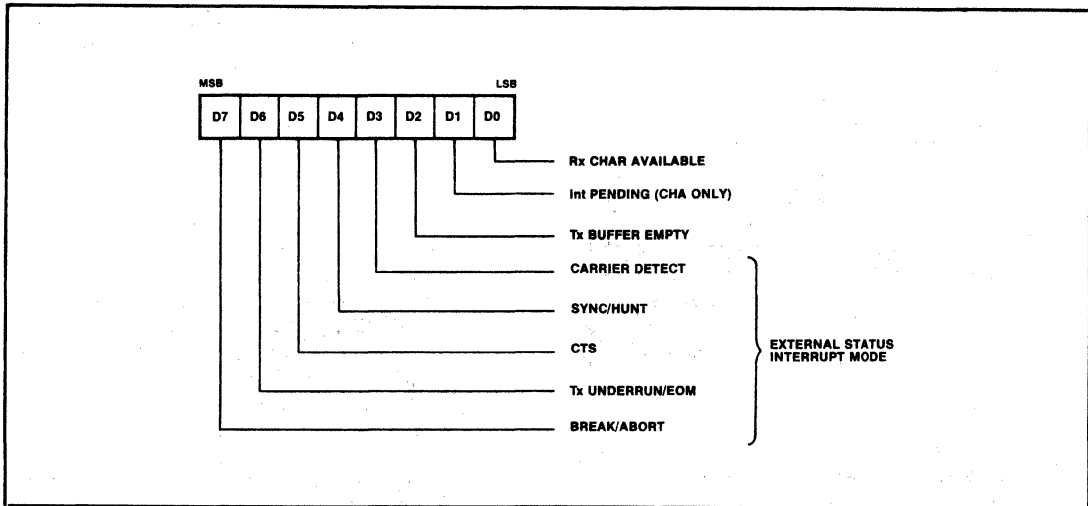
**Write Register 7 (WR7):**



**WR7**

**D7–D0** Sync/Flag—this register contains the receive sync character in Monosync mode, the high order 8 sync bits in Bisync mode, or the Flag character (01111110) in SDLC mode. WR7 is not used in External Sync mode.

## Read Register 0 (RR0):

**RR0**

- D0** Receive Character Available—this bit is set when the receive FIFO contains data and is reset when the FIFO is empty.
- D1** Interrupt Pending\*—This Interrupt-Pending bit is reset when an EOI command is issued and there is no other interrupt request pending at that time.
- D2** Transmit Buffer Empty—This bit is set whenever the transmit buffer is empty except when CRC characters are being sent in a synchronous mode. This bit is reset when the transmit buffer is loaded. This bit is set after an MPSC reset.
- D3** Carrier Detect—This bit contains the state of the CD pin at the time of the last change of any of the External/Status bits ( $\overline{CD}$ , CTS, Sync/Hunt, Break/Abort, or Tx Underrun/EOM). Any change of state of the  $\overline{CD}$  pin causes the CD bit to be latched and causes an External/Status interrupt. This bit indicates current state of the  $\overline{CD}$  pin immediately following a Reset External/Status Interrupt command.
- D4** Sync/Hunt—In asynchronous modes, the operation of this bit is similar to the CD status bit, except that Sync/Hunt shows the state of the SYNDET input. Any High-to-Low transition on the SYNDET pin sets this bit, and causes an External/Status interrupt (if enabled). The Reset External/Status Interrupt command is issued to clear the interrupt. A Low-to-High transition clears this bit and sets the External/Status interrupt. When the External/Status interrupt is set by the change in state of any other input or condition, this bit shows the inverted state of the SYNDET pin at time of the change. This bit must be read immediately following a Reset External/Status interrupt command to read the current state of the SYNDET input.

\*In vector mode this bit is set at the falling edge of the second INTA in an INTA cycle for an internal interrupt request. In non-vector mode, this bit is set at the falling edge of RD input after pointer 2 is specified. This bit is always zero in Channel B.

In the External Sync mode, the Sync/Hunt bit operates in a fashion similar to the Asynchronous mode, except the Enter Hunt Mode control bit enables the external sync detection logic. When the External Sync Mode and Enter Hunt Mode bits are set (for example, when the receiver is enabled following a reset), the SYNDET input must be held High by the external logic until external character synchronization is achieved. A High at the SYNDET input holds the Sync/Hunt status in the reset condition.

When external synchronization is achieved,  $\overline{\text{SYNDET}}$  must be driven Low on the second rising edge of  $\overline{\text{RxC}}$  after the rising edge of  $\overline{\text{RxC}}$  on which the last bit of the sync character was received. In other words, after the sync pattern is detected, the external logic must wait for two full Receive Clock cycles to activate the  $\overline{\text{SYNDET}}$  input. Once  $\overline{\text{SYNDET}}$  is forced Low, it is good practice to keep it Low until the CPU informs the external sync logic that synchronization has been lost or a new message is about to start. The High-to-Low transition of the  $\overline{\text{SYNDET}}$  output sets the Sync/Hunt bit, which sets the External/Status interrupt. The CPU must clear the interrupt by issuing the Reset External/Status Interrupt Command.

When the  $\overline{\text{SYNDET}}$  input goes High again, another External/Status interrupt is generated that must also be cleared. The Enter Hunt Mode control bit is set whenever character synchronization is lost or the end of message is detected. In this case, the MPSC again looks for a High-to-Low transition on the  $\overline{\text{SYNDET}}$  input and the operation repeats as explained previously. This implies the CPU should also inform the external logic that character synchronization has been lost and that the MPSC is waiting for  $\overline{\text{SYNDET}}$  to become active.

In the Monosync and Bisync Receive modes, the Sync/Hunt status bit is initially set to 1 by the Enter Hunt Mode bit. The Sync/Hunt bit is reset when the MPSC establishes character synchronization. The High-to-Low transition of the Sync/Hunt bit causes an External/Status interrupt that must be cleared by the CPU issuing the Reset External/Status Interrupt command. This enables the MPSC to detect the next transition of other External/Status bits.

When the CPU detects the end of message or that character synchronization is lost, it sets the Enter Hunt Mode control bit, which sets the Sync/Hunt bit to 1. The Low-to-High transition of the Sync/Hunt bit sets the External/Status Interrupt, which must also be cleared by the Reset External/Status Interrupt Command. Note that the  $\overline{\text{SYNDET}}$  pin acts as an output in this mode, and goes low every time a sync pattern is detected in the data stream.

In the SDLC mode, the Sync/Hunt bit is initially set by the Enter Hunt mode bit, or when the receiver is disabled. In any case, it is reset to 0 when the opening flag of the first frame is detected by the MPSC. The External/Status interrupt is also generated, and should be handled as discussed previously.

Unlike the Monosync and Bisync modes, once the Sync/Hunt bit is reset in the SDLC mode, it does not need to be set when the end of message is detected. The MPSC automatically maintains synchronization. The only way the Sync/Hunt bit can be set again is by the Enter Hunt Mode bit, or by disabling the receiver.

- D5 Clear to Send—this bit contains the inverted state of the CTS pin at the time of the last change of any of the External/Status bits (CD, CTS, Sync/Hunt, Break/Abort, or Tx Underrun/EOM). Any change of state of the CTS pin causes the CTS bit to be latched and causes an External/Status interrupt. This bit indicates the inverse of the current state of the CTS pin immediately following a Reset External/Status Interrupt command.
- D6 Transmitter Underrun/End of Message—this bit is in a set condition following a reset (internal or external). The only command that can reset this bit is the Reset Transmit Underrun/EOM Latch command (WR0, D<sub>6</sub> and D<sub>7</sub>). When the Transmit Underrun condition occurs, this bit is set, which causes the External/Status Interrupt which must be reset by issuing a Reset External/Status command (WR0; command 2).
- D7 Break/Abort—in the Asynchronous Receive mode, this bit is set when a Break sequence (null character plus framing error) is detected in the data stream. The External/Status interrupt, if enabled, is set when break is detected. The interrupt service routine must issue the Reset External/Status interrupt command (WR0, Command 2) to the break detection logic so the Break sequence termination can be recognized.

SDLC Residue Code Table (1 Field Bits in 2 Previous Bytes)

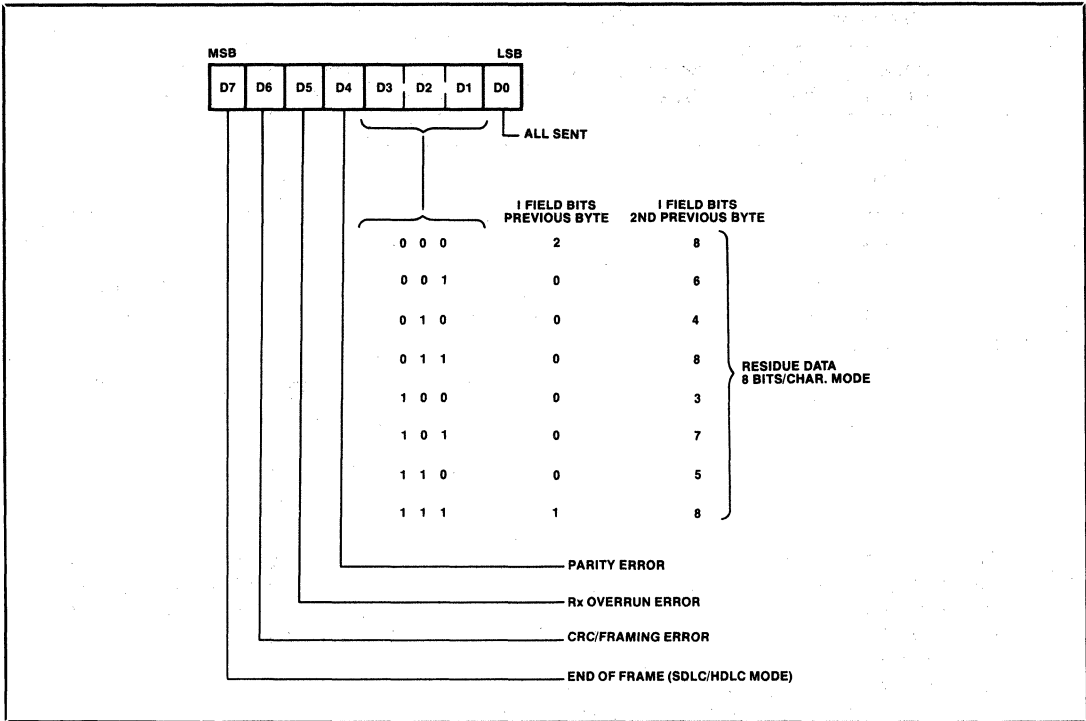
RR1 D3, D2, D1	8 bits/char		7 bits/char		6 bits/char		5 bits/char	
	Previous Byte	2nd Prev. Byte	Previous Byte	2nd Prev. Byte	Previous Byte	2nd Prev. Byte	Previous Byte	2nd Prev. Byte
1 0 0	0	3						
0 1 0	0	4			0	6		
1 1 0	0	5						
0 0 1	0	6					0	5
1 0 1	0	7						
0 1 1	0	8						
1 1 1	1	8						
0 0 0	2	8	0	7				

The Break/Abort bit is reset when the termination of the Break sequence is detected in the incoming data stream. The termination of the Break sequence also causes the External/Status interrupt to be set. The Reset External/Status Interrupt command must be issued to enable the break detection logic to look for the next Break sequence. A single extraneous null character is present in the receiver after the termination of a break; it should be read and discarded.

In the SDLC Receive mode, this status bit is set by the detection of an Abort sequence (seven or more 1's). The External/Status interrupt is handled the same way as in the case of a Break. The Break/Abort bit is not used in the Synchronous Receive mode.

- D0 All sent—this bit is set when all characters have been sent, in asynchronous modes. It is reset when characters are in the transmitter, in asynchronous modes. In synchronous modes, this bit is always set.
- D3, D2, D1 Residue Codes—bit synchronous protocols allow I-fields that are not an integral number of characters. Since transfers from the MPSC to the CPU are character oriented, the residue codes provide the capability of receiving leftover bits. Residue bits are right justified in the last two data bytes received.
- D4 Parity Error—If parity is enabled, this bit is set for received characters whose parity does not match the programmed sense (Even/Odd). This bit is latched. Once an error occurs, it remains set until the Error Reset command is written.

**Read Register 1 (RR1): (Special Receive Condition Mode)**



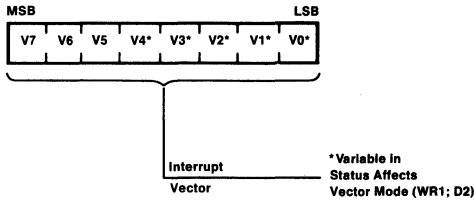
**D5** Receive Overrun Error—this bit indicates that the receive FIFO has been overloaded by the receiver. The last character in the FIFO is overwritten and flagged with this error. Once the overwritten character is read, this error condition is latched until reset by the Error Reset command. If the MPSC is in the status affects vector mode, the overrun causes a special Receive Condition Vector.

**D6** CRC/Framing Error—In async modes, a one in this bit indicates a receive fram-

ing error. In synchronous modes, a one in this bit indicates that the calculated CRC value does not match the last two bytes received. It can be reset by issuing an Error Reset command.

**D7** End of Frame—this bit is valid only in SDLC mode. A one indicates that a valid ending flag has been received. This bit is reset either by an Error Reset command or upon reception of the first character of the next frame.

**Read Register 2 (RR2):**



**RR2 Channel B**

D7-D0 Interrupt vector—contains the interrupt vector programmed into WR2. If the status affects vector mode is selected, it contains the modified vector. (See WR2) RR2 contains the modified vector for the highest priority interrupt pending. If no interrupts are pending, the variable bits in the vector are set to one.

**SYSTEM INTERFACE**

**General**

The MPSC to Microprocessor System interface can be configured in many flexible ways. The basic interface types are polled, wait, interrupt driven, or direct memory access driven.

Polled operation is accomplished by repetitively reading the status of the MPSC, and making decisions based on that status. The MPSC can be polled at any time.

Wait operation allows slightly faster data throughput for the MPSC by manipulating the Ready input to the microprocessor. Block Read or Write Operations to the MPSC are started at will by the microprocessor and the MPSC deactivates its RDY signal if it is not yet ready to transmit the new byte, or if reception of new byte is not completed.

Interrupt driven operation is accomplished via an internal or external interrupt controller. When the MPSC requires service, it sends an interrupt request signal to the microprocessor, which responds with an interrupt acknowledge signal. When the internal or external interrupt controller receives the acknowledge, it vectors the microprocessor to a service routine, in which the transaction occurs.

DMA operation is accomplished via an external DMA controller. When the MPSC needs a data transfer, it request a DMA cycle from the DMA controller. The DMA controller then takes control of the bus and simultaneously does a read from the MPSC and a write to memory or vice-versa.

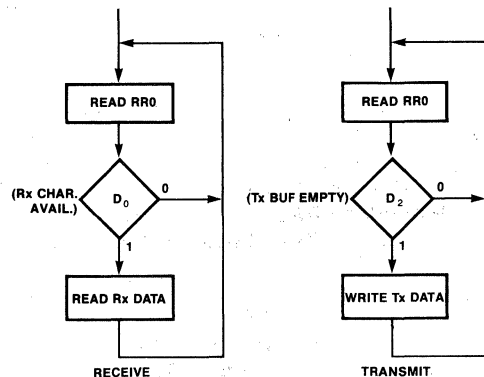
The following section describes the many configurations of these basic types of system interface techniques for both serial channels.

**Polled Operation:**

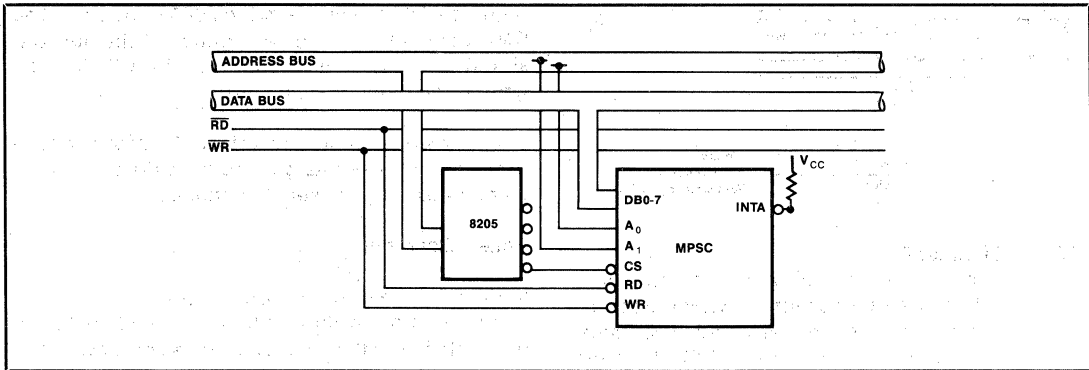
In the polled mode, the CPU must monitor the desired conditions within the MPSC by reading the appropriate bits in the read registers. All data available, status, and error conditions are represented by the appropriate bits in read registers 0 and 1 for channels A and B.

There are two ways in which the software task of monitoring the status of the MPSC has been reduced. One is the "ORing" of all conditions into the Interrupt Pending bit. (RR0; D1 channel A only). This bit is set when the MPSC requires service, allowing the CPU to monitor one bit instead of four status registers. The other is available when the "status-affects-vector" mode is selected. By reading RR2 Channel B, the CPU can read a vector who's value will indicate that one or more of group of conditions has occurred, narrowing the field of possible conditions. See WR2 and RR2 in the Detailed Command Description section.

**Software Flow, Polled Operation**



**Hardware Configuration, Polled Operation**



**WAIT OPERATION:**

Wait Operation is intended to facilitate data transmission or reception using block move operations. If a block of data is to be transmitted, for example, the CPU can execute a String I/O instruction to the MPSC. After writing the first byte, the CPU will attempt to write a second byte immediately as is the case of block move. The MPSC forces the RDY signal low which inserts wait states in the CPU's write cycle until the transmit buffer is ready to accept a new byte. At that time, the RDY signal is high allowing the CPU to finish the write cycle. The CPU then attempts the third write and the process is repeated.

Similar operation can be programmed for the receiver. During initialization, wait on transmit (WR2; D5 = 0) or wait on receive (WR1; D5 = 1) can be selected. The wait operation can be enabled/disabled by setting/resetting the Wait Enable Bit (WR1; D7).

**CAUTION:** ANY CONDITION THAT CAN CAUSE THE TRANSMITTER TO STOP (EG, CTS GOES INACTIVE) OR THE RECEIVER TO STOP (EG, RX DATA STOPS) WILL CAUSE THE MPSC TO HANG THE CPU UP IN WAIT STATES UNTIL RESET. EXTREME CARE SHOULD BE TAKEN WHEN USING THIS FEATURE.

**INTERRUPT DRIVEN OPERATION:**

The MPSC can be programmed into several interrupt modes: Non-Vectored, 8085 vectored, and 8088/86 vectored. In both vectored modes, multiple MPSC's can be daisy-chained.

In the vectored mode, the MPSC responds to an interrupt acknowledge sequence by placing a call instruction (8085 mode) and interrupt vector (8085

and 8088/86 mode) on the data bus. In the non-vectored mode, the MPSC does not respond to  $\overline{INTA}$  sequences and must rely on an external interrupt controller such as the 8259A.

The MPSC can be programmed to cause an interrupt due to up to 14 conditions in each channel. The status of these interrupt conditions is contained in Read Registers 0 and 1. These 14 conditions are all directed to cause 3 different types of internal interrupt request for each channel: receive/interrupts, transmit interrupts and external/status interrupts (if enabled).

This results in up to 6 internal interrupt request signals. The priority of those signals can be programmed to one of two fixed modes:

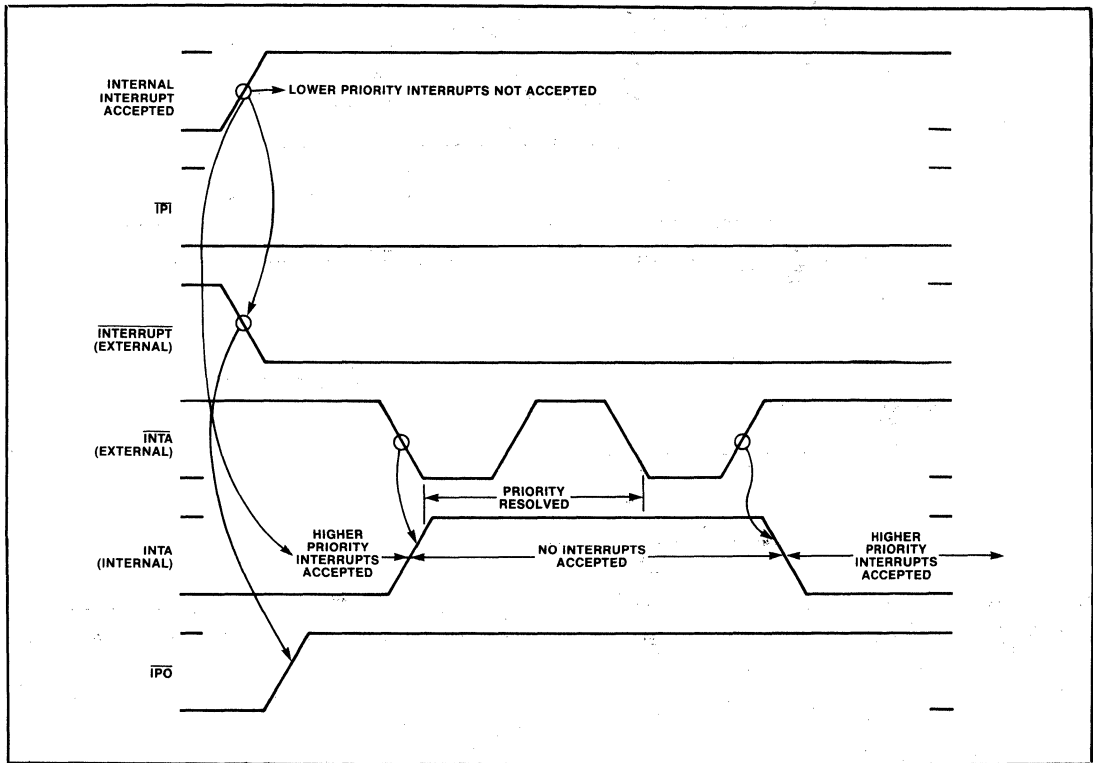
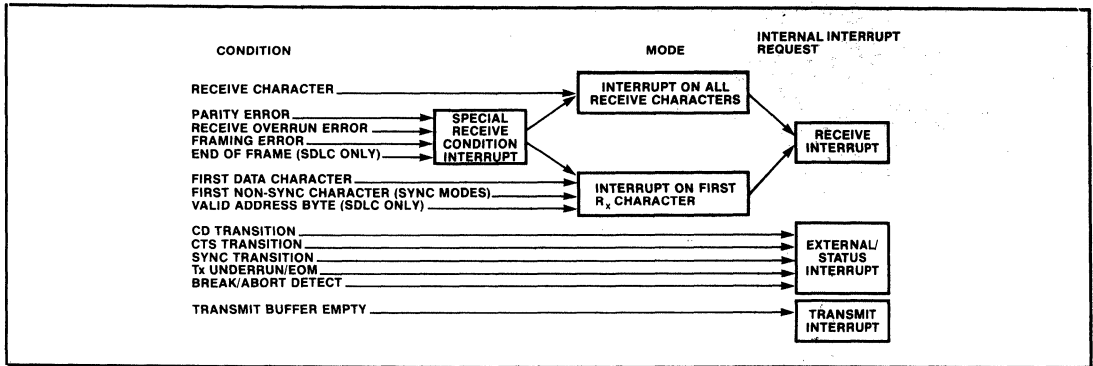
Highest Priority				Lowest Priority			
RxA	RxB	TxA	TxB	ExTA	ExTB		
RxA	TxA	RxB	TxB	ExTA	ExTB		

The interrupt priority resolution works differently for vectored and non-vectored modes.

**PRIORITY RESOLUTION: VECTORED MODE**

Any interrupt condition can be accepted internally to the MPSC at any time, unless the MPSC's internal  $\overline{INTA}$  signal is active, unless a higher priority interrupt is currently accepted, or if  $\overline{IPI}$  is inactive (high). The MPSC's internal  $\overline{INTA}$  is set on the leading (falling) edge of the first External  $\overline{INTA}$  pulse and reset on the trailing (rising) edge of the second External  $\overline{INTA}$  pulse. After an interrupt is accepted internally, an External  $\overline{INT}$  request is generated and the  $\overline{IPO}$  goes inactive.  $\overline{IPO}$  and  $\overline{IPI}$  are used for daisy-chaining MPSC's together.

Interrupt Condition Grouping

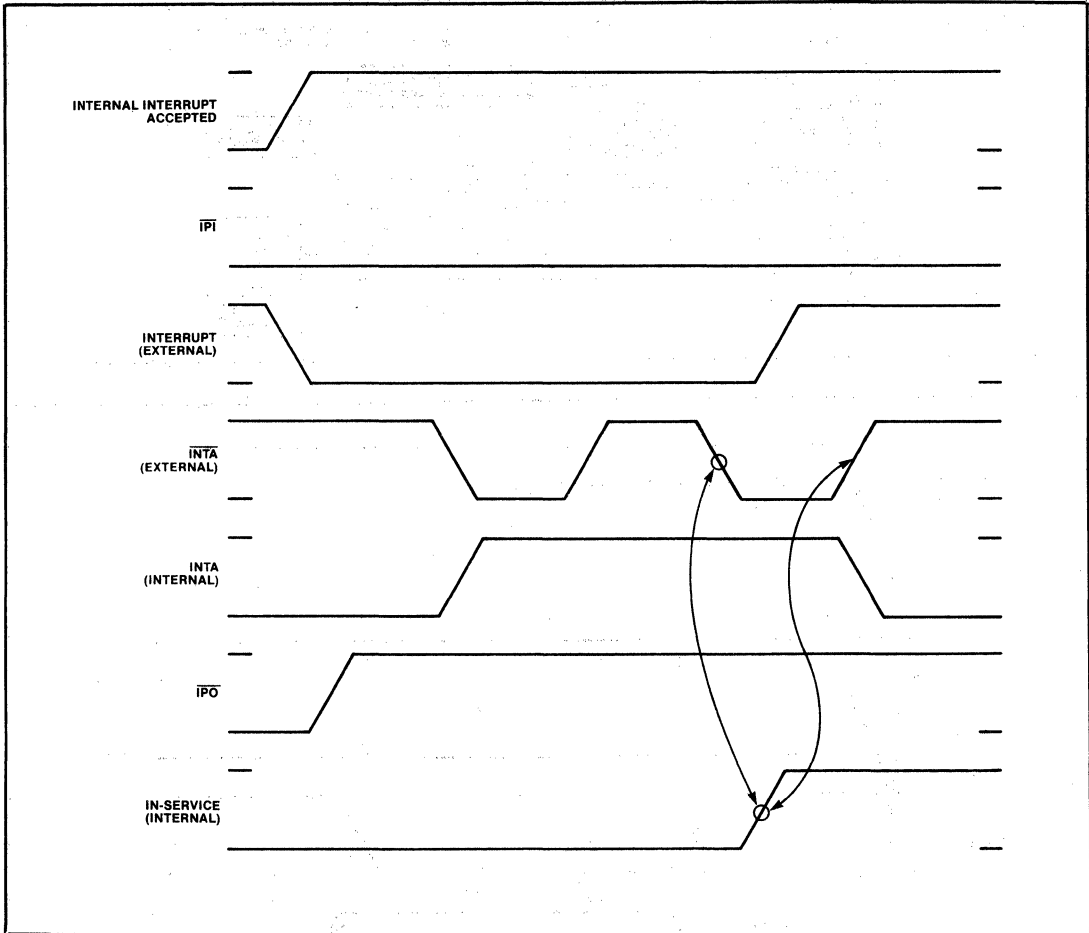


The MPSC's internal INTA is set on the leading (falling) edge of the first external  $\overline{INTA}$  pulse, and reset on the trailing (rising) edge of the second external  $\overline{INTA}$  pulse. After an interrupt is accepted internally,

an external  $\overline{INT}$  request is generated and  $\overline{IPO}$  goes inactive (high).  $\overline{IPO}$  and  $\overline{IPI}$  are used for daisy-chaining MPSC's together.



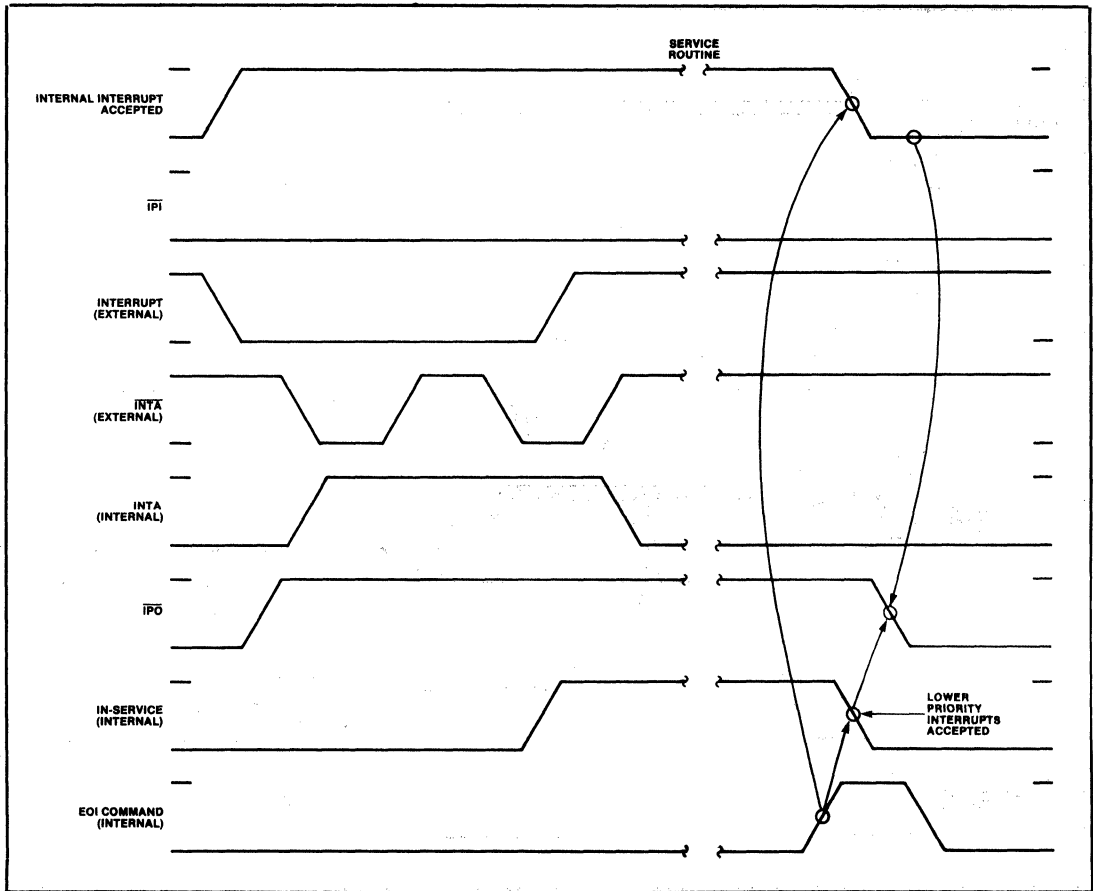
In-Service Timing



Each of the six interrupt sources has an associated In-Service latch. After priority has been resolved, the

highest priority In-Service latch is set. After the In-Service latch is set, the  $\overline{INT}$  pin goes inactive (high).

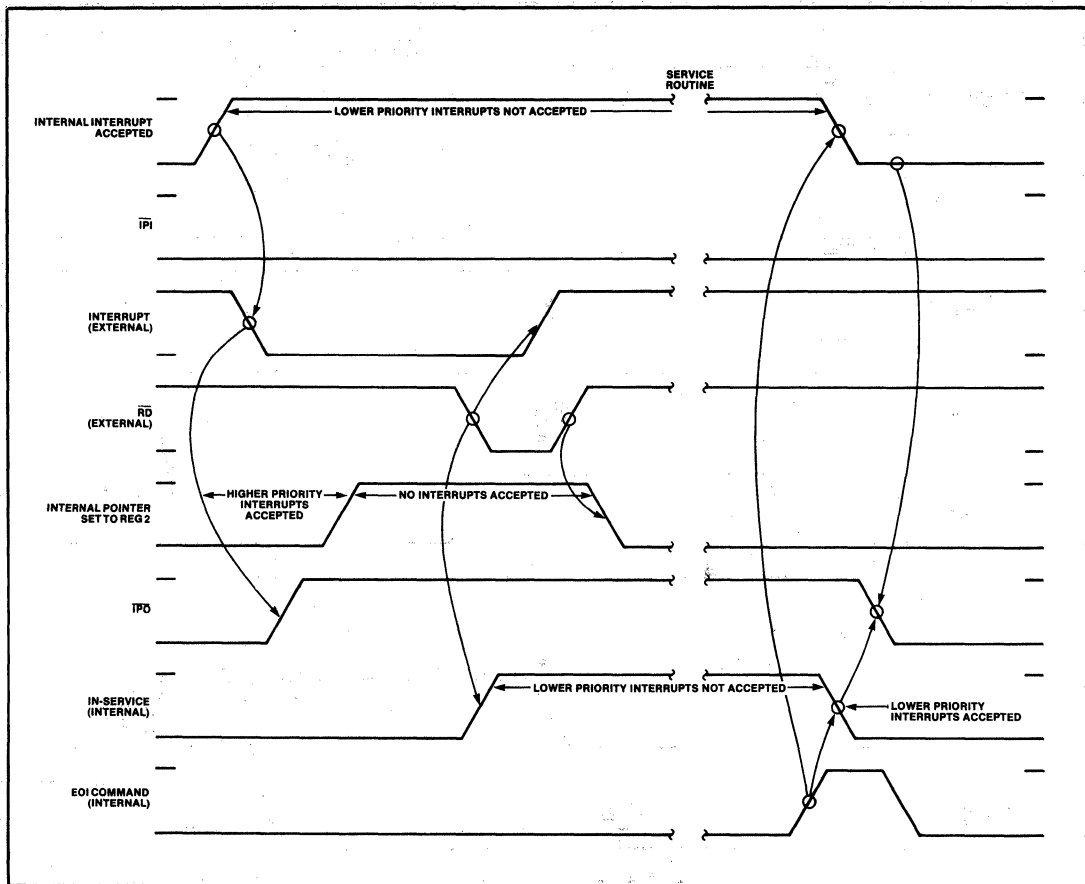
EOI Command Timing



Lower priority interrupts are not accepted internally while the In-Service latch is set. However, higher priority interrupts are accepted internally and a new external  $\overline{INT}$  request is generated. If the CPU responds with a new INTA sequence, the MPSC will respond as before, suspending the lower priority interrupt.

After the interrupt is serviced, the End-of-Interrupt (EOI) command should be written to the MPSC. This command will cause an internal pulse that is used to reset the In-Service Latch which allows service for lower priority interrupts in the daisy-chain to resume, provided a new INTA sequence does not start for a higher priority interrupt (higher than the highest under service). If there is no interrupt pending internally, the IPO follows IPI.

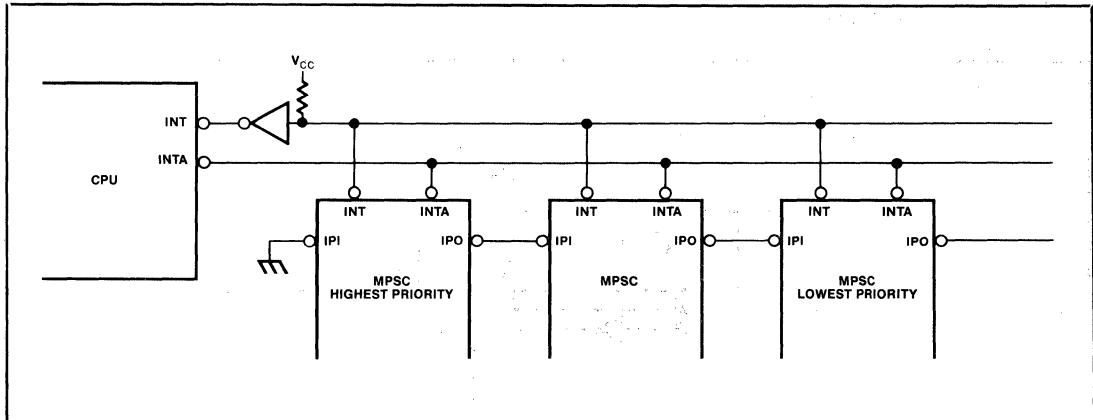
**Non-Vectored Interrupt Timing**



**PRIORITY RESOLUTION:  
NON-VECTORED MODE**

In non-vectored mode, the MPSC does not respond to interrupt acknowledge sequences. The MPSC should be programmed to the Status-Affects-Vector mode, and the CPU should read RR2 (Ch. B) in its service routine to determine which interrupt requires service.

In this case, the internal pointer being set to RR2 provides the same function as the internal INTA signal in the vectored mode. It inhibits acceptance of any additional internal interrupts and its leading edge starts the interrupt priority resolution circuit. The interrupt priority resolution is ended by the leading edge of the read signal used by the CPU to retrieve the modified vector. The leading edge of read sets the In-Service latch and forces the external INT output inactive (high). The internal pointer is reset to zero after the trailing edge of the read pulse.



Note that if RR2 is specified but not read, no internal interrupts, regardless of priority, are accepted.

#### DAISY CHAINING MPSC:

In the vectored interrupt mode, multiple MPSC's can be daisy-chained on the same  $\overline{\text{INT}}$ ,  $\overline{\text{INTA}}$  signals. These signals, in conjunction with the  $\overline{\text{IPI}}$  and  $\overline{\text{IPO}}$  allow a daisy-chain-like interrupt resolution scheme. This scheme can be configured for either 8085 or 8086/88 based system.

In either mode, the same hardware configuration is called for. The  $\overline{\text{INT}}$  request lines are wire-OR'ed together at the input of a TTL inverter which drives the INT pin of the CPU. The  $\overline{\text{INTA}}$  signal from the CPU drives all of the daisy-chained MPSC's.

The MPSC drives  $\overline{\text{IPO}}$  (Interrupt Priority Output) inactive (high) if  $\overline{\text{IPI}}$  (Interrupt Priority Input) is inactive (high), or if the MPSC has an interrupt pending.

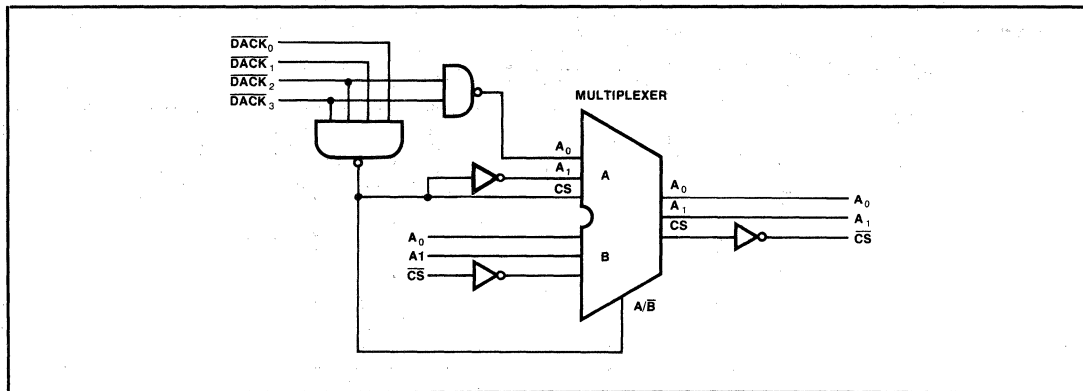
The  $\overline{\text{IPO}}$  of the highest priority MPSC is connected to the  $\overline{\text{IPI}}$  of the next highest priority MPSC, and so on.

If  $\overline{\text{IPI}}$  is active (low), the MPSC knows that all higher priority MPSC's have no interrupts pending. The  $\overline{\text{IPI}}$  pin of the highest priority MPSC is strapped active (low) to ensure that it always has priority over the rest.

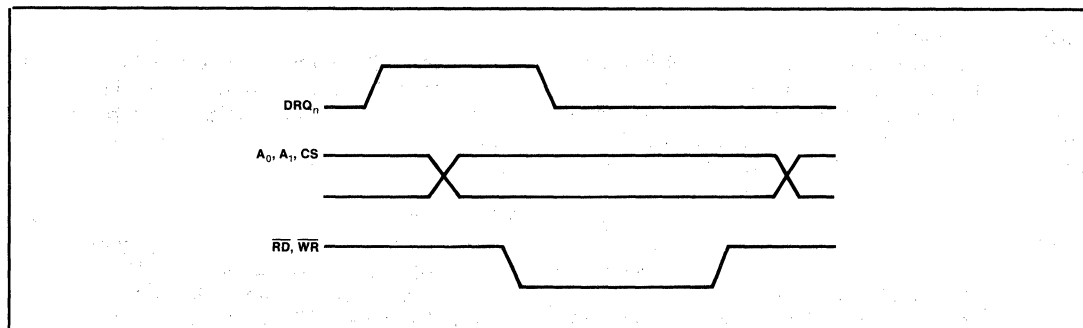
MPSC's Daisy-chained on an 8088/86 CPU should be programmed to the 8088/86 Interrupt mode (WR2; D4, D3 (Ch. A). MPSC's Daisy-chained on an 8085 CPU should be programmed to 8085 interrupt mode 1 if it is the highest priority MPSC. In this mode, the highest priority MPSC issues the CALL instruction during the first  $\overline{\text{INTA}}$  cycle, and the interrupting MPSC provides the interrupt vector during the following  $\overline{\text{INTA}}$  cycles. Lower priority MPSC's should be programmed to 8085 interrupt mode 2.

MPSC's used alone in 8085 systems should be programmed to 8085 mode 1 interrupt operation.

**DMA Acknowledge Circuit**



**DMA Timing**



**DMA OPERATION**

Each MPSC can be programmed to utilize up to four DMA channels: Transmit Channel A, Receive Channel A, Transmit Channel B, Receive Channel B. Each DMA Channel has an associated DMA Request line. Acknowledgement of a DMA cycle is done via normal data read or write cycles. This is accomplished by encoding the  $\overline{DACK}$  signals to generate  $A_0$ ,  $A_1$ , and  $\overline{CS}$  signals, and multiplexing them with the normal  $A_0$ ,  $A_1$ , and  $\overline{CS}$  signals.

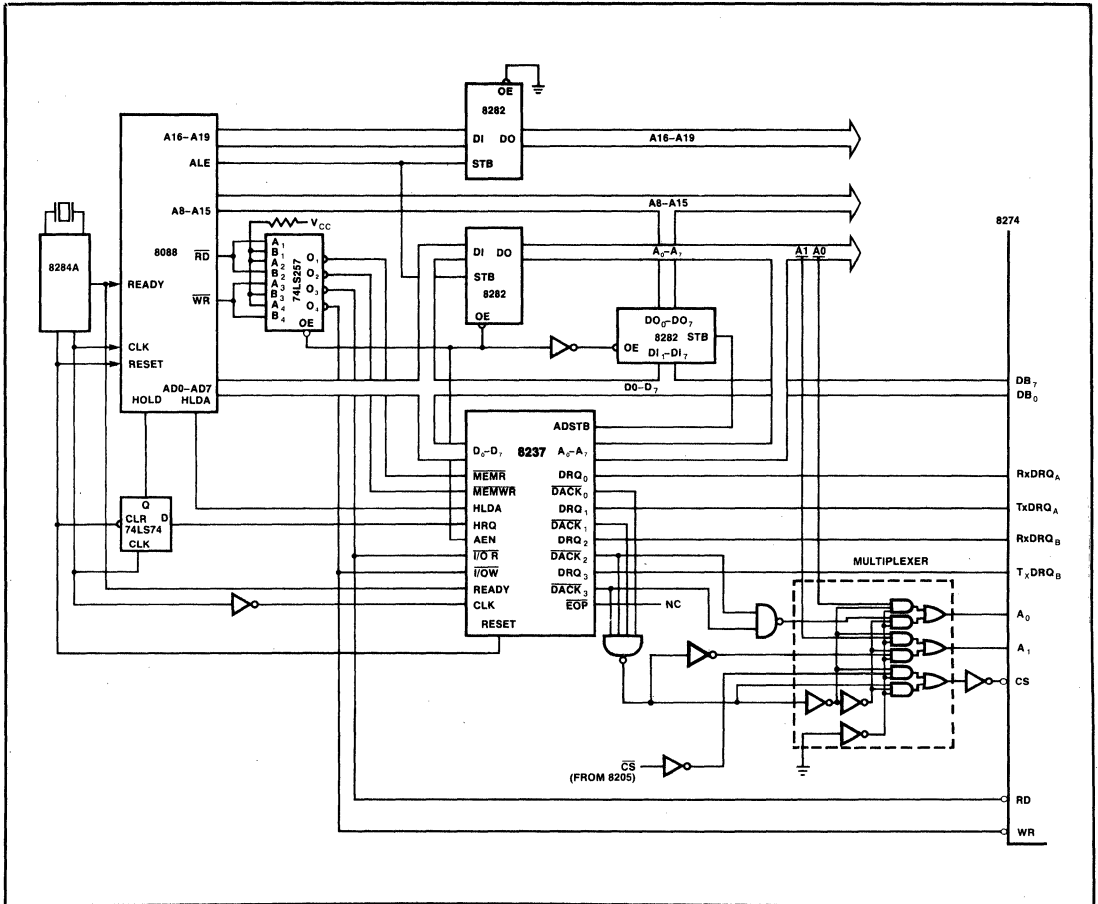
**PERMUTATIONS**

Channels A and B can be used with different system interface modes. In all cases it is impossible to poll the MPSC. The following table shows the possible

permutations of interrupt, wait, and DAM modes for channels A and B. Bits  $D_1$ ,  $D_0$  of WR2 Ch. A determine these permutations.

Permutation WR2 Ch. A $D_1 D_0$	Channel A	Channel B
0 0	Wait Interrupt Polled	Wait Interrupt Polled
0 1	DMA Polled	Interrupt Polled
1 0	DMA Polled	DMA Polled

$D_1, D_0 = 1, 1$  is illegal.



**ABSOLUTE MAXIMUM RATINGS\***

Ambient Temperature  
 Under Bias ..... 0°C to +70°C  
 Storage Temperature  
 (Ceramic Package) ..... -65°C to +150°C  
 (Plastic Package) ..... -40°C to +125°C  
 Voltage On Any Pin With  
 Respect to Ground ..... -0.5V to +7.0V  
 Power Dissipation ..... 1.5W

*\*NOTICE: Stresses above those listed under "Absolute Maximum Ratings" may cause permanent damage to the device. This is a stress rating only and functional operation of the device at these or any other conditions above those indicated in the operational sections of this specification is not implied. Exposure to absolute maximum rating conditions for extended periods may affect device reliability.*

**D.C. CHARACTERISTICS** ( $T_A = 0^\circ\text{C}$  to  $70^\circ\text{C}$ ;  $V_{CC} = +5\text{V} \pm 10\%$ )

Symbol	Parameter	Min.	Max.	Units	Test Conditions
$V_{IL}$	Input Low Voltage	-0.5	+0.8	V	
$V_{IH}$	Input High Voltage	+2.0	$V_{CC} + 0.5$	V	
$V_{OL}$	Output Low Voltage		+0.45	V	$I_{OL} = 2.0\text{mA}$
$V_{OH}$	Output High Voltage	+2.4		V	$I_{OH} = -200\mu\text{A}$
$I_{IL}$	Input Leakage Current		+10	$\mu\text{A}$	$V_{IN} = V_{CC}$ to 0V
$I_{OL}$	Output Leakage Current		+10	$\mu\text{A}$	$V_{OUT} = V_{CC}$ to 0V
$I_{CC}$	$V_{CC}$ Supply Current		180	mA	

**CAPACITANCE** ( $T_A = 25^\circ\text{C}$ ;  $V_{CC} = \text{GND} = 0\text{V}$ )

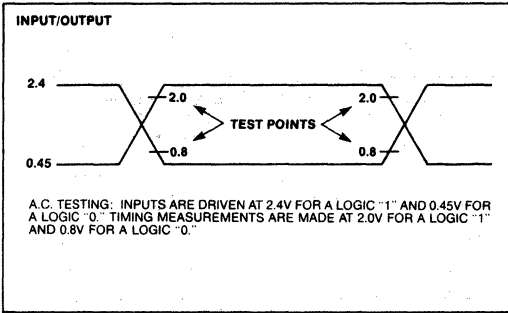
Symbol	Parameter	Min.	Max.	Units	Test Conditions
$C_{IN}$	Input Capacitance		10	pF	$f_c = 1\text{MHz}$ ;
$C_{OUT}$	Output Capacitance		15	pF	Unmeasured
$C_{I/O}$	Input/Output Capacitance		20	pF	pins returned to GND

**A.C. CHARACTERISTICS** ( $T_A = 0^\circ\text{C}$  to  $70^\circ\text{C}$ ;  $V_{CC} = +5\text{V} \pm 10\%$ )

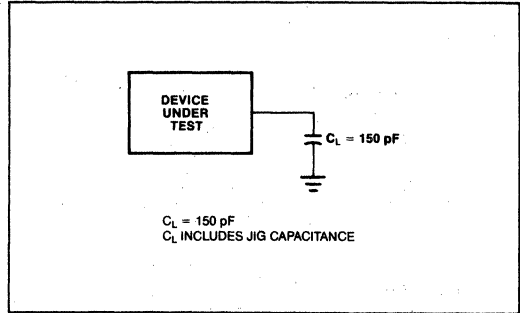
Symbol	Parameter	Min.	Max.	Units	Test Conditions
$t_{CY}$	CLK Period	250	4000	ns	
$t_{CL}$	CLK Low Time	105	2000	ns	
$t_{CH}$	CLK High Time	105	2000	ns	
$t_r$	CLK Rise Time	0	30	ns	
$t_f$	CLK Fall Time	0	30	ns	
$t_{AR}$	A0, A1 Setup to $\overline{RD}$ ↓	0		ns	
$t_{AD}$	A0, A1 to Data Output Delay		200	ns	$C_L = 150$ pf
$t_{RA}$	A0, A1 Hold After $\overline{RD}$ ↑	0		ns	
$t_{RD}$	$\overline{RD}$ ↓ to Data Output Delay		200	ns	$C_L = 150$ pf
$t_{RR}$	$\overline{RD}$ Pulse Width	250		ns	
$t_{DF}$	Output Float Delay		120	ns	
$t_{AW}$	$\overline{CS}$ , A0, A1 Setup to $\overline{WR}$ ↓	0		ns	
$t_{WA}$	$\overline{CS}$ , A0, A1 Hold after $\overline{WR}$ ↑	0		ns	
$t_{WW}$	$\overline{WR}$ Pulse Width	250		ns	
$t_{DW}$	Data Setup to $\overline{WR}$ ↑		150	ns	
$t_{WD}$	Data Hold After $\overline{WR}$ ↑	0		ns	
$t_{PI}$	$\overline{PI}$ Setup to $\overline{INTA}$ ↓	0		ns	
$t_{IP}$	$\overline{PI}$ Hold after $\overline{INTA}$ ↑	0		ns	
$t_{II}$	$\overline{INTA}$ Pulse Width	250		ns	
$t_{IAPO}$	$\overline{INTA}$ ↓ to $\overline{IPO}$ Delay		200	ns	
$t_{PIPO}$	$\overline{PI}$ ↓ to $\overline{IPO}$ Delay		100	ns	
$t_{ID}$	$\overline{INTA}$ ↓ to Data Output Delay		200	ns	
$t_{CQ}$	$\overline{RD}$ or $\overline{WR}$ to $\overline{DRQ}$ ↓		150	ns	
$t_{RV}$	Recovery Time Between Controls	300		ns	
$t_{CW}$	$\overline{CS}$ , A0, A1 to $\overline{RDY}_A$ or $\overline{RDY}_B$ Delay		120	ns	
$t_{DCY}$	Data Clock Cycle	400		ns	
$t_{DCL}$	Data Clock Low Time	180		ns	
$t_{DCH}$	Data Clock High Time	180		ns	
$t_{TD}$	$\overline{TxC}$ to $\overline{TxD}$ Delay		300	ns	
$t_{DS}$	RxD Setup to $\overline{RxC}$ ↑	0		ns	
$t_{DH}$	RxD Hold after $\overline{RxC}$ ↑	140		ns	
$t_{ITD}$	$\overline{TxC}$ to $\overline{INT}$ Delay	4	6	tcy	
$t_{IRD}$	RxC to $\overline{INT}$ Delay	7	10	tcy	
$t_{PL}$	$\overline{CTS}$ , $\overline{CD}$ , SYND $\overline{ET}$ Low Time	200		ns	
$t_{PH}$	$\overline{CTS}$ , $\overline{CD}$ , SYND $\overline{ET}$ High Time	200		ns	
$t_{IPD}$	External $\overline{INT}$ from $\overline{CTS}$ , $\overline{CD}$ , SYND $\overline{ET}$		500	ns	



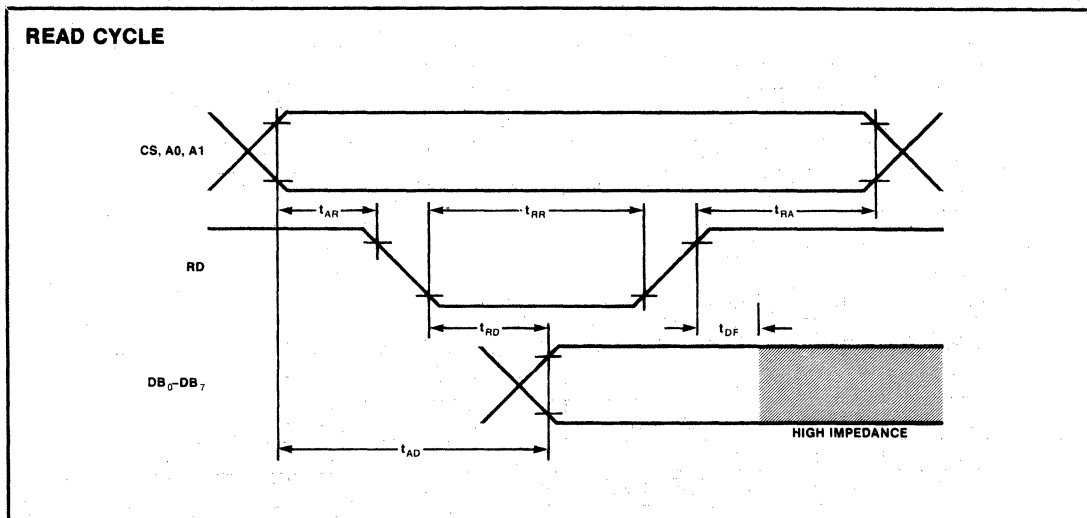
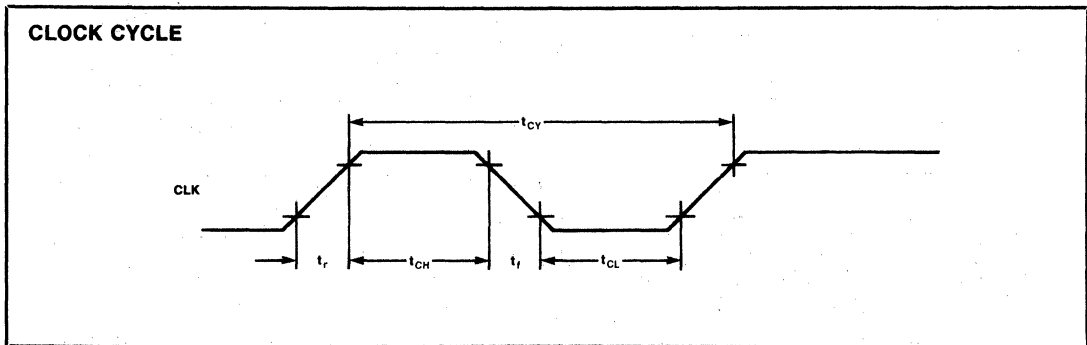
A.C. TESTING INPUT, OUTPUT WAVEFORM



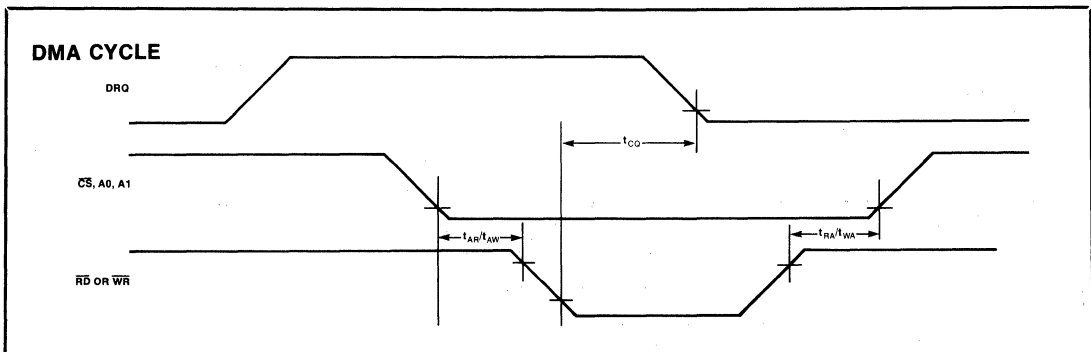
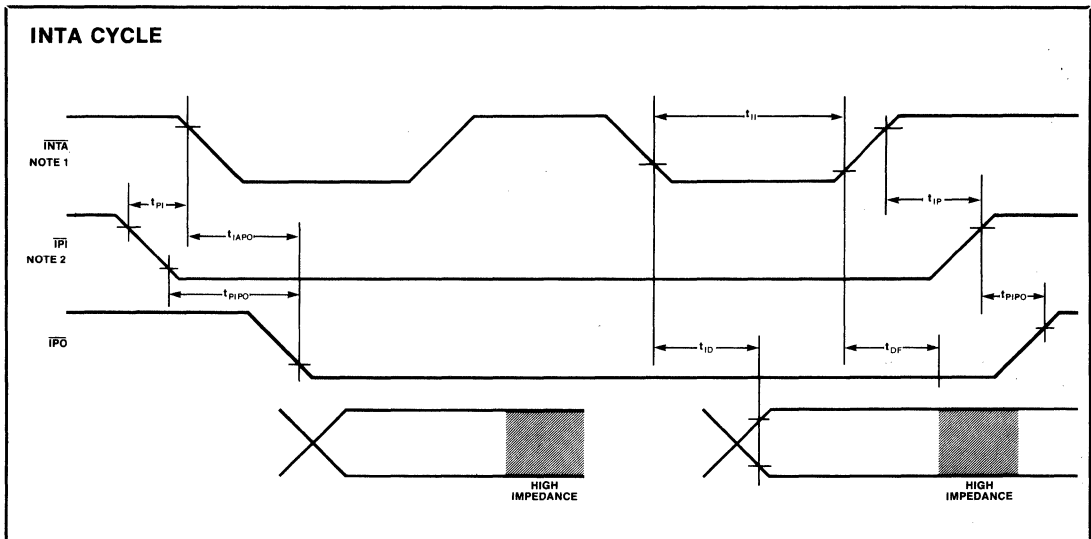
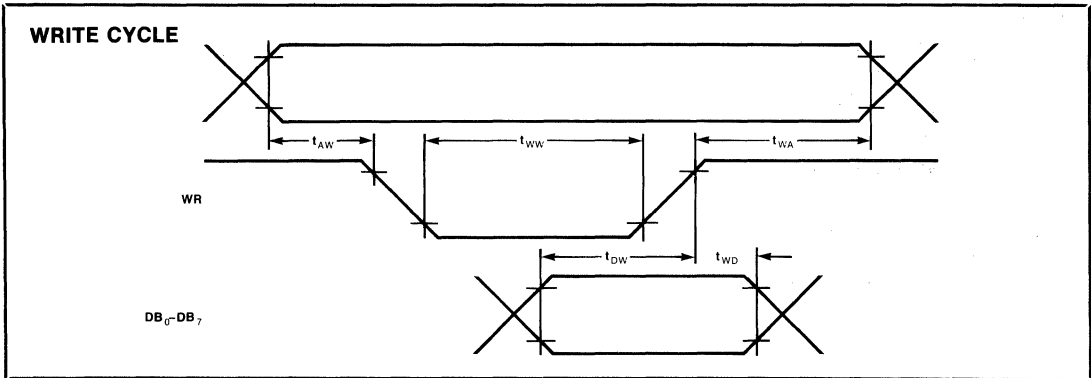
A.C. TESTING LOAD CIRCUIT



WAVEFORMS



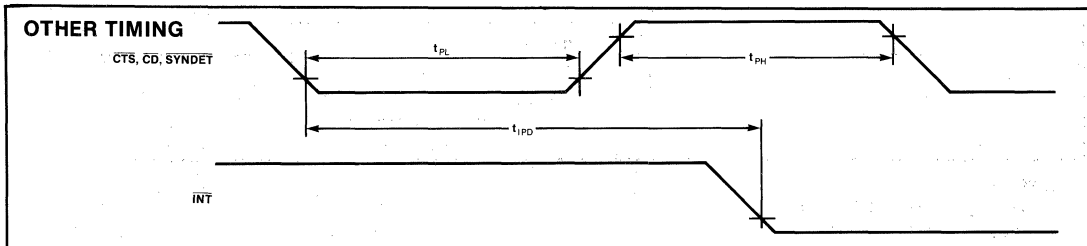
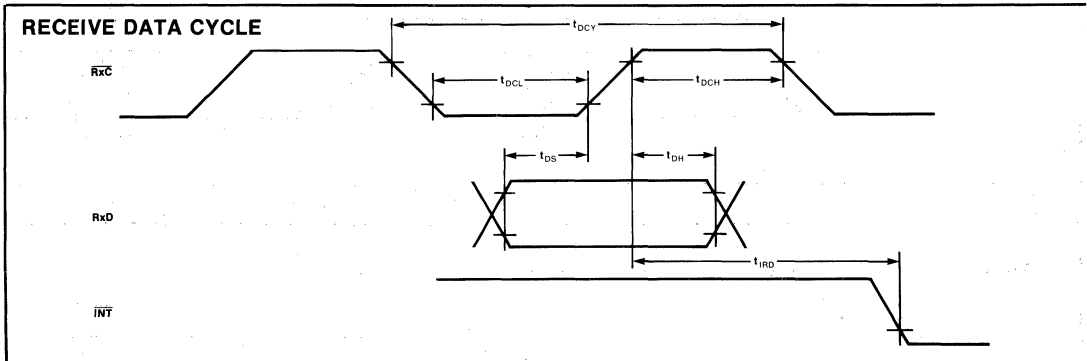
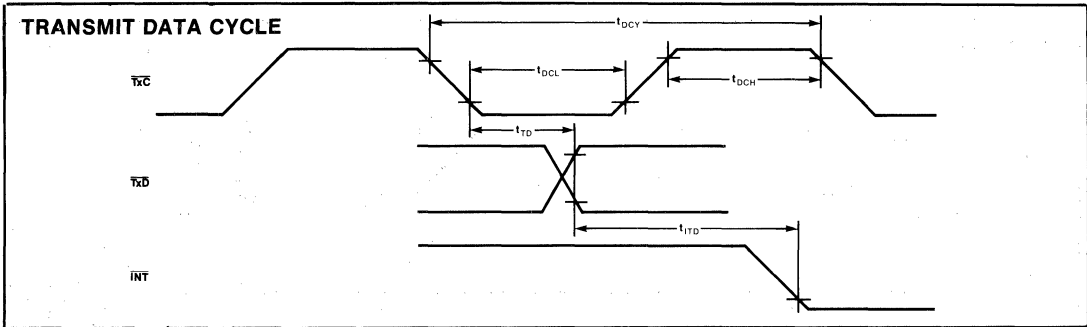
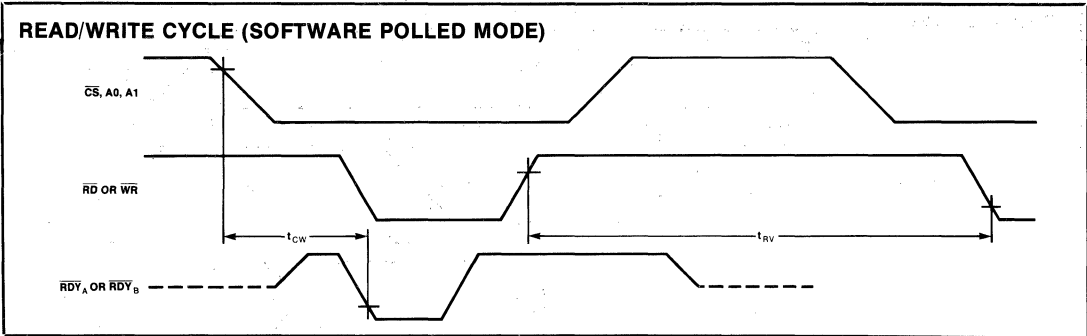
WAVEFORMS (Continued)



NOTES:

1. INTA signal acts as RD signal.
2. IPI signal acts as CS signal.

WAVEFORMS (Continued)



## 8291A GPIB TALKER/LISTENER

- Designed to Interface Microprocessors (e.g., 8048/49, 8051, 8080/85, 8086/88) to an IEEE Standard 488 Digital Interface Bus
- Programmable Data Transfer Rate
- Complete Source and Acceptor Handshake
- Complete Talker and Listener Functions with Extended Addressing
- Service Request, Parallel Poll, Device Clear, Device Trigger, Remote/Local Functions
- Selectable Interrupts
- On-Chip Primary and Secondary Address Recognition
- Automatic Handling of Addressing and Handshake Protocol
- Provision for Software Implementation of Additional Features
- 1–8 MHz Clock Range
- 16 Registers (8 Read, 8 Write), 2 for Data Transfer, the Rest for Interface Function Control, Status, etc.
- Directly Interfaces to External Non-Inverting Transceivers for Connection to the GPIB
- Provides Three Addressing Modes, Allowing the Chip to be Addressed Either as a Major or a Minor Talker/Listener with Primary or Secondary Addressing
- DMA Handshake Provision Allows for Bus Transfers without CPU Intervention
- Trigger Output Pin
- On-Chip EOS (End of Sequence) Message Recognition Facilitates Handling of Multi-Byte Transfers

The 8291A is an enhanced version of the 8291 GPIB Talker/Listener designed to interface microprocessors to an IEEE Standard 488 Instrumentation Interface Bus. It implements all of the Standard's interface functions except for the controller. The controller function can be added with the 8292 GPIB Controller, and the 8293 GPIB Transceiver performs the electrical interface for Talker/Listener and Talker/Listener/Controller configurations.

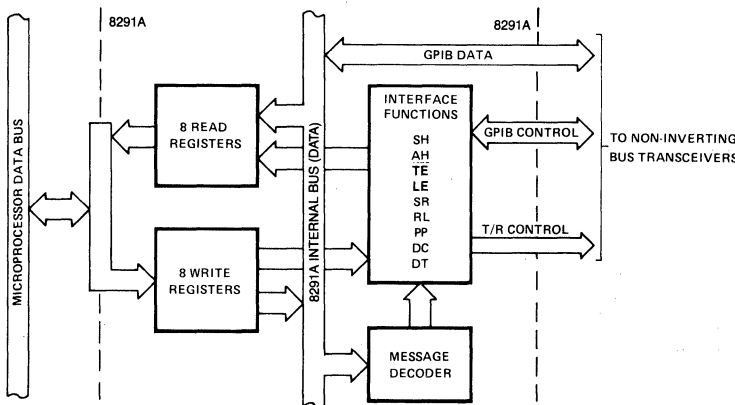


Figure 1. Block Diagram

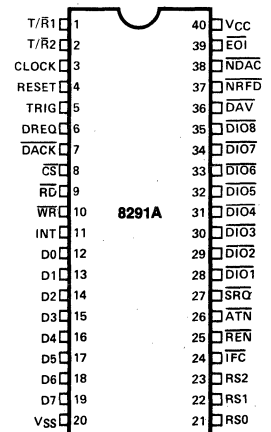


Figure 2. Pin Configuration

## 8291A FEATURES AND IMPROVEMENTS

The 8291A is an improved design of the 8291 GPIB Talker/Listener. Most of the functions are identical to the 8291, and the pin configuration is unchanged.

The 8291A offers the following improvements to the 8291:

1.  $\overline{EOI}$  is active with the data as a ninth data bit rather than as a control bit. This is to comply with some additions to the 1975 IEEE-488 Standard incorporated in the 1978 Standard.
2. The BO interrupt is not asserted until RFD is true. If the Controller asserts  $\overline{ATN}$  synchronously, the data is guaranteed to be transmitted. If the Controller asserts  $\overline{ATN}$  asynchronously, the SH (Source Handshake) will return to SIDS (Source Idle State), and the output data will be cleared. The, if  $\overline{ATN}$  is released while the 8291A is addressed to talk, a new BO interrupt will be generated. This change fixes 8291 problems which caused data to be lost or repeated and a problem with the RQS bit (sometimes cannot be asserted while talking).
3. LLOC and REMC interrupts are setting flipflops rather than toggling flipflops in the interrupt backup register. This ensures that the CPU knows that these state changes have occurred. The actual state can be determined by checking the LLO and REM status bits in the upper nibble of the Interrupt Status 2 Register.
4. DREQ is cleared by  $\overline{DACK}$  ( $\overline{RD} + \overline{WR}$ ). DREQ on the 8291 was cleared only by  $\overline{DACK}$  which is not compatible with the 8089 I/O Processor.
5. The INT bit in Interrupt Status 2 Register is duplicated in bit 7 of the Address 0 Register. If software polling is used to check for an interrupt, INT in the Address 0 Register should be polled rather than the Interrupt Status 2 Register. This ensures that no interrupts are lost due to asynchronous status reads and interrupts.
6. The 8291A's Send  $\overline{EOI}$  Auxiliary Command works on any byte including the first byte of a message. The 8291 did not assert  $\overline{EOI}$  after this command for a one byte message nor on two consecutive bytes.
7. To avoid confusion between holdoff on DAV versus RFD if a device is readdressed from a talker to a listener role or vice-versa during a holdoff, the "Holdoff on Source Handshake" has been eliminated. Only "Holdoff on Acceptor Handshake" is available.
8. The rsv local message is cleared automatically upon exit from SPAS if (APRS.STRS.SPAS) occurred. The automatic resetting of the bit after the serial poll is complete simplifies the service request software.
9. The SPASC interrupt on the 8291 has been replaced by the SPC (Serial Poll Complete) interrupt on the 8291A. SPC interrupt is set on exit from SPAS if APRS.STRS.SPAS occurred, indicating that the controller has read the bus status byte after the 8291A requested service. The SPASC interrupt was ambiguous because a controller could enter SPAS and exit SPAS generating two SPASC interrupts without reading the serial poll status byte. The SPC interrupt also simplifies the CPU's software by eliminating the interrupt when the serial poll is half way done.
10. The rti Auxiliary Command in the 8291 has been replaced by Set and Clear rti Commands in the 8291A. Using the new commands, the CPU has the flexibility to extend the length of local mode or leave it as a short pulse as in the 8291.
11. A holdoff RFD on GET, SDC, and DCL feature has been added to prevent additional bus activity while the CPU is responding to any of these commands. The feature is enabled by a new bit ( $B_4$ ) in the Auxiliary Register B.
12. On the 8291, BO could cease to occur upon  $\overline{IFC}$  going false if  $\overline{IFC}$  occurred asynchronously. On the 8291A, BO continues to occur after  $\overline{IFC}$  has gone false even if it arrived asynchronously.
13. User's software can distinguish between the 8291 and the 8291A as follows:
  - a) pon (00H to register 5)
  - b) RESET (02H to register 5)
  - c) Read Interrupt Status 1 Register. If BO interrupt is set, the device is the 8291. If BO is clear, it is the 8291A.

This can be used to set a flag in the user's software which will permit special routines to be executed for each device. It could be included as part of a normal initialization procedure as the first step after a chip reset.

Table 1. Pin Description

Symbol	Pin No.	Type	Name and Function
D <sub>0</sub> -D <sub>7</sub>	12-19	I/O	<b>Data Bus Port:</b> To be connected to microprocessor data bus.
RS <sub>0</sub> -RS <sub>2</sub>	21-23	I	<b>Register Select:</b> Inputs, to be connected to three non-multiplexed microprocessor address bus lines. Select which of the 8 internal read (write) registers will be read from (written into) with the execution of RD (WR.)
$\overline{CS}$	8	I	<b>Chip Select:</b> When low, enables reading from or writing into the register selected by RS <sub>0</sub> -RS <sub>2</sub> .
$\overline{RD}$	9	I	<b>Read Strobe:</b> When low with $\overline{CS}$ or $\overline{DACK}$ low, selected register contents are read.
$\overline{WR}$	10	I	<b>Write Strobe:</b> When low with $\overline{CS}$ or $\overline{DACK}$ low, data is written into the selected register.
INT (INT)	11	O	<b>Interrupt Request:</b> To the microprocessor, set high for request and cleared when the appropriate register is accessed by the CPU. May be software configured to be active low.
DREQ	6	O	<b>DMA Request:</b> Normally low, set high to indicate byte output or byte input in DMA mode; reset by $\overline{DACK}$ .
$\overline{DACK}$	7	I	<b>DMA Acknowledge:</b> When low, resets DREQ and selects data in/data out register for DMA data transfer (actual transfer done by RD/WR pulse).  Must be high if DMA is not used.
TRIG	5	O	<b>Trigger Output:</b> Normally low; generates a triggering pulse with 1 $\mu$ sec min. width in response to the GET bus command or Trigger auxiliary command.
CLOCK	3	I	<b>External Clock:</b> Input, used only for T <sub>1</sub> delay generator. May be any speed in 1-8 MHz range.

Symbol	Pin No.	Type	Name and Function
RESET	4	I	<b>Reset Input:</b> When high, forces the device into an "idle" (initialization) mode. The device will remain at "idle" until released by the microprocessor, with the "Immediate Execute pon" local message.
$\overline{DIO}_1$ - $\overline{DIO}_8$	28-35	I/O	<b>8-Bit GPIB Data Port:</b> Used for bidirectional data byte transfer between 8291A and GPIB via non-inverting external line transceivers.
$\overline{DAV}$	36	I/O	<b>Data Valid:</b> GPIB handshake control line. Indicates the availability and validity of information on the $\overline{DIO}_1$ - $\overline{DIO}_8$ and $\overline{EOI}$ lines.
$\overline{NRFD}$	37	I/O	<b>Not Ready for Data:</b> GPIB handshake control line. Indicates the condition of readiness of device(s) connected to the bus to accept data.
$\overline{NDAC}$	38	I/O	<b>Not Data Accepted:</b> GPIB handshake control line. Indicates the condition of acceptance of data by the device(s) connected to the bus.
$\overline{ATN}$	26	I	<b>Attention:</b> GPIB command line. Specifies how data on $\overline{DIO}$ lines are to be interpreted.
$\overline{IFC}$	24	I	<b>Interface Clear:</b> GPIB command line. Places the interface functions in a known quiescent state.
$\overline{SRQ}$	27	O	<b>Service Request:</b> GPIB command line. Indicates the need for attention and requests an interruption of the current sequence of events on the GPIB.
$\overline{REN}$	25	I	<b>Remote Enable:</b> GPIB command line. Selects (in conjunction with other messages) remote or local control of the device.
$\overline{EOI}$	39	I/O	<b>End or Identify:</b> GPIB command line. Indicates the end of a multiple byte transfer sequence or, in conjunction with $\overline{ATN}$ , addresses the device during a polling sequence.

Table 1. Pin Description (Continued)

Symbol	Pin No.	Type	Name and Function
T/R1	1	O	<b>External Transceivers Control Line:</b> Set high to indicate output data/signals on the $\overline{DIO}_1$ - $\overline{DIO}_8$ and $\overline{DAV}$ lines and input signals on the NRFD and NDAC lines (active source handshake). Set low to indicate input data/signals on the $\overline{DIO}_1$ - $\overline{DIO}_8$ and $\overline{DAV}$ lines and output signals on the NRFD and NDAC lines (active acceptor handshake).

Symbol	Pin No.	Type	Name and Function
T/R2	2	O	<b>External Transceivers Control Line:</b> Set to indicate output signals on the $\overline{EOI}$ line. Set low to indicate expected input signal on the $\overline{EOI}$ line during parallel poll.
V <sub>cc</sub>	40	P.S.	<b>Positive Power Supply:</b> (5V ±10%).
GND	20	P.S.	<b>Circuit Ground Potential.</b>

**NOTE:**

All signals on the 8291A pins are specified with positive logic. However, IEEE 488 specifies negative logic on its 16 signal lines. Thus, the data is inverted once from  $D_0$ - $D_7$  to  $\overline{DIO}_0$ - $\overline{DIO}_8$  and non-inverting bus transceivers should be used.

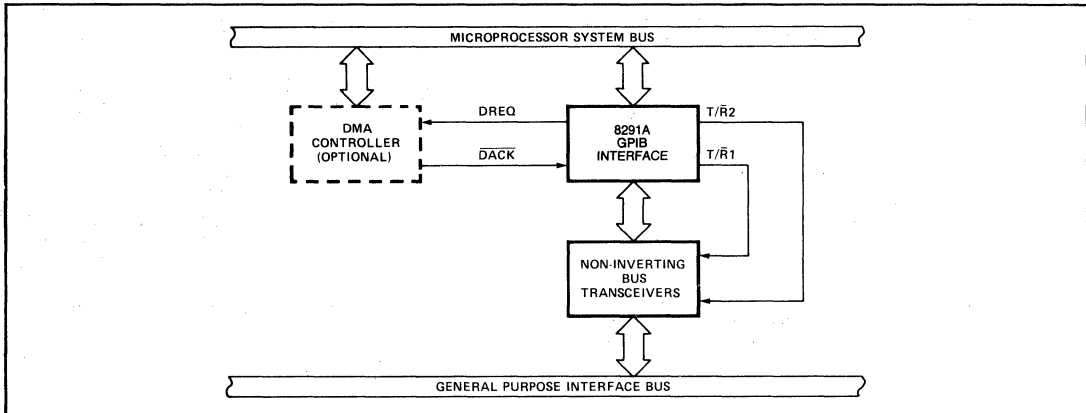


Figure 3. 8291A System Diagram

**THE GENERAL PURPOSE INTERFACE BUS (GPIB)**

The General Purpose Interface Bus (GPIB) is defined in the IEEE Standard 488-1978 "Digital Interface for Programmable Instrumentation." Although a knowledge of this standard is assumed, Figure 4 provides the bus structure for quick reference. Also, Tables 2 and 3 reference the interface state mnemonics and the interface messages respectively. Modified state diagrams for the 8291A are presented in Appendix A.

**General Description**

The 8291A is a microprocessor-controlled device designed to interface microprocessors, e.g., 8048/49, 8051, 8080/85, 8086/88 to the GPIB. It implements all of the interface functions defined in the

IEEE-488 Standard except for the controller function. If an implementation of the Standard's Controller is desired, it can be connected with an Intel® 8292 to form a complete interface.

The 8291A handles communication between a microprocessor-controlled device and the GPIB. Its capabilities include data transfer, handshake protocol, talker/listener addressing procedures, device clearing and triggering, service request, and both serial and parallel polling. In most procedures, it does not disturb the microprocessor unless a byte has arrived (input buffer full) or has to be sent out (output buffer empty).

The 8291A architecture includes 16 registers. Eight of these registers may be written into by the microprocessor. The other eight registers may be read by the microprocessor. One each of these read and

write registers is for direct data transfers. The rest of the write registers control the various features of the chip, while the rest of the read registers provide the microprocessor with a monitor of GPIB states, various bus conditions, and device conditions.

### GPIB Addressing

Each device connected to the GPIB must have at least one address whereby the controller device in charge of the bus can configure it to talk, listen, or send status. An 8291A implementation of the GPIB offers the user three alternative addressing modes for which the device can be initialized for each application. The first of these modes allows for the device to have two separate primary addresses. The second mode allows the user to implement a single talker/listener with a two byte address (primary address + secondary address). The third mode again allows for two distinct addresses but in this instance, they can each have a ten-bit address (5 low-order bits of each of two bytes). However, this mode requires that the secondary addresses be passed to the microprocessor for verification. These three addressing schemes are described in more detail in the discussion of the Address Registers.

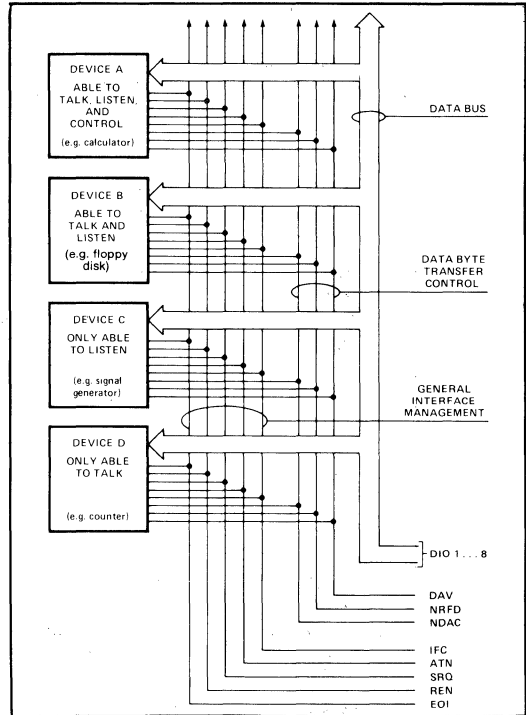


Figure 4. Interface Capabilities and Bus Structure

Table 2. IEEE 488 Interface State Mnemonics

Mnemonic	State Represented
ACDS	Accept Data State
ACRS	Acceptor Ready State
AIDS	Acceptor Idle State
ANRS	Acceptor Not Ready State
APRS	Affirmative Poll Response State
AWNS	Acceptor Wait for New Cycle State
CACS	Controller Active State
CADS	Controller Addressed State
CAWS	Controller Active Wait State
CIDS	Controller Idle State
CPPS	Controller Parallel Poll State
CPWS	Controller Parallel Poll Wait State
CSBS	Controller Standby State
CSNS	Controller Service Not Requested State
CSRS	Controller Service Requested State
CSWS	Controller Synchronous Wait State
CTRS	Controller Transfer State
DCAS	Device Clear Active State
DCIS	Device Clear Idle State
DTAS	Device Trigger Active State
DTIS	Device Trigger Idle State
LACS	Listener Active State
LADS	Listener Addressed State
LIDS	Listener Idle State
LOCS	Local State
LPAS	Listener Primary Addressed State
LPIS	Listener Primary Idle State
LWLS	Local With Lockout State
NPRS	Negative Poll Response State

Mnemonic	State Represented
PACS	Parallel Poll Addressed to Configure State
PPAS	Parallel Poll Active State
PPIS	Parallel Poll Idle State
PPSS	Parallel Poll Standby State
PUCS	Parallel Poll Unaddressed to Configure State
REMS	Remote State
RWLS	Remote With Lockout State
SACS	System Control Active State
SDYS	Source Delay State
SGNS	Source Generate State
SIAS	System Control Interface Clear Active State
SIDS	Source Idle State
SIIS	System Control Interface Clear Idle State
SINS	System Control Interface Clear Not Active State
SIWS	Source Idle Wait State
SNAS	System Control Not Active State
SPAS	Serial Poll Active State
SPIS	Serial Poll Idle State
SPMS	Serial Poll Mode State
SRAS	System Control Remote Enable Active State
SRIS	System Control Remote Enable Idle State
SRNS	System Control Remote Enable Not Active State
SRQS	Service Request State
STRS	Source Transfer State
SWNS	Source Wait for New Cycle State
TACS	Talker Active State
TADS	Talker Addressed State
TIDS	Talker Idle State
TPIS	Talker Primary Idle State

\*The Controller function is implemented on the Intel® 8292.



Table 3. IEEE 488 Interface Message Reference List

Mnemonic	Message	Interface Function(s)
LOCAL MESSAGES RECEIVED (By Interface Functions)		
<sup>1</sup> gts	go to standby	C
ist	individual status	PP
lon	listen only	L, LE
lpe	local poll enable	PP
nba	new byte available	SH
pon	power on	SH,AH,T,TE,L,LE,SR,RL,PP,C
rdy	ready	AH
<sup>1</sup> rpp	request parallel poll	C
<sup>1</sup> rsc	request system control	C
rsv	request service	SR
rtl	return to local	RL
<sup>1</sup> sic	send interface clear	C
<sup>1</sup> sre	send remote enable	C
<sup>1</sup> tca	take control asynchronously	C
<sup>1</sup> tcs	take control synchronously	AH, C
ton	talk only	T, TE
REMOTE MESSAGES RECEIVED		
ATN	Attention	SH,AH,T,TE,L,LE,PP,C
DAB	Data Byte	(Via L, LE)
DAC	Data Accepted	SH
DAV	Data Valid	AH
DCL	Device Clear	DC
END	End	(via L, LE)
GET	Group Execute Trigger	DT
GTL	Go to Local	RL
IDY	Identify	L,LE,PP
IFC	Interface Clear	T,TE,L,LE,C
LLO	Local Lockout	RL
MLA	My Listen Address	L,LE,RL,T,TE
MSA	My Secondary Address	TE,LE,RL
MTA	My Talk Address	T,TE,L,LE
OSA	Other Secondary Address	TE
OTA	Other Talk Address	T, TE
PCG	Primary Command Group	TE,LE,PP
<sup>2</sup> PPC	Parallel Poll Configure	PP
<sup>2</sup> [PPD]	Parallel Poll Disable	PP
<sup>2</sup> [PPE]	Parallel Poll Enable	PP
<sup>1</sup> PPR <sub>N</sub>	Parallel Poll Response N	(via C)
<sup>2</sup> PPU	Parallel Poll Unconfigure	PP
REN	Remote Enable	RL
RFD	Ready for Data	SH
RQS	Request Service	(via L, LE)
[SDC]	Select Device Clear	DC
SPD	Serial Poll Disable	T, TE
SPE	Serial Poll Enable	T, TE
<sup>1</sup> SQR	Service Request	(via C)
STB	Status Byte	(via L, LE)
<sup>1</sup> TCT or [TCT]	Take Control	C
UNL	Unlisten	L, LE

**NOTE:**

1. These messages are handled only by Intel's 8292.
2. Undefined commands which may be passed to the microprocessor.

**Table 3. (Cont'd)**  
**IEEE 488 Interface Message Reference List**

Mnemonic	Message	<sup>3</sup> Interface Function(s)
REMOTE MESSAGES SENT		
ATN	Attention	C
DAB	Data Byte	(via T, TE)
DAC	Data Accepted	AH
DAV	Data Valid	SH
DCL	Device Clear	(via C)
END	End	(via T)
GET	Group Execute Trigger	(via C)
GTL	Go to Local	(via C)
IDY	Identify	C
IFC	Interface Clear	C
LLO	Local Lockout	(via C)
MLA or [MLA]	My Listen Address	(via C)
MSA or [MSA]	My Secondary Address	(via C)
MTA or [MTA]	My Talk Address	(via C)
OSA	Other Secondary Address	(via C)
OTA	Other Talk Address	(via C)
PCG	Primary Command Group	(via C)
PPC	Parallel Poll Configure	(via C)
[PPD]	Parallel Poll Disable	(via C)
[PPE]	Parallel Poll Enable	(via C)
PPRN	Parallel Poll Response N	PP
PPU	Parallel Poll Unconfigure	(via C)
REN	Remote Enable	C
RFD	Ready for Data	AH
RQS	Request Service	T, TE
[SDC]	Selected Device Clear	(via C)
SPD	Serial Poll Disable	(via C)
SPE	Serial Poll Enable	(via C)
SRQ	Service Request	SR
STB	Status Byte	(via T, TE)
TCT	Take Control	(via C)
UNL	Unlisten	(via C)

**NOTE:**

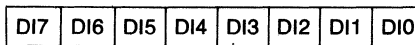
3. All Controller messages must be sent via Intel's 8292.

**8291A Registers**

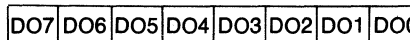
A bit-by-bit map of the 16 registers on the 8291A is presented in Figure 5. A more detailed explanation of each of these registers and their functions follows. The access of these registers by the microprocessor is accomplished by using the  $\overline{CS}$ , RD, WR, and  $RS_0$ - $RS_2$  pins.

Register	$\overline{CS}$	$\overline{RD}$	$\overline{WR}$	$RS_0$ - $RS_2$
All Read Registers	0	0	1	CCC
All Write Registers	0	1	0	CCC
High Impedance	1	d	d	ddd

**Data Registers**



DATA-IN REGISTER (0R)



DATA-OUT REGISTER (0W)

The Data-In Register is used to move data from the GPIB to the microprocessor or to memory when the 8291A is addressed to listen. Incoming information is separately latched by this register, and its contents are not destroyed by a write to the data-out

register. The RFD (Ready for Data) message is held false until the byte is removed from the data in register, either by the microprocessor or by DMA. The 8291A then completes the handshake automatically. In RFD holdoff mode (see Auxiliary Register A), the handshake is not finished until a command is sent telling the 8291A to release the holdoff. In this way, the same byte may be read several times, or an over anxious talker may be held off until all available data has been processed.

When the 8291A is addressed to talk, it uses the data-out register to move data onto the GPIB. After the BO interrupt is received and a byte is written to this register, the 8291A initiates and completes the handshake while sending the byte out over the bus. In the BO interrupt disable mode, the user should wait until BO is active before writing to the register. (In the DMA mode, this will happen automatically.) A read of the Data-In Register does not destroy the information in the Data-Out Register.

**Interrupt Registers**

CPT	APT	GET	END	DEC	ERR	BO	BI
-----	-----	-----	-----	-----	-----	----	----

INTERRUPT STATUS 1 (1R)

INT	SPAS	LLO	REM	SPC	LLOC	REMC	ADSC
-----	------	-----	-----	-----	------	------	------

INTERRUPT STATUS 2 (2R)

CPT	APT	GET	END	DEC	ERR	BO	BI
-----	-----	-----	-----	-----	-----	----	----

INTERRUPT ENABLE 1 (1W)

0	0	DMAO	DMAI	SPC	LLOC	REMC	ADSC
---	---	------	------	-----	------	------	------

INTERRUPT ENABLE 2 (2W)

INT	DT0	DL0	AD5-0	AD4-0	AD3-0	AD2-0	AD1-0
-----	-----	-----	-------	-------	-------	-------	-------

ADDRESS 0 REGISTER

**Figure 5. 8291A Registers**

READ REGISTERS								REGISTER SELECT CODE			WRITE REGISTERS															
								RS2	RS1	RS0																
DI7	DI6	DI5	DI4	DI3	DI2	DI1	DI0	0	0	0	DO7	DO6	DO5	DO4	DO3	DO2	DO1	DO0	DATA IN				DATA OUT			
CPT	APT	GET	END	DEC	ERR	BO	BI	0	0	1	CPT	APT	GET	END	DEC	ERR	BO	BI	INTERRUPT STATUS 1				INTERRUPT ENABLE 1			
INT	SPAS	LLO	REM	SPC	LLOC	REMC	ADSC	0	1	0	0	0	DMAO	DMAI	SPC	LLOC	REMC	ADSC	INTERRUPT STATUS 2				INTERRUPT ENABLE 2			
S8	SRQS	S6	S5	S4	S3	S2	S1	0	1	1	S8	rsv	S6	S5	S4	S3	S2	S1	SERIAL POLL STATUS				SERIAL POLL MODE			
ton	ion	EOI	LPAS	TPAS	LA	TA	MJMN	1	0	0	TO	LO	0	0	0	0	ADM1	ADM0	ADDRESS STATUS				ADDRESS MODE			
CPT7	CPT6	CPT5	CPT4	CPT3	CPT2	CPT1	CPT0	1	0	1	CNT2	CNT1	CNT0	COM4	COM3	COM2	COM1	COM0	COMMAND PASS THROUGH				AUX MODE			
INT	DT0	DL0	AD5-0	AD4-0	AD3-0	AD2-0	AD1-0	1	1	0	ARS	DT	DL	AD5	AD4	AD3	AD2	AD1	ADDRESS 0				ADDRESS 0/1			
X	DT1	DL1	AD5-1	AD4-1	AD3-1	AD2-1	AD1-1	1	1	1	EC7	EC6	EC5	EC4	EC3	EC2	EC1	EC0	ADDRESS 1				EOS			

The 8291A can be configured to generate an interrupt to the microprocessor upon the occurrence of any of 12 conditions or events on the GPIB. Upon receipt of an interrupt, the microprocessor must read the Interrupt Status Registers to determine which event has occurred, and then execute the appropriate service routine (if necessary). Each of the 12 interrupt status bits has a matching enable bit in the interrupt enable registers. These enable bits are used to select the events that will cause the INT pin to be asserted. Writing a logic "1" into any of these bits enables the corresponding interrupt status bits to generate an interrupt. Bits in the Interrupt Status Registers are set regardless of the states of the enable bits. The Interrupt Status Registers are then cleared upon being read or when a local pon (power-on) message is executed. If an event occurs while one of the Interrupt Status Registers is being read, the event is held until after its register is cleared and then placed in the register.

The mnemonics for each of the bits in these registers and a brief description of their respective functions appears in Table 4. This table also indicates how each of the interrupt bits is set.

**NOTE:** The INT bit in the Address 0 Register is a duplicate of the INT bit in the Interrupt Status 2 Register. It is only a status bit. It does not generate interrupts and thus does not have a corresponding enable bit.

The BO and BI interrupts enable the user to perform data transfer cycles. BO indicates that a data byte should be written to the Data Out Register. It is set by TACS · (SWNS + SGNS) · RFD. It is reset when the data byte is written, ATN is asserted, or the 8291A exits TACS. Data should never be written to the Data Out Register before BO is set. Similarly, BI is set when an input byte is accepted into the 8291A and reset when the microprocessor reads the Data In Register. BO and BI are also reset by pon (power-on local message) and by a read of the Interrupt

**Table 4. Interrupt Bits**

Indicates Undefined Commands	CPT	An undefined command has been received.
Set by (TPAS + LPAS)•SCG•ACDS•MODE 3	APT	A secondary address must be passed through to the microprocessor for recognition.
Set by DTAS	GET	A group execute trigger has occurred.
Set by (EOS + EOI)•LACS	END	An EOS or EOI message has been received.
Set by DCAS	DEC	Device Clear Active State has occurred.
Set by TACS•nba•DAC•RFD	ERR	Interface error has occurred; no listeners are active.
TACS•(SWNS + SGNS)	BO	A byte should be output.
Set by LACS•ACDS	BI	A byte has been input.
Shows status of the INT pin	INT	These are status only. They will <u>not</u> generate interrupts, nor do they have corresponding mask bits.
The device has been enabled for a serial poll	SPAS	
The device is in local lock out state. (LWLS+RWLS)	LLO	
The device is in a remote state. (REMS+RWLS)	REM	
SPAS → $\overline{\text{SPAS}}$ if APRS.STRS.SPAS was true	SPC	Serial Poll Complete interrupt.
LLO $\rightleftarrows$ NO LLO	LLOC	Local lock out change interrupt.
Remote $\rightleftarrows$ Local	REMC	Remote/Local change interrupt.
Addressed $\rightleftarrows$ Unaddressed	ADSC	Address status change interrupt. <sup>1</sup>

**NOTE:** <sup>1</sup>In ton (talk-only) and lon (listen-only) modes, no ADSC interrupt is generated.

Status 1 Register. However, if it is so desired, data transfer cycles may be performed without reading the Interrupt Status 1 Register if all interrupts except for BO or BI are disabled; BO and BI will automatically reset after each byte is transferred.

If the 8291A is used in the interrupt mode, the INT and DREQ pins can be dedicated to data input and output functions respectively by enabling BI and DMAO, provided that no other interrupts are enabled. This eliminates the need to read the interrupt status registers if a byte is received or transmitted.

The ERR bit is set to indicate the bus error condition when the 8291A is an active talker and tries to send a byte to the GPIB, but there are no active listeners (e.g., all devices on the GPIB are in AIDS). The logical equivalent of (nba · TACS · DAC · RFD) will set this bit.

The DEC bit is set whenever DCAS has occurred. The user must define a known state to which all device functions will return in DCAS. Typically this state will be a power-on state. However, the state of the device functions at DCAS is at the designer's discretion. It should be noted that DCAS has no effect on the interface functions which are returned to a known state by the IFC (interface clear) message or the pon local message.

The END interrupt bit may be used by the microprocessor to detect that a multi-byte transfer has been completed. The bit will be set when the 8291A is an active listener (LACS) and either EOS (provided the End on EOS Received feature is enabled in the Auxiliary Register A) or EOI is received. EOS will generate an interrupt when the byte in the Data In Register matches the byte in the EOS register. Otherwise the interrupt will be generated when a true input is detected on EOI.

The GET interrupt bit is used by the microprocessor to detect that DTAS has occurred. It is set by the 8291A when the GET message is received while it is addressed to listen. The TRIG output pin of the 8291A fires when the GET message is received. Thus, the basic operation of device trigger may be started without microprocessor software intervention.

The APT interrupt bit indicates to the processor that a secondary address is available in the CPT register for validation. This interrupt will only occur if Mode 3 addressing is in effect. (Refer to the section on addressing.) In Mode 2, secondary addresses will be recognized automatically on the 8291A. They will be ignored in Mode 1.

The CPT interrupt bit flags the occurrence of an undefined command and of all secondary commands following an undefined command. The Command Pass Through feature is enabled by the B0 bit of Auxiliary Register B. Any message not decoded by the 8291A (not included in the state diagrams in Appendix B) becomes an undefined command. Note that any addressed command is automatically ignored when the 8291A is not addressed.

Undefined commands are read by the CPU from the Command Pass Through register of the 8291A. This register reflects the logic levels present on the data lines at the time it is read. If the CPT feature is enabled, the 8291A will hold off the handshake until this register is read.

An especially useful feature of the 8291A is its ability to generate interrupts from state transitions in the interface functions. In particular, the lower 3 bits of the Interrupt Status 2 Register, if enabled by the corresponding enable bits, will cause an interrupt upon changes in the following states as defined in the IEEE 488 Standard.

Bit 0	ADSC	change in LIDS or TIDS or MJMN
Bit 1	REMC	change in LOCS or REMS
Bit 2	LLOC	change in LWLS or RWLS

The upper 4 bits of the Interrupt Status 2 Register are available to the processor as status bits. Thus, if one of the bits 0–2 generates an interrupt indicating a state change has taken place, the corresponding status bit (bits 3–5 may be read to determine what the new state is. To determine the nature of a change in addressed status (bit 0) the Address Status Register is available to be read. The SPC interrupt (bit 3 in Interrupt Status 2) is set upon exit from SPAS if APRS.STRS.SPAS occurred which indicates that the GPIB controller has read the bus serial poll status byte after the 8291A requested service (asserted SRQ). The SPC interrupt occurs once after the controller reads the status byte if service was requested.

The controller may read the status byte later, and the byte will contain the last status the 8291A's CPU wrote to the Serial Poll Mode Register, but the SRQS bit will not be set and no interrupt will be generated. Finally, bit 7 monitors the state of the 8291A INT pin. Logically, it is an OR of all enabled interrupt status bits. One should note that bits 3–6 of the Interrupt Status 2 Register do not generate interrupts, but are available only to be read as status bits by the processor. Bit 7 in Interrupt Status 2 is duplicated in Address 0 Register, and the latter should be used when polling for interrupts to avoid losing one of the interrupts in Interrupt Status 2 Register.

Bits 4 and 5 (DMAI, DMAO) of the Interrupt Mask 2 Register are available to enable direct data transfers between memory and the GPIB; DMAI (DMA in) enables the DREQ (DMA request) pin of the 8291A to be asserted upon the occurrence of BI. Similarly, DMAO (DMA out) enables the DREQ pin to be asserted upon the occurrence of BO. One might note that the DREQ pin may be used as a second interrupt output pin, monitoring BI and/or BO and enabled by DMAI and DMAO. One should note that the DREQ pin is not affected by a read of the Interrupt Status 1 Register. It is reset whenever a byte is written to the Data Out Register or read from the Data In Register.

To ensure that an interrupt status bit will not be cleared without being read, and will not remain un-cleared after being read, the 8291A implements a special interrupt handling procedure. When an enabled interrupt bit is set in either of the Interrupt Status Registers, the input of the registers are blocked until the set bit is read and reset by the microprocessor. Thus, potential problems arise when interrupt status changes while the register is being blocked. However, the 8291A stores all new interrupts in a temporary register and transfers them to the appropriate Interrupt Status Register after the interrupt has been reset. This transfer takes place only if the corresponding bits were read as zeroes.

**Serial Poll Registers**

S8	SRQS	S6	S5	S4	S3	S2	S1
----	------	----	----	----	----	----	----

SERIAL POLL STATUS (3R)

S8	rsv	S6	S5	S4	S3	S2	S1
----	-----	----	----	----	----	----	----

SERIAL POLL MODE (3W)

The Serial Poll Mode Register determines the status byte that the 8291A sends out on the GPIB data lines when it receives the SPE (Serial Poll Enable) message. Bit 6 of this register is reserved for the rsv (request service) local message. Setting this bit to 1 causes the 8291A to assert its  $\overline{SRQ}$  line, indicating its need for attention from the controller-in-charge of the GPIB. The other bits of this register are available for sending status information over the GPIB. Sometime after the microprocessor initiates a request for service by setting bit 6, the controller of the GPIB sends the SPE message and then addresses the 8291A to talk. At this point, one byte of status is returned by the 8291A via the Serial Poll Mode Register. After the status byte is read by the controller, rsv is automatically cleared by the 8291A and an SPC interrupt is generated. The CPU may request service again by writing another byte to the Serial Poll Mode Register with the rsv bit set. If the controller performs a serial poll when the rsv bit is clear, the last status byte written will be read, but the  $\overline{SRQ}$  line will not be driven by the 8291A and the SRQS bit will be clear in the status byte.

The Serial Poll Status Register is available for reading the status byte in the Serial Poll Mode Register. The processor may check the status of a request for service by polling bit 6 of this register, which corresponds to SRQS (Service Request State). When a Serial Poll is conducted and the controller-in-charge reads the status byte, the SRQS bit is cleared. The  $\overline{SRQ}$  line and the rsv bit are tied together.

**Address Registers**

ton	Ion	EOI	LPAS	TPAS	LA	TA	MJMN
-----	-----	-----	------	------	----	----	------

ADDRESS STATUS (4R)

INT	DT0	DL0	AD5-0	AD4-0	AD3-0	AD2-0	AD1-0
-----	-----	-----	-------	-------	-------	-------	-------

ADDRESS 0 (6R)

X	DT1	DL1	AD5-1	AD4-1	AD3-1	AD2-1	AD1-1
---	-----	-----	-------	-------	-------	-------	-------

ADDRESS 1 (7R)

TO	LO	0	0	0	0	ADM1	ADM0
----	----	---	---	---	---	------	------

ADDRESS MODE (4W)

ARS	DT	DL	AD5	AD4	AD3	AD2	AD1
-----	----	----	-----	-----	-----	-----	-----

ADDRESS 0/1 (6W)

The Address Mode Register is used to select one of the five modes of addressing available on the 8291A. It determines the way in which the 8291A uses the information in the Address 0 and Address 1 Registers.

—In Mode 1, the contents of the Address 0 Register constitute the “Major” talker/listener address while the Address 1 Register represents the “Minor” talker/listener address. In applications where only one address is needed, the major talker/listener is used, and the minor talker/listener should be disabled. Loading an address via the Address 0/1 Register into Address Registers 0 and 1 enables the major and minor talker/listener functions respectively.

—In Mode 2 the 8291A recognizes two sequential address bytes: a primary followed by a secondary. Both address bytes must be received in order to enable the device to talk or listen. In this manner, Mode 2 addressing implements the extended talker and listener functions as defined in IEEE-488.

To use Mode 2 addressing the primary address must be loaded into the Address 0 Register, and the Secondary Address is placed in the Address 1 Register. With both primary and secondary addresses residing on chip, the 8291A can handle all addressing sequences without processor intervention.

—In Mode 3, the 8291A handles addressing just as it does in Mode 1, except that each Major or Minor primary address must be followed by a secondary address. All secondary addresses must be verified by the microprocessor when Mode 3 is used. When the 8291A is in TPAS or LPAS (talker/listener primary addressed state), and it does not recognize the byte on the DIO lines, an APT interrupt is generated (see section on Interrupt Registers) and the byte is available in the CPT (Command Pass-Through) Register. As part of its interrupt service routine, the microprocessor must read the CPT Register and write one of the following responses to the Auxiliary Mode Register:

1. 07H implies a non-valid secondary address
2. 0FH implies a valid secondary address

Setting the TO bit generates the local ton (talk-only) message and sets the 8291A to a talk-only mode. This mode allows the device to operate as a talker in an interface system without a controller.

Setting the LO bit generates the local lon (listen-only) message and sets the 8291A to a listen-only mode. This mode allows the device to operate as a listener in an interface system without a controller. The above bits may also be used by a controller-in-charge to set itself up for remote command or data communication.

The mode of addressing implemented by the 8291A may be selected by writing one of the following bytes to the Address Mode Register.

Register Contents	Mode
10000000	Enable talk only mode (ton)
01000000	Enable listen only mode (lon)
11000000	The 8291 may talk to itself
00000001	Mode 1, (Primary-Primary)
00000010	Mode 2 (Primary-Secondary)
00000011	Mode 3 (Primary/APT-Primary/APT)

The Address Status Register contains information used by the microprocessor to handle its own addressing. This information includes status bits that monitor the address state of each talker/listener, “ton” and “lon” flags which indicate the talk and listen only states, and an EOI bit which, when set, signifies that the END message came with the last data byte. LPAS and TPAS indicate that the listener or talker primary address has been received. The microprocessor can use these bits when the secondary address is passed through to determine whether the 8291A is addressed to talk or listen. The LA (listener addressed) bit will be set when the 8291A is in LACS (Listener Active State) or in LADS (Listener Addressed State). Similarly, the TA (Talker Addressed bit) will be set to indicate TACS or TADS, but also to indicate SPAS (Serial Poll Active State). The MJMN bit is used to determine whether the information in the other bits applies to the Major or Minor talker/listener. It is set to “1” when the Minor talker/listener is addressed. It should be noted that only one talker/listener may be active at any one time. Thus, the MJMN bit will indicate which, if either, of the talker/listeners is addressed or active.

The Address 0/1 Register is used for specifying the device’s addresses according to the format selected in the Address Mode Register. Five bit addresses may be loaded into the Address 0 and Address 1 Registers by writing into the Address 0/1 Register. The ARS bit is used to select which of these registers the other seven bits will be loaded into. The DT and DL bits may be used to disable the talker or listener function at the address signified by the other five

bits. When Mode 1 addressing is used and only one primary address is desired, *both* the talker and the listener should be disabled at the Minor address.

As an example of how the Address 0/1 Register might be used, consider an example where two primary addresses are needed in the device. The Major primary address will be selectable only as a talker and the Minor primary address will be selectable only as a listener. This configuration of the 8291A is formed by the following sequence of writes by the microprocessor.

Operation	CS	RD	WR	Data	RS2-RS0
1. Select addressing Mode 1	0	1	0	00000001	100
2. Load major address into Address 0 Register with listener function disabled.	0	1	0	001AAAAA	110
3. Load minor address into Address 1 Register with talker function disabled.	0	1	0	110BBBBB	110

At this point, the addresses AAAAA and BBBBB are stored in the Address 0 and Address 1 Registers respectively, and are available to be read by the microprocessor. Thus, it is not necessary to store any address information elsewhere. Also, with the information stored in the Address 0 and Address 1 Registers, processor intervention is not required to recognize addressing by the controller. Only in Mode 3, where secondary addresses are passed through, must the processor intervene in the addressing sequence.

The Address 0 Register contains a copy of bit 7 of the Interrupt Status 2 Register (INT). This is to be used when polling for interrupts. Software should poll register 6 checking for INT (bit 7) to be set. When INT is set, the Interrupt Status Register should be read to determine which interrupt was received.

### Command Pass Through Register

CPT7	CPT6	CPT5	CPT4	CPT3	CPT2	CPT1	CPT0
------	------	------	------	------	------	------	------

COMMAND PASS THROUGH (5R)

The Command Pass Through Register is used to transfer undefined 8-bit remote message codes from the GPIB to the microprocessor. When the CPT feature is enabled (bit B0 in Auxiliary Register B), any message not decoded by the 8291A becomes an undefined command. When Mode 3 addressing is used secondary addresses are also passed through

the CPT Register. In either case, the 8291A will hold-off the handshake until the microprocessor reads this register and issues the VSCMD auxiliary command.

The CPT and APT interrupts flag the availability of undefined commands and secondary addresses in the CPT Register. The details of these interrupts are explained in the section on Interrupt Registers.

An added feature of the 8291A is its ability to handle undefined secondary commands following undefined primaries. Thus, the number of available commands for future IEEE-488 definition is increased; one undefined primary command followed by a sequence of as many as 32 secondary commands can be processed. The IEEE-488 Standard does not permit users to define their own commands, but upgrades of the standard are thus provided for.

The recommended use of the 8291A's undefined command capabilities is for a controller-configured Parallel Poll. The PPC message is an undefined primary command typically followed by PPE, an undefined secondary command. For details on this procedure, refer to the section on Parallel Poll Protocol.

### Auxiliary Mode Register

CNT2	CNT1	CNT0	COM4	COM3	COM2	COM1	COM0
------	------	------	------	------	------	------	------

AUX MODE (5W)

CNT0-2:CONTROL BITS  
COM0-4:COMMAND BITS

The Auxiliary Mode Register contains a three-bit control field and a five-bit command field. It is used for several purposes on the 8291A:

1. To load "hidden" auxiliary registers on the 8291A.
2. To issue commands from the microprocessor to the 8291A.
3. To preset an internal counter used to generate T1, delay in the Source Handshake function, as defined in IEEE-488.

Table 5 summarizes how these tasks are performed with the Auxiliary Mode Register. Note that the three control bits determine how the five command bits are interpreted.



Table 5

CODE		COMMAND
CONTROL BITS	COMMAND BITS	
000	0CCCC	Execute auxiliary command CCCC
001	ODDDD	Preset internal counter to match external clock frequency of DDDD MHz (DDDD binary representation of 1 to 8 MHz)
100	DDDDD	Write DDDDD into auxiliary register A
101	DDDDD	Write DDDDD into auxiliary register B
011	USP <sub>3</sub> P <sub>2</sub> P <sub>1</sub>	Enable/disable parallel poll either in response to remote messages (PPC followed by PPE or PPD) or as a local lpe message. (Enable if U = 0, disable if U = 1.)

## AUXILIARY COMMANDS

Auxiliary commands are executed by the 8291A whenever 0000CCCC is written into the Auxiliary Mode Register, where CCCC is the 4-bit command code.

**0000**—Immediate Execute pon: This command resets the 8291A to a power up state (local pon message as defined in IEEE-488).

The following conditions constitute the power up state:

1. All talkers and listeners are disabled.
2. No interrupt status bits are set.

The 8291A is designed to power up in certain states as specified in the IEEE-488 state diagrams. Thus, the following states are in effect in the power up state: SIDS, AIDS, TIDS, LIDS, NPRS, LOCS, and PPIS.

The "0000" pon is an immediate execute command (a pon pulse). It is also used to release the "initialize" state generated by either an external reset pulse or the "0010" Chip Reset command.

**0010**—Chip Reset (Initialize): This command has the same effect as a pulse applied to the Reset pin. (Refer to the section on Reset Procedure.)

**0011**—Finish Handshake : This command finishes a handshake that was stopped because of a holdoff on RFD. (Refer to Auxiliary Register A.)

**0100**—Trigger: A "Group Execute Trigger" is forced by this command. It has the same effect as a GET command issued by the controller-in-charge of the GPIB, but does not cause a GET interrupt.

**0101, 1101**—Clear/Set rtl: These commands correspond to the local rtl message as defined by the IEEE-488. The 8291A will go into local mode when a Set rtl Auxiliary Command is received if local lockout is not in effect. The 8291A will exit local mode after receiving a Clear rtl Auxiliary Command if the 8291A is addressed to listen.

**0110**—Send EOI: The EOI line of the 8291A may be asserted with this command. The command causes EOI to go true with the next byte transmitted. The EOI line is then cleared upon completion of the handshake for that byte.

**0111, 1111**—Non Valid/Valid Secondary Address or Command (VSCMD): This command informs the 8291A that the secondary address received by the microprocessor was valid or invalid (0111 = invalid, 1111 = valid). If Mode 3 addressing is used, the processor must field each extended address and respond to it, or the GPIB will hang up. Note that the COM3 bit is the invalid/valid flag.

The valid (1111) command is also used to tell the 8291A to continue from the command-pass-through-state, or from RFD holdoff on GET, SDC or DCL.

**1000**—pon: This command puts the 8291A into the pon (power on) state and holds it there. It is similar to a Chip Reset except none of the Auxiliary Mode Registers are cleared. In this state, the 8291A does not participate in any bus activity. An Immediate Execute pon releases the 8291A from the pon state and permits the device to participate in the bus activity again.

**0001, 1001**—Parallel Poll Flag (local "ist" message): This command sets (1001) or clears (0001) the parallel poll flag. A "1" is sent over the assigned data line (PRR = Parallel Poll Response true) only if the parallel poll flag matches the sense bit from the lpe local message (or indirectly from the PPE message). For a more complete description of the Parallel Poll features and procedures refer to the section on Parallel Poll Protocol.

## INTERNAL COUNTER

The internal counter determines the delay time allowed for the setting of data on the DIO lines. This delay time is defined as  $T_i$  in IEEE-488 and appears in the Source Handshake state diagram between the

SDYS and STRS. As such, DAV is asserted  $T_1$  after the DIO lines are driven. Consequently,  $T_1$  is a major factor in determining the data transfer rate of the 8291A over the GPIB ( $T_1 = \text{TWRDV2-TWRD15}$ ).

When open-collector transceivers are used for connection to the GPIB,  $T_1$  is defined by IEEE-488 to be  $2\mu\text{sec}$ . By writing 0010DDDD into the Auxiliary Mode Register, the counter is preset to match a  $f_c$  MHz clock input, where DDDD is the binary representation of  $N_F$  [ $1 \leq N_F \leq 8$ ,  $N_F = (\text{DDDD})_2$ ]. When  $N_F = f_c$ , a  $2\mu\text{sec}$   $T_1$  delay will be generated before each DAV asserted.

$$T_{1(\mu\text{sec})} = \frac{2N_F}{f_c} + t_{\text{SYNC}}, \quad 1 \leq N_F \leq 8$$

$t_{\text{SYNC}}$  is a synchronization error, greater than zero and smaller than the larger of T clock high and T clock low. (For a 50% duty cycle clock,  $t_{\text{SYNC}}$  is less than half the clock cycle).

If it is necessary that  $T_1$  be different from  $2\mu\text{sec}$ ,  $N_F$  may be set to a value other than  $f_c$ . In this manner, data transfer rates may be programmed for a given system. In small systems, for example, where transfer rates exceeding GPIB specifications are required, one may set  $N_F < f_c$  and decrease  $T_1$ .

When tri-state transceivers are used, IEEE-488 allows a higher transfer rate (lower  $T_1$ ). Use of the 8291A with such transceivers is enabled by setting  $B_2$  in Auxiliary Register B. In this case, setting  $N_F = f_c$  causes a  $T_1$  delay of  $2\mu\text{sec}$  to be generated for the first byte transmitted — all subsequent bytes will have a delay of 500 nsec.

$$T_1(\text{High Speed}) \mu\text{sec} = \frac{N_F}{2f_c} + t_{\text{SYNC}}$$

Thus, the shortest  $T_1$  is achieved by setting  $N_F = 1$  using an 8 MHz clock with a 50% duty cycle clock ( $t_{\text{SYNC}} < 63 \text{ nsec}$ ):

$$T_{1(\text{HS})} = \frac{1}{2 \times 8} + 0.063 = 125 \text{ nsec max.}$$

## AUXILIARY REGISTER A

Auxiliary Register A is a "hidden" 5-bit register which is used to enable some of the 8291A features. Whenever a 100  $A_4A_3A_2A_1A_0$  byte is written into the

Auxiliary Register, it is loaded with the data  $A_4A_3A_2A_1A_0$ . Setting the respective bits to "1" enables the following features.

**A<sub>0</sub>**—RFD Holdoff on all Data: If the 8291A is listening, RFD will not be sent true until the "finish handshake" auxiliary command is issued by the microprocessor. The holdoff will be in effect for each data byte.

**A<sub>1</sub>**—RFD Holdoff on End: This feature enables the holdoff on EOI or EOS (if enabled). However, no holdoff will be in effect on any other data bytes.

**A<sub>2</sub>**—End on EOS Received: Whenever the byte in the Data In Register matches the byte in the EOS Register, the END interrupt bit will be set in the Interrupt Status 1 Register.

**A<sub>3</sub>**—Output EOI on EOS Sent: Any occurrence of data in the Data Out Register matching the EOS Register causes the EOI line to be sent true along with the data.

**A<sub>4</sub>**—EOS Binary Compare: Setting this bit causes the EOS Register to function as a full 8-bit word. When it is not set, the EOS Register is a 7-bit word (for ASCII characters).

If  $A_0 = A_1 = 1$ , a special "continuous Acceptor Handshake cycling" mode is enabled. This mode should be used only in a controller system configuration, where both the 8291A and the 8292 are used. It provides a continuous cycling through the Acceptor Handshake state diagram, requiring no local messages from the microprocessor; the rdy local message is automatically generated when in ANRS. As such, the 8291A Acceptor Handshake serves as the controller Acceptor Handshake. Thus, the controller cycles through the Acceptor Handshake without delaying the data transfer in progress. When the tcs local message is executed, the 8291A should be taken out of the "continuous AH cycling" mode, the GPIB will hang up in ANRS, and a BI interrupt will be generated to indicate that control may be taken. A simpler procedure may be used when a "tcs on end of block" is executed; the 8291A may stay in "continuous AH cycling". Upon the end of a block (EOI or EOS received), a holdoff is generated, the GPIB hangs up in ANRS, and control may be taken.

## AUXILIARY REGISTER B

Auxiliary Register B is a "hidden" 4-bit register which is used to enable some of the features of the 8291A. Whenever a 101  $B_4B_3B_2B_1B_0$  is written into the Auxiliary Mode Register, it is loaded with the data  $B_4B_3B_2B_1B_0$ . Setting the respective bits to "1" enables the following features:

**B<sub>0</sub>**—Enable Undefined Command Pass Through: This feature allows any commands not recognized by the 8291A to be handled in software. If enabled, this feature will cause the 8291A to holdoff the handshake when an undefined command is received. The microprocessor must then read the command from the Command Pass Through Register and send the VSCMD auxiliary command. Until the VSCMD command is sent, the handshake holdoff will be in effect.

**B<sub>1</sub>**—Send EOI in SPAS: This bit enables EOI to be sent with the status byte; EOI is sent true in Serial Poll Active State. Otherwise, EOI is sent false in SPAS.

**B<sub>2</sub>**—Enable High Speed Data Transfer: This feature may be enabled when tri-state external transceivers are used. The data transfer rate is limited by  $T_1$ , delay time generated in the Source Handshake function, which is defined according to the type of transceivers used. When the "High Speed" feature is enabled,  $T_1 = 2$  microseconds is generated for the first byte transmitted after each true to false transition of ATN. For all subsequent bytes,  $T_1 = 500$  nanoseconds. Refer to the Internal Counter section for an explanation of  $T_1$  duration as a function of  $B_2$  and of clock frequency.

**B<sub>3</sub>**—Enable Active Low Interrupt: Setting this bit causes the polarity of the INT pin to be reversed, providing an output signal compatible with Intel's MCS-48® Family. Interrupt registers are not affected by this bit.

**B<sub>4</sub>**—Enable RFD Holdoff on GET or DEC: Setting this bit causes RFD to be held false until the "VSCMD" auxiliary command is written after GET, SDC, and DCL commands. This allows the device to hold off the bus until it has completed a clear or trigger similar to an unrecognized command.

## PARALLEL POLL PROTOCOL

Writing a 011USP<sub>3</sub>P<sub>2</sub>P<sub>1</sub> into the Auxiliary Mode Register will enable (U=0) or disable (U=1) the 8291A for a parallel poll. When U=0, this command is the "lpe" (local poll enable) local message as defined in IEEE-488. The "S" bit is the sense in which the 8291A is enabled; only if the Parallel Poll Flag ("ist" local message) matches this bit will the Parallel Poll Response, PPR<sub>N</sub>, be sent true (Response = S + ist). The bits P<sub>3</sub>P<sub>2</sub>P<sub>1</sub> specify which of the eight data lines PPR<sub>N</sub> will be sent over. Thus, once the 8291A has been configured for Parallel Poll, whenever it senses both EOI and ATN true, it will automatically compare its PP flag with the sense bit and send PPR<sub>N</sub> true or false according to the comparison.

If a PP2\* implementation is desired, the "lpe" and "ist" local messages are all that are needed. Typically, the user will configure the 8291A for Parallel Poll immediately after initialization. During normal operation the microprocessor will set or clear the Parallel Poll Flag (ist) according to the device's need for service. Consequently the 8291A will be set up to give the proper response to IDY (EOI · ATN) without directly involving the microprocessor.

If a PP1\* implementation is desired, the undefined command features of the 8291A must be used. In PP1, the 8291A is indirectly configured for Parallel Poll by the active controller on the GPIB. The sequence at the 8291A being enabled or disabled remotely is as follows:

1. The PPC message is received and is loaded into the Command Pass Through Register as an undefined command. A CPT Interrupt is sent to the microprocessor; the handshake is automatically held off.
2. The microprocessor reads the CPT Register and sends VSCMD to the 8291A, releasing the handshake.
3. Having received an undefined primary command, the 8291A is set up to receive an undefined secondary command (the PPE or PPD message). This message is also received into the CPT Register, the handshake is held off, and the CPT interrupt is generated.

**NOTE:** \*As defined in IEEE Standard 488.

- The microprocessor reads the PPE or PPD message and writes the command into the Auxiliary Mode Register (bit 7 should be cleared first). Finally, the microprocessor sends VSCMD and the handshake is released.

### End of Sequence (EOS) Register

EC7	EC6	EC5	EC4	EC3	EC2	EC1	EC0
-----	-----	-----	-----	-----	-----	-----	-----

EOS REGISTER

The EOS Register and its features offer an alternative to the "Send EOI" auxiliary command. A seven or eight bit byte (ASCII or binary) may be placed in the register to flag the end of a block or read. The type of EOS byte to be used is selected in Auxiliary Register bit  $A_4$ .

If the 8291A is a listener, and the "End on EOS Received" is enabled with bit  $A_2$ , then an END interrupt is generated in the Interrupt Status 1 Register whenever the byte in the Data-In Register matches the byte in the EOS Register.

If the 8291A is a talker, and the "Output EOI on EOS Sent" is enabled with bit  $A_3$ , then the EOI line is sent true with the next byte whenever the contents of the Data Out Register match the EOS register.

### Reset Procedure

The 8291A is reset to an initialization state either by a pulse applied to its Reset pin, or by a reset auxiliary command (02H written into the Auxiliary Command Register). The following conditions are caused by a reset pulse (or local reset command):

- A "pon" local message as defined by IEEE-488 is held true until the initialization state is released.
- The Interrupt Status Registers are cleared (not Interrupt Enable Registers).
- Auxiliary Registers A and B are cleared.
- The Serial Poll Mode Register is cleared.
- The Parallel Poll Flag is cleared.
- The EOI bit in the Address Status Register is cleared.
- $N_F$  in the Internal Counter is set to 8 MHz. This setting causes the longest possible  $T_1$  delay to be generated in the Source Handshake (16  $\mu$ sec for 1 MHz clock).
- The rdy local message is sent.

The initialization state is released by an "immediate execute pon" command (00H written into the Auxiliary Command Register).

The suggested initialization sequence is:

- Apply a reset pulse or send the reset auxiliary command.
- Set the desired initial conditions by writing into the Interrupt Enable, Serial Poll Mode, Address Mode, Address 0/1, and EOS Registers. Auxiliary Registers A and B, and the internal counter should also be initialized.
- Send the "immediate execute pon" auxiliary command to release the initialization state.
- If a PP2 Parallel Poll implementation is to be used the "lpe" local message may be sent, enabling the 8291A for a Parallel Poll Response on an assigned line. (Refer to the section on Parallel Poll Protocol.)

### Using DMA

The 8291A may be connected to the Intel® 8237 or 8257 DMA Controllers or the 8089 I/O Processor for DMA operation. The 8237 will be used to refer to any DMA controller. The DREQ pin of the 8291A requests a DMA byte transfer from the 8237. It is set by BO or BI flip flops, enabled by the DMAO and DMAI bits in the Interrupt Enable 2 Register. (After reading, the INT1 register BO and BI interrupts will be cleared but not BO and BI in DREQ equation.)

The  $\overline{DACK}$  pin is driven by the 8237 in response to the DMA request. When  $\overline{DACK}$  is true (active low) it sets  $\overline{CS} = RS0 = RS1 = RS2 = 0$  such that the  $\overline{RD}$  and  $\overline{WR}$  signals sent by the 8237 refer to the Data In and Data Out Registers. Also, the DMA request line is reset by  $\overline{DACK} (\overline{RD} + \overline{WR})$ .

DMA input sequence:

- A data byte is accepted from the GPIB by the 8291A.
- A BI interrupt is generated and DREQ is set.
- $\overline{DACK}$  and  $\overline{RD}$  are driven by the 8237, the contents of the Data In Register are transferred to the system bus, and DREQ is reset.
- The 8291A sends RFD true on the GPIB and proceeds with the Acceptor Handshake protocol.

DMA output sequence:

- A BO interrupt is generated (indicating that a byte should be output) and DREQ is asserted.

2.  $\overline{DACK}$  and  $\overline{WR}$  are driven by the 8237, a byte is transferred from the MCS bus into the Data Out Register, and DREQ is reset.
3. The 8291A sends DAV true on the GPIB and proceeds with the Source Handshake protocol.

It should be noted that each time the device is addressed ( $MTA + MLA + ton + Ion$ ), the Address Status Register should be read, and the 8237 should be initialized accordingly. (Refer to the 8237 or 8257 Data Sheets.)

## APPLICATION BRIEF

### System Configuration

#### MICROPROCESSOR BUS CONNECTION

The 8291A is 8048/49, 8051, 8080/85, and 8086/88

compatible. The three address pins ( $RS_0, RS_1, RS_2$ ) should be connected to the non-multiplexed address bus (for example:  $A_8, A_9, A_{10}$ ). In case of 8080, any address lines may be used. If the three lowest address bits are used ( $A_0, A_1, A_2$ ), then they must be demultiplexed first.

#### EXTERNAL TRANSCEIVERS CONNECTION

The 8293 GPIB Transceiver interfaces the 8291A directly to the IEEE-488 bus. The 8291A and two 8293's can be configured as a talker/listener (see Figure 6) or with the 8292 as a talker/listener/controller (see Figure 7). Absolutely no active or passive external components are required to comply with the complete IEEE-488 electrical specification.

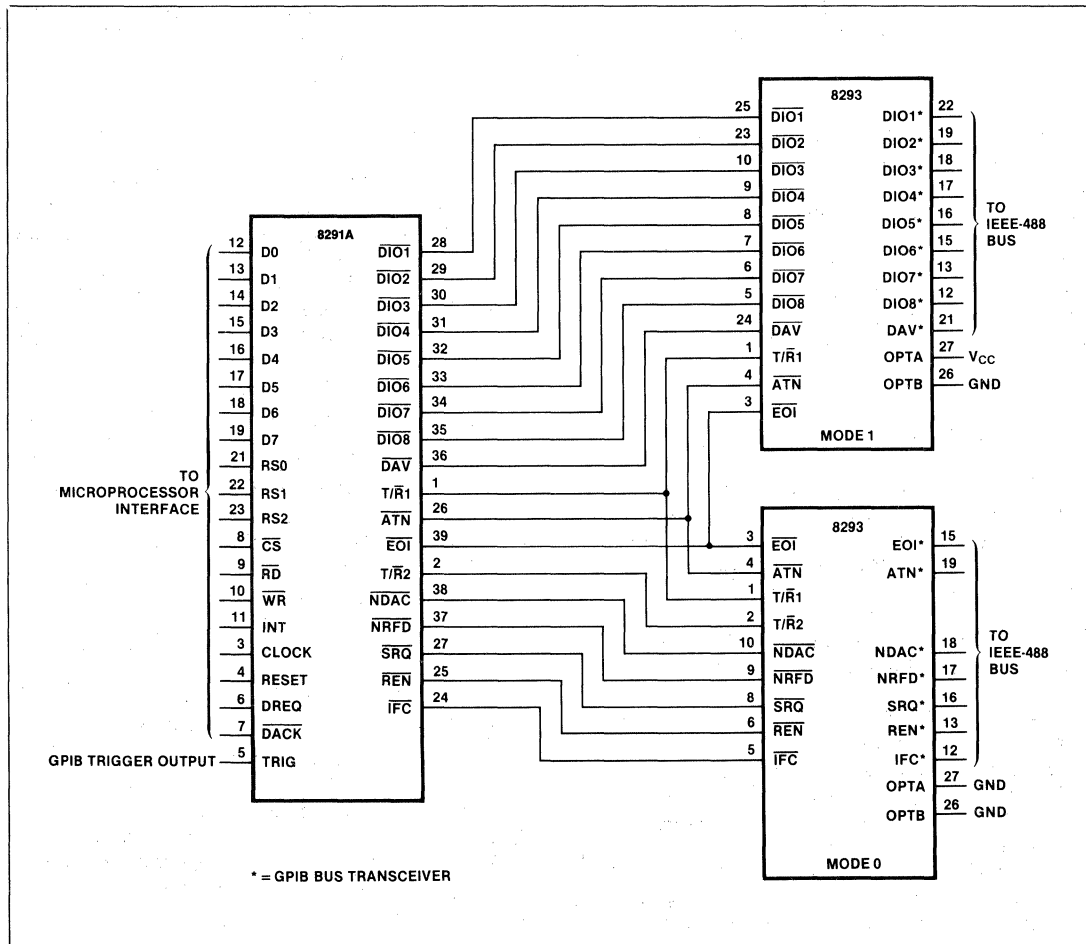


Figure 6. 8291A and 8293 System Configuration

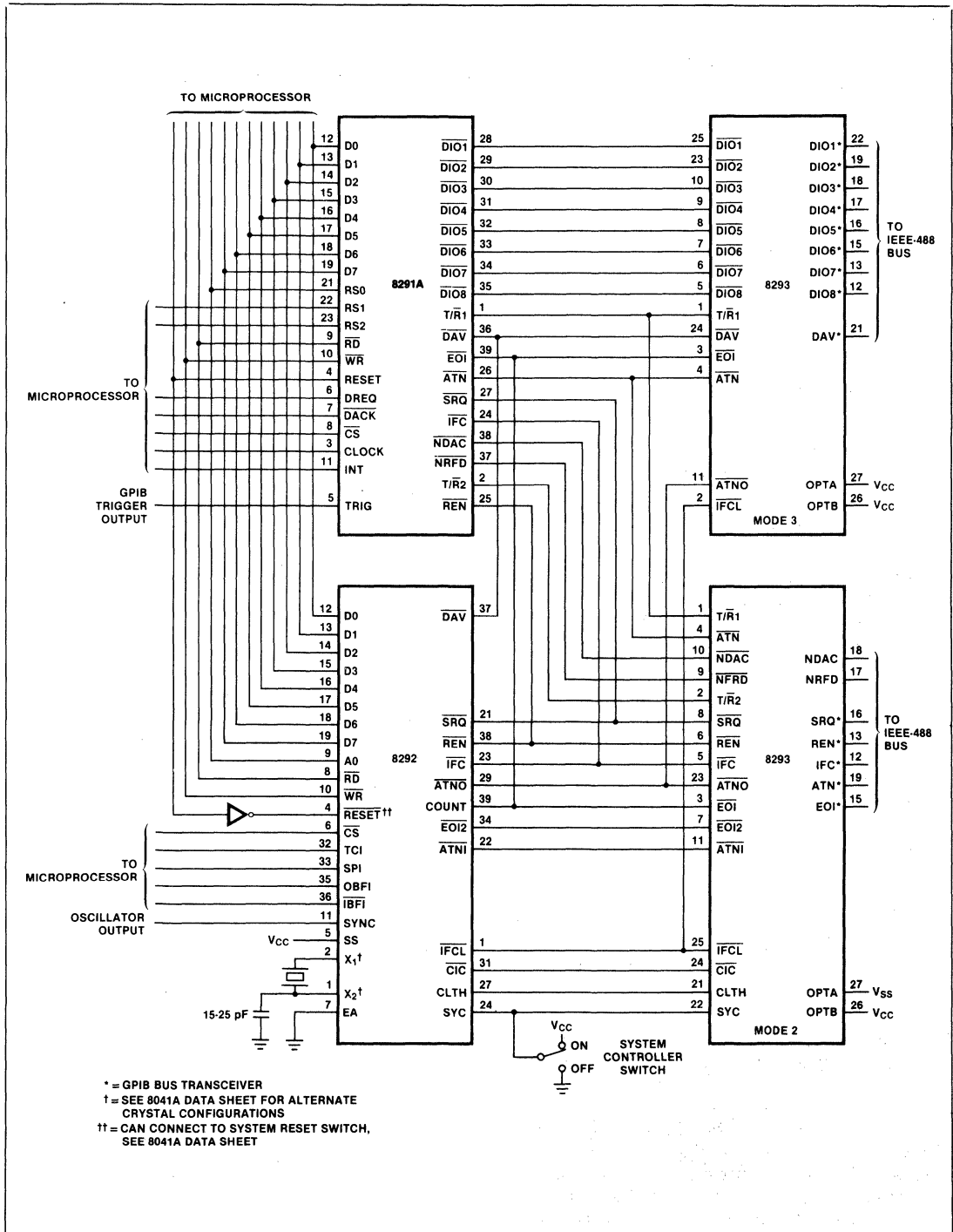


Figure 7. 8291A, 8292, and 8293 System Configuration

## Start-Up Procedures

The following section describes the steps needed to initialize a typical 8291A system implementing a talker/listener interface and an 8291A/8292 system implementing a talker/listener/controller interface.

### TALKER/LISTENER SYSTEM

Assume a general system configuration with the following features: (i) Polled system interface; (ii) Mode 1 addressing; (iii) same address for talker and listener; (iv) ASCII carriage return as the end-of-sequence (EOS) character; (v) EOI sent true with the last byte; and, (vi) 8 MHz clock.

**Initialization.** Initialization is accomplished with the following steps:

1. Pulse the RESET input or write 02H to the Auxiliary Mode Register.
2. Write 00H to the Interrupt Enable Registers 1 and 2. This disables interrupt and DMA.
3. Write 01H to the Address Mode Register to select Mode 1 addressing.
4. Write 28H to the Auxiliary Mode Register. This loads 8H to the Auxiliary Register A matching the 8 MHz clock input to the internal T1 delay counter to generate the delay meeting the IEEE spec.
5. Write the talker/listener address to the Address 0/1 register. The three most significant bits are zero.
6. Write an ASCII carriage return (0DH) to the EOS register.
7. Write 84H to the Auxiliary Mode Register to allow  $\overline{EOI}$  to be sent true when the EOS character is sent.
8. Write 00H to the Auxiliary Mode Register. This writes the "Immediate Execute pon" message and takes the 8291A from the initialization state into the idle state. The 8291A will remain idle until the controller initiates some activity by driving  $\overline{ATN}$  true.

**Communication.** The local CPU now polls the 8291A to determine which controller command has been received.

The controller addresses the 8291A by driving  $\overline{ATN}$ , placing MLA (My Listen Address) on the bus and driving  $\overline{DAV}$ . If the lower five bits of the MLA message match the address programmed into the Address 0/1 register, the 8291A is addressed to listen. It would be addressed to talk if the controller sent the MTA message instead of MLA.

The ADSC bit in the Interrupt Status 2 Register indicates that the 8291A has been addressed or unaddressed. The TA and LA bits in the Address Status Register indicate whether the 8291A is talker (TA=1), listener (LA=1), both (TA=LA=1) or unaddressed (TA=LA=0).

If the 8291A is addressed to listen, the local CPU can read the Data-In Register whenever the BI (Byte In) interrupt occurs in the Interrupt Status 1 Register. If the END bit in the same register is also set, either EOI or a data byte matching the pattern in the EOS register has been received.

In the talker mode, the CPU writes data into the Byte-Out Register on BO (Byte Out) true.

### TALKER/LISTENER/CONTROLLER SYSTEM

Combined with the Intel 8292, the 8291A executes a complete IEEE-488-1978 controller function. The 8291A talks and listens via the data and handshake lines (NRFD, NDAC and DAV). The 8292 controls four of the five bus management lines (IFC,  $\overline{SRQ}$ ,  $\overline{ATN}$  and REN).  $\overline{EOI}$ , the fifth line, is shared. The 8291A drives and receives  $\overline{EOI}$  when  $\overline{EOI}$  is used as an end-of-block indicator. The 8292 drives  $\overline{EOI}$  along with  $\overline{ATN}$  during a parallel poll command.

Once again, assume a general system configuration with the following features: (i) Polled system interface; (ii) 8292 as the system controller and controller-in-charge; (iii) ASCII carriage return (0DH) as the EOS identifier; (iv)  $\overline{EOI}$  sent with the last character; and, (v) an external buffer (8282) used to monitor the TCI line.

**Initialization.** In order to send a command across the GPIB, the 8292 has to drive  $\overline{ATN}$ , and the 8291A has to drive the data lines. Both devices therefore need initialization.

To initialize the 8292:

1. Pulse the RESET input. The 8292 will initially drive all outputs high. TCI, SPI, OBF, IBFI and CLTH will then go low. The Interrupt Status, Interrupt Mask, Error Flag, Error Mask and Timeout registers will be cleared. The interrupt counter will be disabled and loaded with 255. The 8292 will then monitor the status of the SYC pin. If high, the 8292 will pulse IFC true for at least 100 $\mu$ s in compliance with the IEEE-488-1978 standard. It will then take control by asserting  $\overline{ATN}$ .

To initialize the 8291A, the following is necessary:

1. Write 00H to Interrupt Enable registers 1 and 2. This disables interrupt and DMA.

2. With the 8292 as the controller-in-charge, it is impossible to address the 8292 via the GPIB. Therefore, the ton or lon modes of the 8291A must be used. To send commands, set the 8291A in the ton mode by writing 80H to the Address Mode Register.
3. Write 26H to the Auxiliary Mode Register to match the T1 data settling time to the 6 MHz clock input.
4. Write an ASCII carriage return (0DH) to the EOS Register.
5. Write 84H to the Auxiliary Mode Register in order to enable "Output EOI on EOS sent" and thus send EOI with the last character.
6. Write 00H—Immediate Execute pon—to the Auxiliary Mode Register to put the 8291A in the idle state.

**Communication.** Since the 8291A is in the ton mode, a BO interrupt is generated as soon as the immediate Execute pon command is written. The CPU writes the command into the Data Out Register, and repeats it on BO becoming true for as many commands as necessary.  $\overline{ATN}$  remains continuously

true unless the GTSB (Go To Standby) command is sent to the 8292.

$\overline{ATN}$  has to be false in order to send data rather than commands from the controller. To do this, the following steps are needed:

1. Enable the TCI interrupt if not already enabled.
2. Wait for IBF (Input Buffer Full) in the 8292 Interrupt Status Register to be reset.
3. Write the GTSB (F6H) command to the 8292 Command Field Register.
4. Read the 8282 and wait for TCI to be true.
5. Write the ton (80H) and pon (00H) command to the 8291A Address Mode Register and Auxiliary Mode Registers respectively.
6. Wait for the BO interrupt to be set in the 8291A.
7. Write the data to the 8291A Data-Out Register.

Identically, the user could command the controller to listen rather than talk. To do that, write lon (40H) instead of ton into the Address Mode Register. Then wait for BI rather than BO to go true. Read the data Register.



**ABSOLUTE MAXIMUM RATINGS**

Ambient Temperature Under Bias ..... 0°C to 70°C  
 Storage Temperature ..... -65°C to +150°C  
 Voltage on Any Pin  
 With Respect to Ground ..... -0.5V to +7V  
 Power Dissipation ..... 0.65 Watts

*\*NOTICE: Stresses above those listed under "Absolute Maximum Ratings" may cause permanent damage to the device. This is a stress rating only and functional operation of the device at these or any other conditions above those indicated in the operational sections of this specification is not implied. Exposure to absolute maximum rating conditions for extended periods may affect device reliability.*

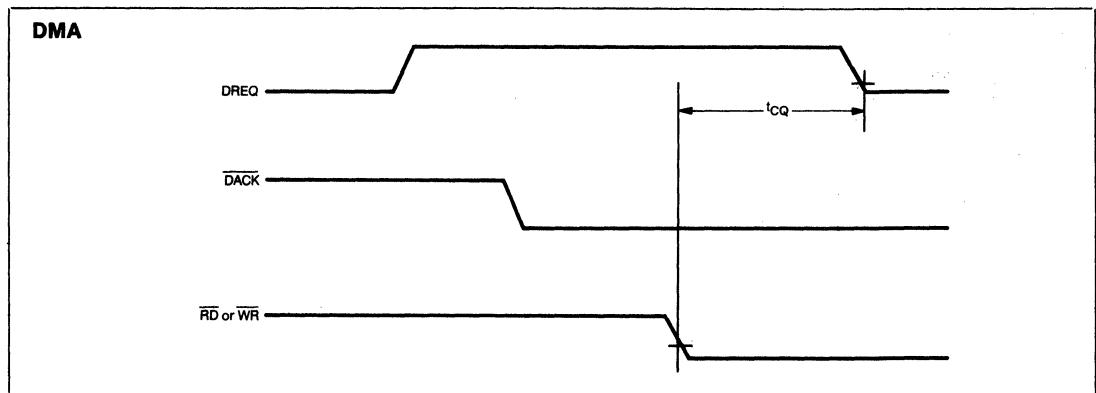
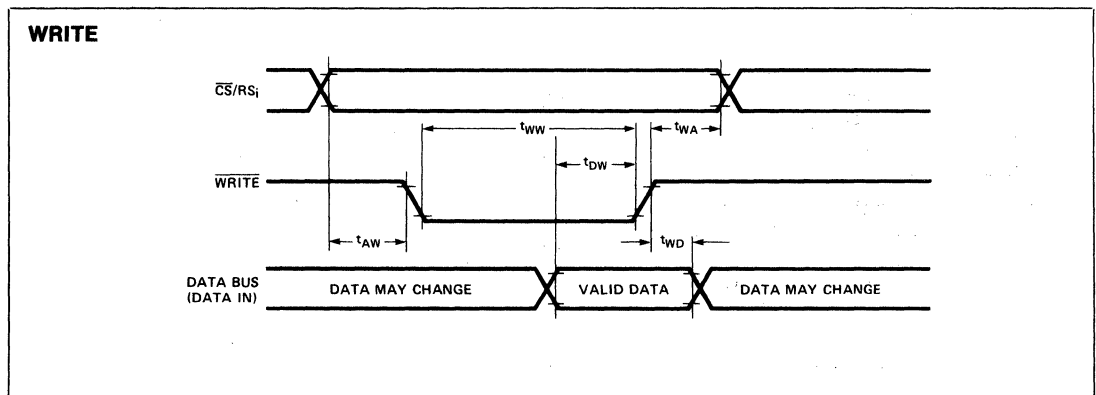
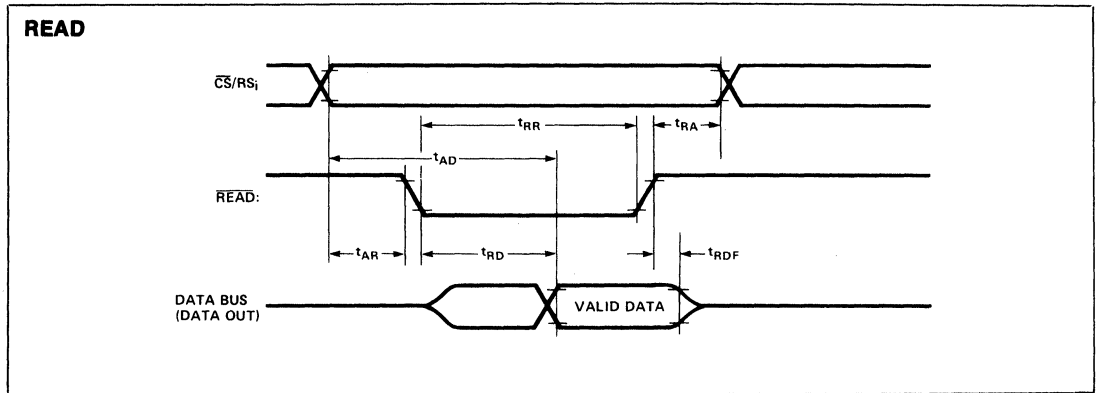
**D.C. CHARACTERISTICS** [ $V_{CC} = 5V \pm 10\%$ ,  $T_A = 0^\circ C$  to  $70^\circ C$  (Commercial)]

Symbol	Parameter	Min.	Max.	Unit	Test Conditions
V <sub>IL</sub>	Input Low Voltage	-0.5	0.8	V	
V <sub>IH</sub>	Input High Voltage	2	V <sub>CC</sub> +0.5	V	
V <sub>OL</sub>	Output Low Voltage		0.45	V	I <sub>OL</sub> =2mA (4mA for TR1 pin)
V <sub>OH</sub>	Output High Voltage	2.4		V	I <sub>OH</sub> = -400µA (-150µA for SRQ pin)
V <sub>OH-INT</sub>	Interrupt Output High Voltage	2.4		V	I <sub>OH</sub> =-400µA
		3.5		V	I <sub>OH</sub> =-50µA
I <sub>IL</sub>	Input Leakage		10	µA	V <sub>IN</sub> =0V to V <sub>CC</sub>
I <sub>LOL</sub>	Output Leakage Current		-10	µA	V <sub>OUT</sub> =0.45V
I <sub>LOH</sub>	Output Leakage Current		10	µA	V <sub>OUT</sub> =V <sub>CC</sub>
I <sub>CC</sub>	V <sub>CC</sub> Supply Current		120	mA	T <sub>A</sub> =0°C

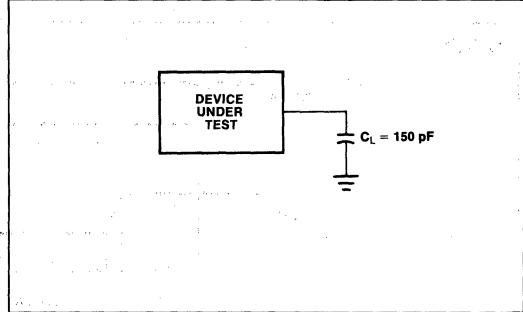
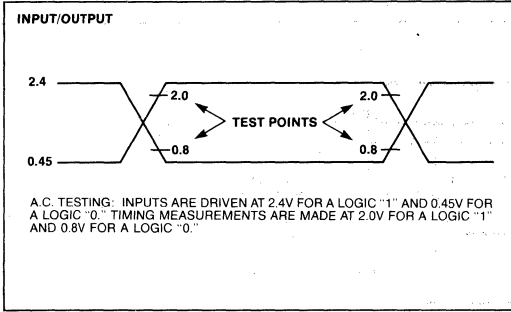
**A.C. CHARACTERISTICS** [ $V_{CC} = 5V \pm 10\%$ ,  $T_A = 0^\circ C$  to  $70^\circ C$  (Commercial)]

Symbol	Parameter	Min.	Max.	Unit	Test Conditions
t <sub>AR</sub>	Address Stable Before $\overline{READ}$	0		nsec	
t <sub>RA</sub>	Address Hold After $\overline{READ}$	0		nsec	
t <sub>RR</sub>	$\overline{READ}$ width	140		nsec	
t <sub>AD</sub>	Address Stable to Data Valid		250	nsec	
t <sub>RD</sub>	$\overline{READ}$ to Data Valid		100	nsec	
t <sub>RDF</sub>	Data Float After $\overline{READ}$	0	60	nsec	
t <sub>AW</sub>	Address Stable Before $\overline{WRITE}$	0		nsec	
t <sub>WA</sub>	Address Hold After $\overline{WRITE}$	0			
t <sub>WW</sub>	$\overline{WRITE}$ Width	170		nsec	
t <sub>DW</sub>	Data Set Up Time to the Trailing Edge of $\overline{WRITE}$	130		nsec	
t <sub>WD</sub>	Data Hold Time After $\overline{WRITE}$	0		nsec	
t <sub>DKDR4</sub>	$\overline{RD}\downarrow$ or $\overline{WR}\downarrow$ to $\overline{DREQ}\downarrow$		130	nsec	
t <sub>DKDA6</sub>	$\overline{RD}\downarrow$ to Valid Data (D <sub>0</sub> -D <sub>7</sub> )		200	nsec	$\overline{DACK}\downarrow$ to $\overline{RD}\downarrow$ 0 ≤ t ≤ 50nsec

WAVEFORMS



**A.C. TIMING MEASUREMENT POINTS AND LOAD CONDITIONS**



**GPIO TIMINGS<sup>1</sup>**

Symbol	Parameter	Max.	Units	Test Conditions
TEOT13 <sup>2</sup>	$\overline{EOI} \downarrow$ to TR1 $\uparrow$	135	nsec	PPSS, ATN=0.45V
TEOD16	$\overline{EOI} \downarrow$ to $\overline{DIO}$ Valid	155	nsec	PPSS, ATN=0.45V
TEOT12	$\overline{EOI} \uparrow$ to TR1 $\downarrow$	155	nsec	PPSS, ATN=0.45V
TATND4	$\overline{ATN} \downarrow$ to $\overline{NDAC} \downarrow$	155	nsec	TACS, AIDS
TATT14	$\overline{ATN} \downarrow$ to TR1 $\downarrow$	155	nsec	TACS, AIDS
TATT24	$\overline{ATN} \downarrow$ to TR2 $\downarrow$	155	nsec	TACS, AIDS
TDVND3-C	$\overline{DAV} \downarrow$ to $\overline{NDAC} \uparrow$	650	nsec	AH, CACS
TNDDV1	$\overline{NDAC} \uparrow$ to $\overline{DAV} \uparrow$	350	nsec	SH, STRS
TNRDR1	$\overline{NRFD} \uparrow$ to DREQ $\uparrow$	400	nsec	SH
TDVDR3	$\overline{DAV} \downarrow$ to DREQ $\uparrow$	600	nsec	AH, LACS, ATN=2.4V
TDVND2-C	$\overline{DAV} \uparrow$ to $\overline{NDAC} \downarrow$	350	nsec	AH, LACS
TDVNR1-C	$\overline{DAV} \uparrow$ to $\overline{NRFD} \uparrow$	350	nsec	AH, LACS, rdy=True
TRDNR3	$\overline{RD} \downarrow$ to $\overline{NRFD} \uparrow$	500	nsec	AH, LACS
TWRD15	$\overline{WR} \uparrow$ to $\overline{DIO}$ Valid	280	nsec	SH, TACS, RS=0.4V
TWREO5	$\overline{WR} \uparrow$ to $\overline{EOI}$ Valid	350	nsec	SH, TACS
TWRDV2	$\overline{WR} \uparrow$ to $\overline{DAV} \downarrow$	$830 + t_{\text{SYNC}}$	nsec	High Speed Transfers Enabled, $N_F = f_C, t_{\text{SYNC}} = 1/2 \cdot f_C$

**NOTES:**

- All GPIO timings are at the pins of the 8291A.
- The last number in the symbol for any GPIO timing parameter is chosen according to the transition directions of the reference signals. The following table describes the numbering scheme.

$\uparrow$ to $\uparrow$	1
$\uparrow$ to $\downarrow$	2
$\downarrow$ to $\uparrow$	3
$\downarrow$ to $\downarrow$	4
$\uparrow$ to VALID	5
$\downarrow$ to VALID	6

## APPENDIX A

### MODIFIED STATE DIAGRAMS

Figure A-1 presents the interface function state diagrams. It is derived from IEEE Std. state diagrams, with the following changes:

A. The 8291A supports the complete set of IEEE-488 interface functions except for the controller. These include: SH1, AH1, T5, TE5, L3, LE3, SR1, RL1, PP1, DC1, DT1, and C0.

B. Addressing modes included in T,L state diagrams.

Note that in Mode 3, MSA, OSA are generated only after secondary address validity check by the microprocessor (APT interrupt).

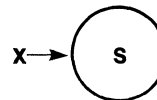
C. In these modified state diagrams, the IEEE-488-1978 convention of negative (low true) logic is followed. This should not be confused with the Intel pin- and signal-naming convention based on positive logic. Thus, while the state diagrams below carry low true logic, the signals described elsewhere in this data sheet are consistent with Intel notation and are based on positive logic.

Level	Logic	Convention	
		IEEE-488	Intel
0	T	DAV	$\overline{\text{DAV}}$
1	F	DAV	DAV
0	T	NDAC	$\overline{\text{NDAC}}$
1	F	NDAC	NDAC
0	T	NRFD	$\overline{\text{NRFD}}$
1	F	$\overline{\text{NRFD}}$	NRFD

Consider the condition when the Not-Ready-For-Data signal (pin 37) is active. Intel indicates this active low signal with the symbol  $\overline{\text{NRFD}}$  ( $V_{\text{OUT}} \leq V_{\text{OL}}$  for AH;  $V_{\text{IN}} \leq V_{\text{IL}}$  for SH). The IEEE-488-1978 Standard, in its state diagrams, indicates the active state of this signal (True condition) with NRFD.

D. All remote multiline messages decoded are conditioned by ACDS. The multiplication by ACDS is not drawn to simplify the diagrams.

E. The symbol



indicates:

1. When event X occurs, the function returns to state S.
2. X overrides any other transition condition in the function.

Statement 2 simplifies the diagram, avoiding the explicit use of X to condition all transitions from S to other states.

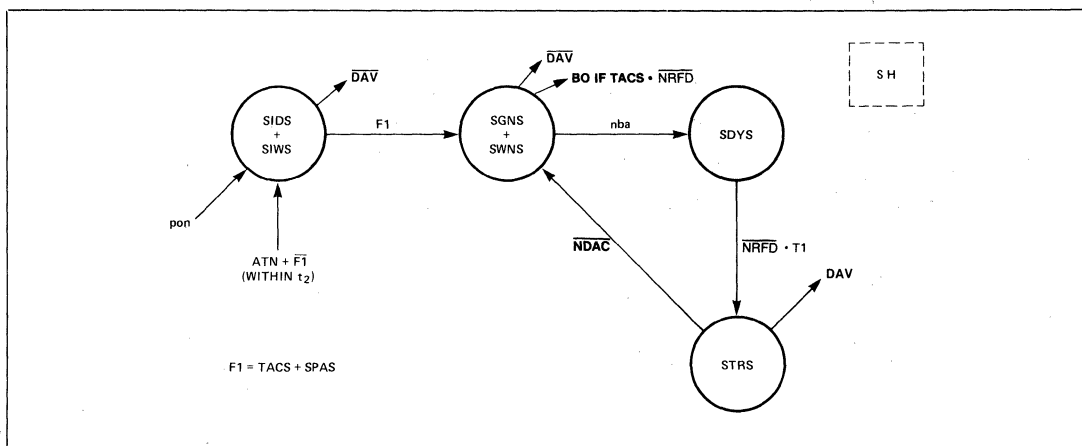


Figure A-1. 8291A State Diagrams (Continued next page)

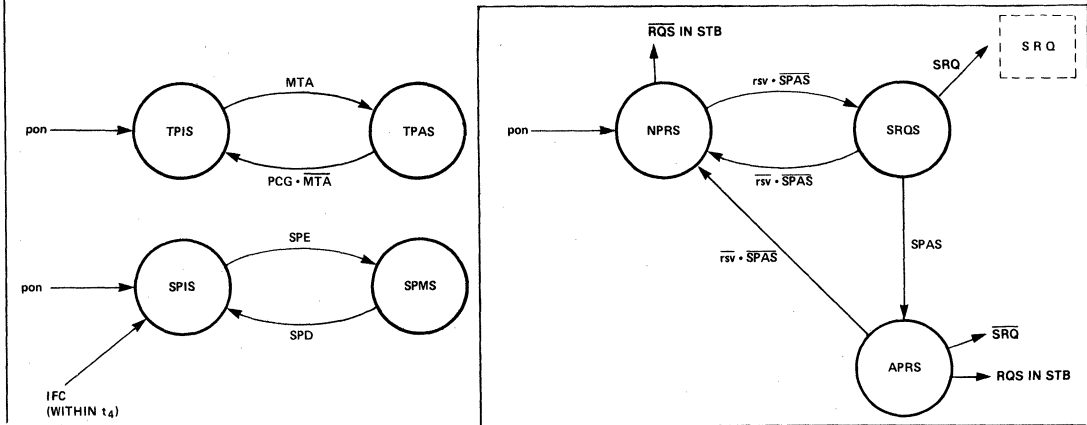
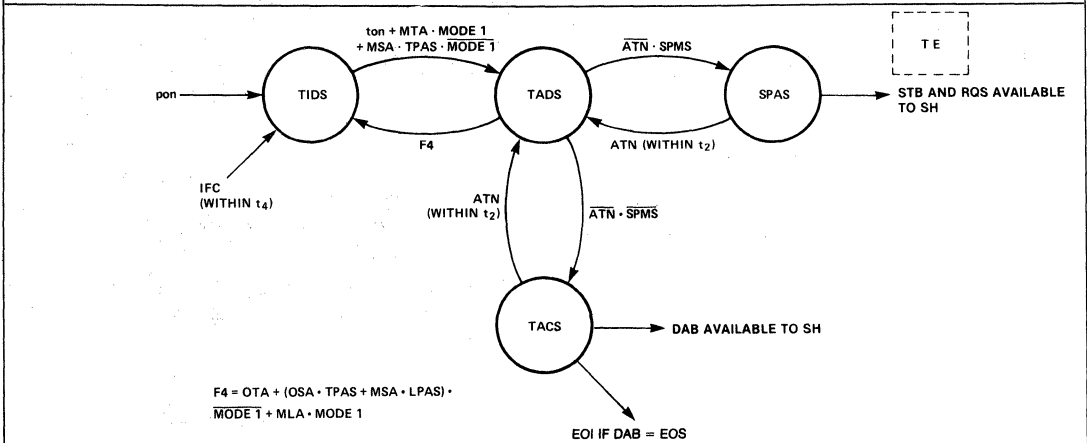
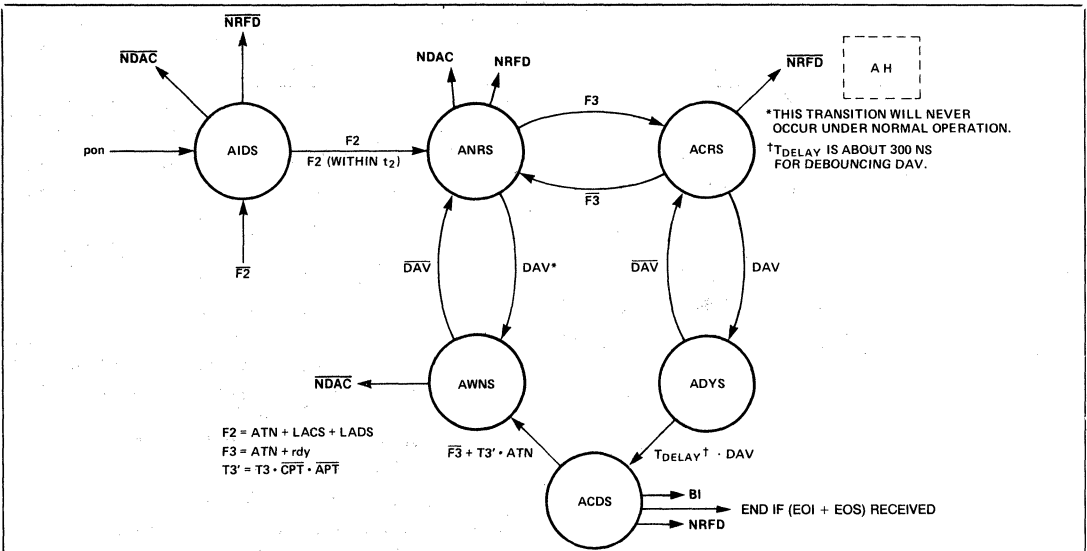


Figure A-1. 8291A State Diagrams (Continued next page)

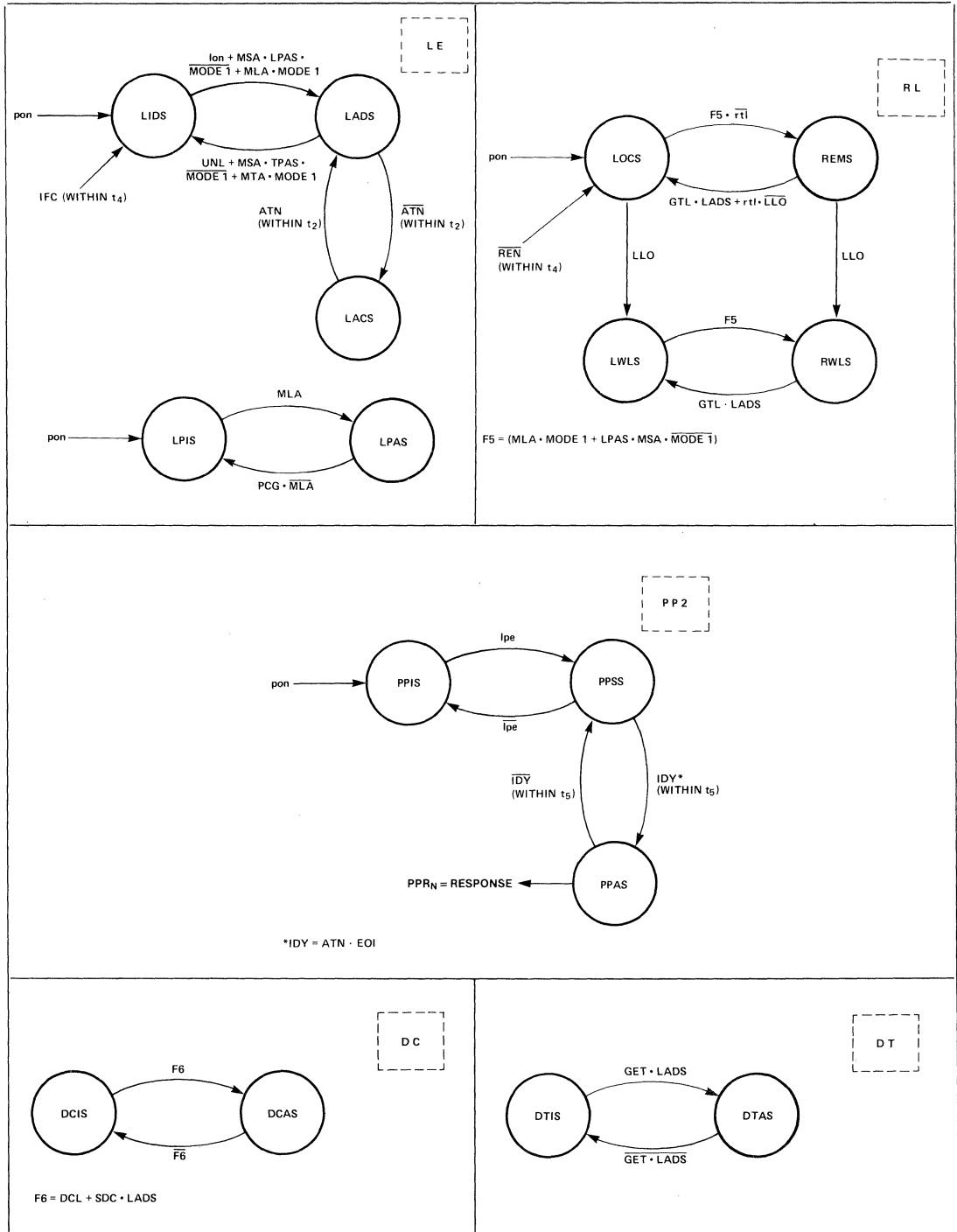


Figure A-1. 8291A State Diagrams

## APPENDIX B

Table B-1. IEEE 488 Time Values

Time Value Identifier <sup>1</sup>	Function (Applies to)	Description	Value
T <sub>1</sub>	SH	Settling Time for Multiline Messages	$\geq 2\mu s^2$
t <sub>2</sub>	LC, $\overline{IC}$ , SH, AH, T, L	Response to ATN	$\leq 200ns$
T <sub>3</sub>	AH	Interface Message Accept Time <sup>3</sup>	$> 0^4$
t <sub>4</sub>	T, TE, L, LE, C, CE	Response to IFC or REN False	$< 100\mu s$
t <sub>5</sub>	PP	Response to ATN+EOI	$\leq 200ns$
T <sub>6</sub>	C	Parallel Poll Execution Time	$\geq 2\mu s$
T <sub>7</sub>	C	Controller Delay to Allow Current Talker to see ATN Message	$\geq 500 ns$
T <sub>8</sub>	C	Length of IFC or REN False	$> 100\mu s$
T <sub>9</sub>	C	Delay for EOI <sup>5</sup>	$\geq 1.5\mu s^6$

## NOTES:

<sup>1</sup>Time values specified by a lower case t indicate the maximum time allowed to make a state transition. Time values specified by an upper case T indicate the minimum time that a function must remain in a state before exiting.

<sup>2</sup>If three-state drivers are used on the  $\overline{DIO}$ ,  $\overline{DAV}$ , and  $\overline{EOI}$  lines, T<sub>1</sub> may be:

1.  $\geq 1100 ns$ .
2. Or  $\geq 700 ns$  if it is known that within the controller ATN is driven by a three-state driver.
3. Or  $\geq 500ns$  for all subsequent bytes following the first sent after each false transition of ATN (the first byte must be sent in accordance with (1) or (2)).
4. Or  $\geq 350ns$  for all subsequent bytes following the first sent after each false transition of ATN under conditions specified in Section 5.2.3 and warning note. See IEEE Standard 488.

<sup>3</sup>Time required for interface functions to accept, not necessarily respond to interface messages.

<sup>4</sup>Implementation dependent.

<sup>5</sup>Delay required for  $\overline{EOI}$ ,  $\overline{NDAC}$ , and  $\overline{NRFD}$  signal lines to indicate valid states.

<sup>6</sup> $\geq 600 ns$  for three-state drivers.

### APPENDIX C THE THREE-WIRE HANDSHAKE

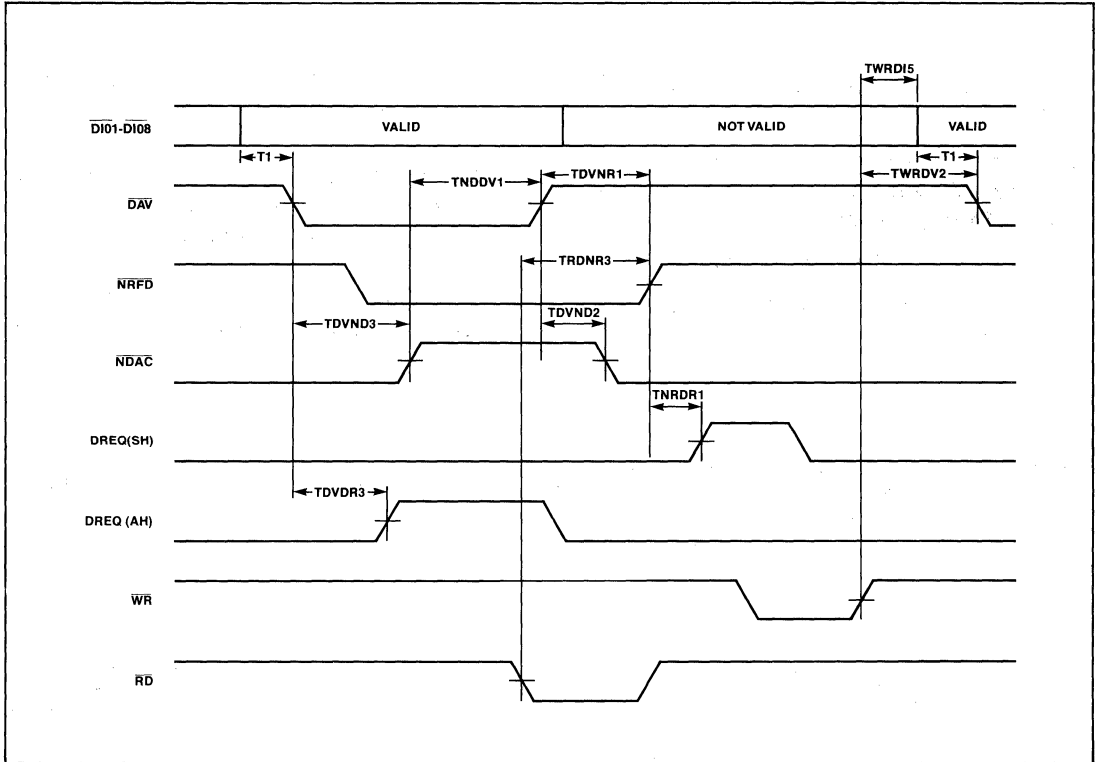


Figure C-1. 3-Wire Handshake Timing at 8291A





# 8292 GPIB CONTROLLER

- Complete IEEE Standard 488 Controller Function
  - Interface Clear (IFC) Sending Capability Allows Seizure of Bus Control and/or Initialization of the Bus
  - Responds to Service Requests (SRQ)
  - Sends Remote Enable (REN), Allowing Instruments to Switch to Remote Control
- Complete Implementation of Transfer Control Protocol
  - Synchronous Control Seizure Prevents the Destruction of Any Data Transmission in Progress
  - Connects with the 8291 to Form a Complete IEEE Standard 488 Interface Talker/Listener/Controller

The 8292 GPIB Controller is a microprocessor-controlled chip designed to function with the 8291 GPIB Talker/Listener to implement the full IEEE Standard 488 controller function, including transfer control protocol. The 8292 is a pre-programmed Intel® 8041A.

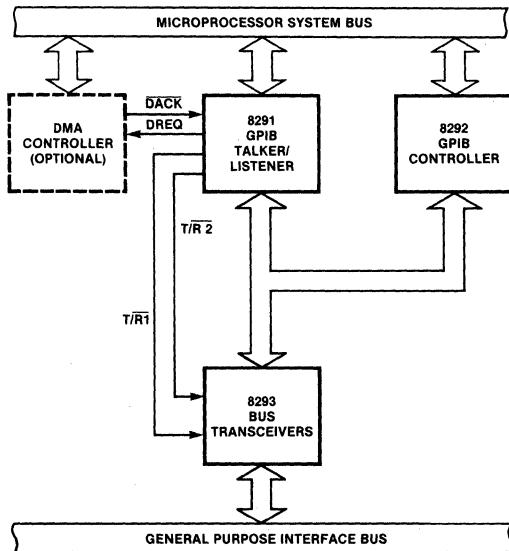


Figure 1. 8291, 8292 Block Diagram

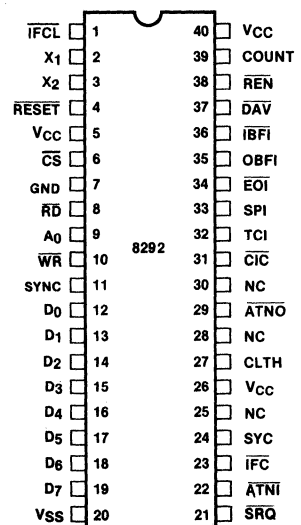


Figure 2. Pin Configuration

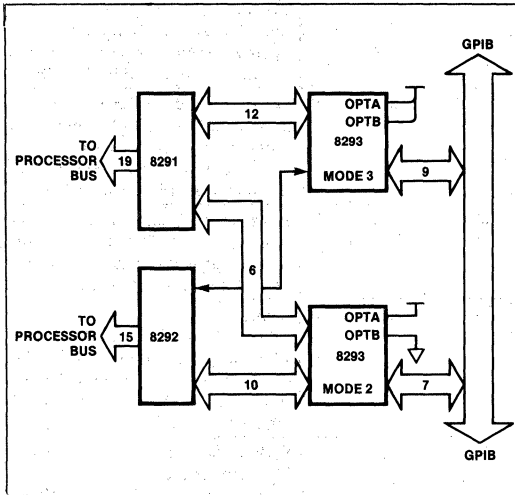
Table 1. Pin Description

Symbol	Pin No.	Type	Name and Function
$\overline{\text{IFCL}}$	1	I	<b>IFC Received (Latched):</b> The 8292 monitors the IFC Line (when not system controller) through this pin.
$X_1, X_2$	2, 3	I	<b>Crystal Inputs:</b> Inputs for a crystal, LC or an external timing signal to determine the internal oscillator frequency.
RESET	4	I	<b>Reset:</b> Used to initialize the chip to a known state during power on.
CS	6	I	<b>Chip Select Input:</b> Used to select the 8292 from other devices on the common data bus.
$\overline{\text{RD}}$	8	I	<b>Read Enable:</b> Allows the master CPU to read from the 8292.
$A_0$	9	I	<b>Address Line:</b> Used to select between the data bus and the status register during read operations and to distinguish between data and commands written into the 8292 during write operations.
$\overline{\text{WR}}$	10	I	<b>Write Enable:</b> Allows the master CPU to write to the 8292.
SYNC	11	O	<b>Sync:</b> 8041A instruction cycle synchronization signal; it is an output clock with a frequency of XTAL $\div$ 15.
$D_0\text{--}D_7$	12-19	I/O	<b>Data:</b> 8 bidirectional lines used for communication between the central processor and the 8292's data bus buffers and status register.
$V_{SS}$	7, 20	P.S.	<b>Ground:</b> Circuit ground potential.
$\overline{\text{SRQ}}$	21	I	<b>Service Request:</b> One of the IEEE control lines. Sampled by the 8292 when it is controller in charge. If true, SPI interrupt to the master will be generated.
$\overline{\text{ATNI}}$	22	I	<b>Attention In:</b> Used by the 8292 to monitor the GPIB ATN control line. It is used during the transfer control procedure.
$\overline{\text{IFC}}$	23	I/O	<b>Interface Clear:</b> One of the GPIB management lines, as defined by IEEE Std. 488-1978, places all devices in a known quiescent state.
SYC	24	I	<b>System Controller:</b> Monitors the system controller switch.
CLTH	27	O	<b>Clear Latch:</b> Used to clear the $\overline{\text{IFCR}}$ latch after being recognized by the 8292. Usually low (except after hardware Reset), it will be pulsed high when $\overline{\text{IFCR}}$ is recognized by the 8292.
ATNO	29	O	<b>Attention Out:</b> Controls the ATN control line of the bus through external logic for tcs and tca procedures. (ATN is a GPIB control line, as defined by IEEE Std. 488-1978.)

Symbol	Pin No.	Type	Name and Function
$V_{CC}$	5, 26, 40	P.S.	<b>Voltage:</b> +5V supply input $\pm$ 10%.
COUNT	39	I	<b>Event Count:</b> When enabled by the proper command the internal counter will count external events through this pin. High to low transition will increment the internal counter by one. The pin is sampled once per three internal instruction cycles (7.5 $\mu$ sec sample period when using 5 MHz XTAL). It can be used for byte counting when connected to NDAC, or for block counting when connected to the EOI.
$\overline{\text{REN}}$	38	O	<b>Remote Enable:</b> The Remote Enable bus signal selects remote or local control of the device on the bus. A GPIB bus management line, as defined by IEEE Std. 488-1978.
$\overline{\text{DAV}}$	37	I/O	<b>Data Valid:</b> Used during parallel poll to force the 8291 to accept the parallel poll status bits. It is also used during the tcs procedure.
IBFI	36	O	<b>Input Buffer Not Full:</b> Used to interrupt the central processor while the input buffer of the 8292 is empty. This feature is enabled and disabled by the interrupt mask register.
OBFI	35	O	<b>Output Buffer Full:</b> Used as an interrupt to the central processor while the output buffer of the 8292 is full. The feature can be enabled and disabled by the interrupt mask register.
$\overline{\text{EOI2}}$	34	I/O	<b>End Or Identify:</b> One of the GPIB management lines, as defined by IEEE Std. 488-1978. Used with ATN as Identify Message during parallel poll.
SPI	33	O	<b>Special Interrupt:</b> Used as an interrupt on events not initiated by the central processor.
TCI	32	O	<b>Task Complete Interrupt:</b> Interrupt to the control processor used to indicate that the task requested was completed by the 8292 and the information requested is ready in the data bus buffer.
CIC	31	O	<b>Controller In Charge:</b> Controls the S/R input of the SRQ bus transceiver. It can also be used to indicate that the 8292 is in charge of the GPIB bus.

**FUNCTIONAL DESCRIPTION**

The 8292 is an Intel 8041A which has been programmed as a GPIB Controller interface element. It is used with the 8291 GPIB Talker/Listener and two 8293 GPIB Transceivers to form a complete IEEE-488 Bus Interface for a microprocessor. The electrical interface is performed by the transceivers, data transfer is done by the talker/listener, and control of the bus is done by the 8292. Figure 3 is a typical controller interface using Intel's GPIB peripherals.



**Figure 3. Talker/Listener/Controller Configuration**

The internal RAM in the 8041A is used as a special purpose register bank for the 8292. Most of these registers (except for the interrupt flag) can be accessed through commands to the 8292. Table 2 identifies the registers used by the 8292 and how they are accessed.

**Interrupt Status Register**

SYC	ERR	SRQ	EV	X	IFCR	IBF	OBF
D <sub>7</sub>				D <sub>0</sub>			

The 8292 can be configured to interrupt the microprocessor on one of several conditions. Upon receipt of the interrupt the microprocessor must read the 8292 interrupt status register to determine which event caused the interrupt, and then the appropriate subroutine can be performed. The interrupt status register is read with A<sub>0</sub> high. With the exception of OBF and IBF, these interrupts are enabled or disabled by the SPI interrupt mask. OBF and IBF have their own bits in the interrupt mask (OBF<sub>I</sub> and IBF<sub>I</sub>).

**OBF** Output Buffer Full. A byte is waiting to be read by the microprocessor. This flag is cleared when the output data bus buffer is read.

**IBF** Input Buffer Full. The byte previously written by the microprocessor has not been read yet by the 8292. If another byte is written to the 8292 before this flag clears, data will be lost. IBF is cleared when the 8292 reads the data byte.

**IFCR** Interface Clear Received. The GPIB system controller has set IFC. The 8292 has become idle and is no longer in charge of the bus. The flag is cleared when the IACK command is issued.

**EV** Event Counter Interrupt. The requested number of blocks or data bytes has been transferred. The EV interrupt flag is cleared by the IACK command.

**SRQ** Service Request. Notifies the 8292 that a service request (SRQ) message has been received. It is cleared by the IACK command.

**ERR** Error occurred. The type of error can be determined by reading the error status register. This interrupt flag is cleared by the IACK command.

**SYC** System Controller Switch Change. Notifies the processor that the state of the system controller switch has changed. The actual state is contained in the GPIB Status Register. This flag is cleared by the IACK command.

**Table 2. 8292 Registers**

READ FROM 8292								WRITE TO 8292							
INTERRUPT STATUS								INTERRUPT MASK							
SYC	ERR	SRQ	EV	X	IFCR	IBF	OBF	1	SPI	TCI	SYC	OBF <sub>I</sub>	IBF <sub>I</sub>	0	SRQ
D <sub>7</sub>				D <sub>0</sub>				D <sub>7</sub>				D <sub>0</sub>			
ERROR FLAG								ERROR MASK							
X	X	USER	X	X	TOUT <sub>3</sub>	TOUT <sub>2</sub>	TOUT <sub>1</sub>	0	0	USER	0	0	TOUT <sub>3</sub>	TOUT <sub>2</sub>	TOUT <sub>1</sub>
CONTROLLER STATUS								COMMAND FIELD							
CSBS	CA	X	X	SYCS	IFC	REN	SRQ	1	1	1	OP	C	C	C	C
GPIB (BUS) STATUS								EVENT COUNTER							
REN	DAV	EOI	X	SYC	IFC	ANTI	SRQ	D	D	D	D	D	D	D	D
EVENT COUNTER STATUS								TIME OUT							
D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D
TIME OUT STATUS															
D	D	D	D	D	D	D	D								

Note: These registers are accessed by a special utility command, see page 6.

**Interrupt Mask Register**

1	SPI	TCl	SYC	OBFi	IBFi	0	SRQ
D <sub>7</sub>				D <sub>0</sub>			

The Interrupt Mask Register is used to enable features and to mask the SPI and TCI interrupts. The flags in the Interrupt Status Register will be active even when masked out. The Interrupt Mask Register is written when A<sub>0</sub> is low and reset by the RINM command. When the register is read, D<sub>1</sub> and D<sub>7</sub> are undefined. An interrupt is enabled by setting the corresponding register bit.

- SRQ** Enable interrupts on SRQ received.
- IBFi** Enable interrupts on input buffer empty.
- OBFi** Enable interrupts on output buffer full.
- SYC** Enable interrupts on a change in the system controller switch.
- TCl** Enable interrupts on the task completed.
- SPI** Enable interrupts on special events.

NOTE: The event counter is enabled by the GSEC command, the error interrupt is enabled by the error mask register, and IFC cannot be masked (it will always cause an interrupt).

**Controller Status Register**

CSBS	CA	X	X	SYCS	IFC	REN	SRQ
D <sub>7</sub>				D <sub>0</sub>			

The Controller Status Register is used to determine the status of the controller function. This register is accessed by the RCST command.

- SRQ** Service Request line active (CSRS).
- REN** Sending Remote Enable.
- IFC** Sending or receiving interface clear.
- SYCS** System Controller Switch Status (SACS).
- CA** Controller Active (CACS + CAWS + CSWS).
- CSBS** Controller Stand-by State (CSBS, CA) = (0,0) — Controller Idle

**GPiB Bus Status Register**

REN	DAV	EOI	X	SYC	IFC	ATNI	SRQ
D <sub>7</sub>				D <sub>0</sub>			

This register contains GPiB bus status information. It can be used by the microprocessor to monitor and manage the bus. The GPiB Bus Register can be read using the RBST command.

Each of these status bits reflect the current status of the corresponding pin on the 8292.

- SRQ** Service Request
- ATNI** Attention In
- IFC** Interface Clear
- SYC** System Controller Switch
- EOI** End or Identify
- DAV** Data Valid
- REN** Remote Enable

**Event Counter Register**

D <sub>7</sub>	D <sub>6</sub>	D <sub>5</sub>	D <sub>4</sub>	D <sub>3</sub>	D <sub>2</sub>	D <sub>1</sub>	D <sub>0</sub>
----------------	----------------	----------------	----------------	----------------	----------------	----------------	----------------

The Event Counter Register contains the initial value for the event counter. The counter can count pulses on pin 39 of the 8292 (COUNT). It can be connected to EOI or NDAC to count blocks or bytes respectively during standby state. A count of zero equals 256. This register cannot be read, and is written using the WEVC command.

**Event Counter Status Register**

D <sub>7</sub>	D <sub>6</sub>	D <sub>5</sub>	D <sub>4</sub>	D <sub>3</sub>	D <sub>2</sub>	D <sub>1</sub>	D <sub>0</sub>
----------------	----------------	----------------	----------------	----------------	----------------	----------------	----------------

This register contains the current value in the event counter. The event counter counts back from the initial value stored in the Event Counter Register to zero and then generates an Event Counter Interrupt. This register cannot be written and can be read using a REVC command.

**Time Out Register**

D <sub>7</sub>	D <sub>6</sub>	D <sub>5</sub>	D <sub>4</sub>	D <sub>3</sub>	D <sub>2</sub>	D <sub>1</sub>	D <sub>0</sub>
----------------	----------------	----------------	----------------	----------------	----------------	----------------	----------------

The Time Out Register is used to store the time used for the time out error function. See the individual timeouts (TOUT1, 2, 3) to determine the units of this counter. This Time Out Register cannot be read, and it is written with the WTOUT command.

**Time Out Status Register**

D <sub>7</sub>	D <sub>6</sub>	D <sub>5</sub>	D <sub>4</sub>	D <sub>3</sub>	D <sub>2</sub>	D <sub>1</sub>	D <sub>0</sub>
----------------	----------------	----------------	----------------	----------------	----------------	----------------	----------------

This register contains the current value in the time out counter. The time out counter decrements from the original value stored in the Time Out Register. When zero is reached, the appropriate error interrupt is generated. If the register is read while none of the time out functions are active, the register will contain the last value reached the last time a function was active. The Time Out Status Register cannot be written, and it is read with the RTOUT command.

**Error Flag Register**

X	X	USER	X	X	TOUT <sub>3</sub>	TOUT <sub>2</sub>	TOUT <sub>1</sub>
D <sub>7</sub>				D <sub>0</sub>			

Four errors are flagged by the 8292 with a bit in the Error Flag Register. Each of these errors can be masked by the Error Mask Register. The Error Flag Register cannot be written, and it is read by the IACK command when the error flag in the Interrupt Status Register is set.

**TOUT1** Time Out Error 1 occurs when the current controller has not stopped sending ATN after receiving the TCT message for the time period specified by the Time Out Register. Each count in the Time Out Register is at least 1800 tcy. After flagging the error, the 8292 will remain in a loop trying to take control until the current controller stops sending ATN or a new command is written by the microprocessor. If a new command is written, the 8292 will return to the loop after executing it.

**TOUT2** Time Out Error 2 occurs when the transmission between the addressed talker and listener has not started for the time period specified by the Time Out Register. Each count in the Time Out Register is at least 45 t<sub>cy</sub>. This feature is only enabled when the controller is in the CSBS state.

**TOUT3** Time Out Error 3 occurs when the handshake signals are stuck and the 8292 is not succeeding in taking control synchronously for the time period specified by the Time Out Register. Each count in the Time Out Register is at least 1800 t<sub>cy</sub>. The 8292 will continue checking  $\overline{ATN1}$  until it becomes true or a new command is received. After performing the new command, the 8292 will return to the  $\overline{ATN1}$  checking loop.

**USER** User error occurs when request to assert IFC or REN was received and the 8292 was not the system controller.

**Error Mask Register**

0	0	USER	0	0	TOUT <sub>3</sub>	TOUT <sub>2</sub>	TOUT <sub>1</sub>
D <sub>7</sub>				D <sub>0</sub>			

The Error Mask Register is used to mask the interrupt from a particular type of error. Each type of error interrupt is enabled by setting the corresponding bit in the Error Mask Register. This register can be read with the RERM command and written with A<sub>0</sub> low.

**Command Register**

1	1	1	OP	C	C	C	C
D <sub>7</sub>				D <sub>0</sub>			

Commands are performed by the 8292 whenever a byte is written with A<sub>0</sub> high. There are two categories of commands distinguished by the OP bit (bit 4). The first category is the operation command (OP=1). These commands initiate some action on the interface bus. The second category is the utility commands (OP=0). These commands are used to aid the communication between the processor and the 8292.

**OPERATION COMMANDS**

Operation commands initiate some action on the GPIB interface bus. It is using these commands that the control functions such as polling, taking and passing control, and system controller functions are performed. A TCI interrupt is generated upon successful completion of each of these functions.

**F0 — SPCNI — Stop Counter Interrupts**

This command disables the internal counter interrupt so that the 8292 will stop interrupting the master on event counter underflows. However, the counter will continue counting and its contents can still be used.

**F1 — GIDL — Go To Idle**

This command is used during the transfer of control procedure while transferring control to another controller. The 8292 will respond to this command only if it is in the active state.  $\overline{ATN0}$  will go high, and  $\overline{CIC}$  will be high so that this 8292 will no longer be driving the ATN line on the GPIB interface bus.

**F2 — RST — Reset**

This command has the same effect as asserting the external reset on the 8292. For details, refer to the reset procedure described later.

**F3 — RSTI — Reset Interrupts**

This command resets any pending interrupts and clears the error flags. The 8292 will not return to any loop it was in (such as from the time out interrupts).

**F4 — GSEC — Go To Standby, Enable Counting**

The function causes  $\overline{ATN0}$  to go high and the counter will be enabled. If the 8292 was not the active controller, this command will exit immediately. If the 8292 is the active controller, the counter will be loaded with the value stored in the Event Counter Register, and the internal interrupt will be enabled so that when the counter reaches zero, the SPI interrupt will be generated. SPI will be generated every 256 counts thereafter until the controller exits the standby state or the SPCNI command is written. An initial count of 256 (zero in the Event Counter Register) will be used if the WEVC command is not executed. If the data transmission does not start, a TOUT2 error will be generated.

**F5 — EXPP — Execute Parallel Poll**

This command initiates a parallel poll by asserting ATN and EOI (IDY message) true. The 8291 should be previously configured as a listener. Upon detection of DAV true, the 8291 enters ACDS and latches the parallel poll response (PPR) byte into its data in register. The master will be interrupted by the 8291 BI interrupt when the PPR byte is available. No interrupts except the  $\overline{IBF1}$  will be generated by the 8292. The 8292 will respond to this command only when it is the active controller.

**F6 — GTSB — Go To Standby**

If the 8292 is the active controller,  $\overline{ATN0}$  will go high then TCI will be generated. If the data transmission does not start, a TOUT2 error will be generated.

**F7 — SLOC — Set Local Mode**

If the 8292 is the system controller, then REN will be asserted false for at least 100  $\mu$ sec. If it is not the system controller, the User Error bit will be set in the Error Flag Register.

**F8 — SREM — Set Interface To Remote Control**

This command will set REN true if this 8292 is the system controller. If not, the User Error bit will be set in the Error Flag Register.

**F9 — ABORT — Abort All Operation, Clear Interface**

This command will cause IFC to be asserted true for at least 100  $\mu$ sec if this 8292 is the system controller. If it is in CIDS, it will take control over the bus (see the TCNTR command).

**FA — TCNTR — Take Control**

The transfer of control procedure is coordinated by the master with the 8291 and 8292. When the master receives a TCT message from the 8291, it should issue the TCNTR command to the 8292. The following events occur to take control:

1. The 8292 checks to see if it is in CIDS, and if not, it exits.
2. Then  $\overline{ATNI}$  is checked until it becomes high. If the current controller does not release ATN for the time specified by the Time Out Register, then a TOUT1 error is generated. The 8292 will return to this loop after an error or any command except the RST and RSTI commands.
3. After the current controller releases ATN, the 8292 will assert  $\overline{ATNO}$  and  $\overline{CIC}$  low.
4. Finally, the TCI interrupt is generated to inform the master that it is in control of the bus.

**FC — TCASY — Take Control Asynchronously**

TCAS transfers the 8292 from CSBS to CACS independent of the handshake lines. If a bus hangup is detected (by an error flag), this command will force the 8292 to take control (asserting ATN) even if the AH function is not in ANRS (Acceptor Not Ready State). This command should be used very carefully since it may cause the loss of a data byte. Normally, control should be taken synchronously. After checking the controller function for being in the CSBS (else it will exit immediately),  $\overline{ATNO}$  will go low, and a TCI interrupt will be generated.

**FD — TCSY — Take Control Synchronously**

There are two different procedures used to transfer the 8292 from CSBS to CACS depending on the state of the 8291 in the system. If the 8291 is in "continuous AH cycling" mode (Aux. Reg. A0=A1=1), then the following procedure should be followed:

1. The master microprocessor stops the continuous AH cycling mode in the 8291;
2. The master reads the 8291 Interrupt Status 1 Register;
3. If the END bit is set, the master sends the TCSY command to the 8292;
4. If the END bit was not set, the master reads the 8291 Data In Register and then waits for another BI interrupt from the 8291. When it occurs, the master sends the 8292 the TCSY command.

If the 8291 is not in AH cycling mode, then the master just waits for a BI interrupt and then sends the TCSY command. After the TCSY command has been issued, the 8292 checks for CSBS. If  $\overline{CSBS}$ , then it exits the routine. Otherwise, it then checks the DAV bit in the GPIB status. When DAV becomes false, the 8292 will

wait for at least 1.5  $\mu$ sec. (T10) and then  $\overline{ATNO}$  will go low. If DAV does not go low, a TOUT3 error will be generated.

**FE — STCNI — Start Counter Interrupts**

This command enables the internal counter interrupt. The counter is enabled by the GSEC command.

**UTILITY COMMANDS**

All these commands are either Read or Write to registers in the 8292. Upon completion of Read commands, the TCI (Task Completed Interrupt) will be generated. Note that writing to the Error Mask Register and the Interrupt Mask Register are done directly.

**E1 — WTOUT — Write To Time Out Register**

The byte written to the data bus buffer (with A<sub>0</sub>=0) following this command will determine the time used for the time out function. Since this function is implemented in software, this will not be an accurate time measurement. This feature is enable or disable by the Error Mask Register. No interrupts except for the  $\overline{IBFI}$  will be generated upon completion.

**E2 — WEVC — Write To Event Counter**

The byte written to the data bus buffer (with A<sub>0</sub>=0) following this command will be loaded into the Event Counter Register and the Event Counter Status for byte counting or EOI counting. Only  $\overline{IBFI}$  will indicate completion of this command.

**E3 — REVC — Read Event Counter Status**

This command transfers the contents of the Event Counter into the data bus buffer. A TCI is generated when the data is available in the data bus buffer.

**E4 — RERF — Read Error Flag Register**

This command transfers the contents of the Error Flag Register into the data bus buffer. A TCI is generated when the data is available.

**E5 — RINM — Read Interrupt Mask Register**

This command transfers the contents of the Interrupt Mask Register into the data bus buffer. This register is available to the processor so that it does not need to store this information elsewhere. A TCI is generated when the data is available in the data bus buffer.

**E6 — RCST — Read Controller Status Register**

This command transfers the contents of the Controller Status Register into the data bus buffer and a TCI interrupt is generated.

**E7 — RBST — Read GPIB Bus Status Register**

This command transfers the contents of the GPIB Bus Status Register into the data bus buffer, and a TCI interrupt is generated when the data is available.



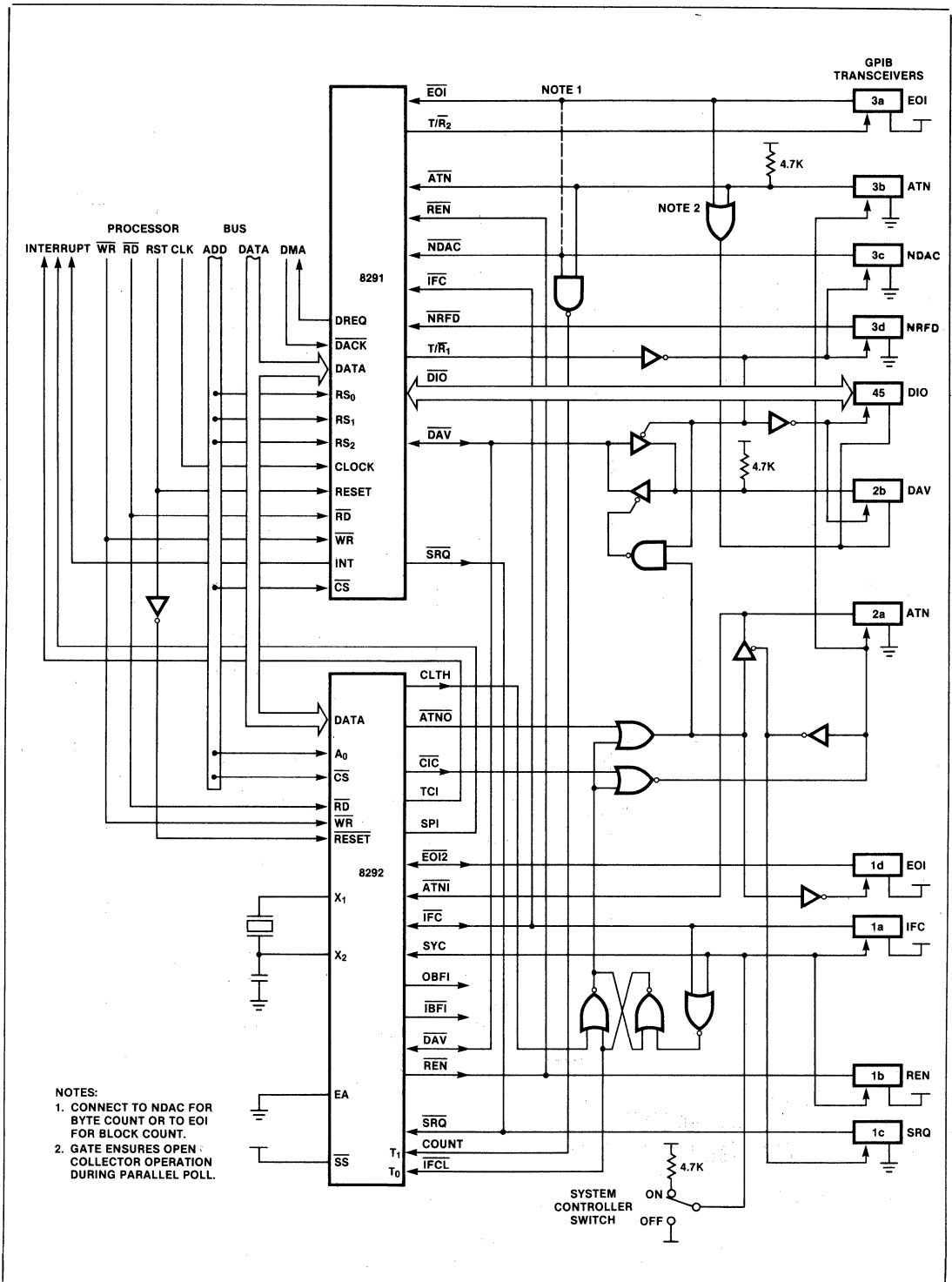


Figure 4. 8291 and 8292 System Configuration



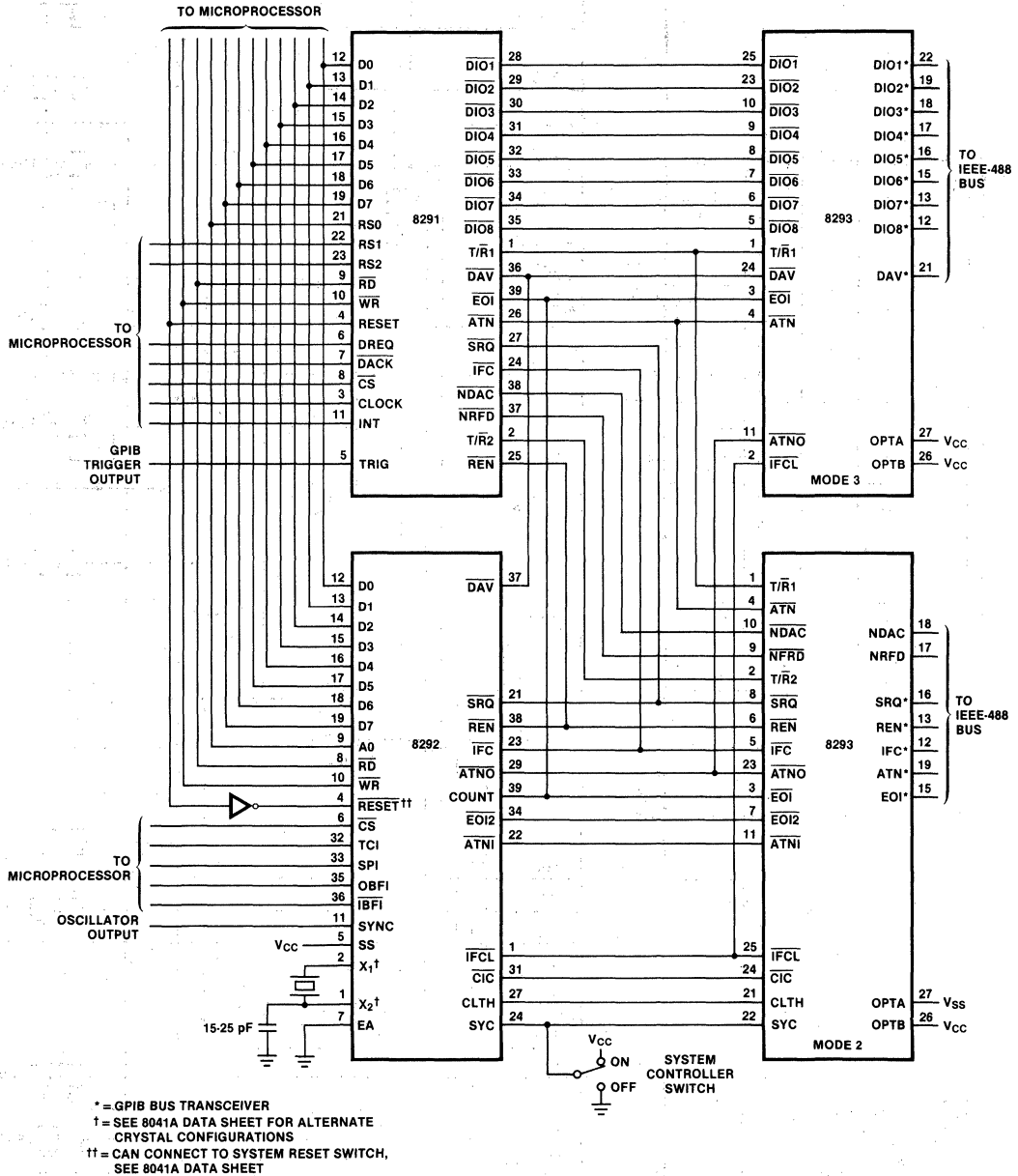


Figure 5. 8291, 8292, and 8293 System Configuration

**ABSOLUTE MAXIMUM RATINGS\***

Ambient Temperature Under Bias . . . . . 0°C to 70°C  
 Storage Temperature . . . . . -65°C to +150°C  
 Voltage on Any Pin With Respect  
 to Ground . . . . . 0.5V to +7V  
 Power Dissipation . . . . . 1.5 Watt

*\*NOTICE: Stresses above those listed under "Absolute Maximum Ratings" may cause permanent damage to the device. This is a stress rating only and functional operation of the device at these or any other conditions above those indicated in the operational sections of this specification is not implied. Exposure to absolute maximum rating conditions for extended periods may affect device reliability.*

**D.C. CHARACTERISTICS** ( $T_A = 0^\circ\text{C}$  to  $70^\circ\text{C}$ ,  $V_{SS} = 0\text{V}$ ; 8292,  $V_{CC} = \pm 5\text{V} \pm 10\%$ )

Symbol	Parameter	Min.	Max.	Unit	Test Conditions
$V_{IL1}$	Input Low Voltage (All Except $X_1, X_2, \overline{\text{RESET}}$ )	-0.5	0.8	V	
$V_{IL2}$	Input Low Voltage ( $X_1, X_2, \overline{\text{RESET}}$ )	-0.5	0.6	V	
$V_{IH1}$	Input High Voltage (All Except $X_1, X_2, \overline{\text{RESET}}$ )	2.2	$V_{CC}$	V	
$V_{IH2}$	Input High Voltage ( $X_1, X_2, \overline{\text{RESET}}$ )	3.8	$V_{CC}$	V	
$V_{OL1}$	Output Low Voltage ( $D_0$ - $D_7$ )		0.45	V	$I_{OL} = 2.0 \text{ mA}$
$V_{OL2}$	Output Low Voltage (All Other Outputs)		0.45	V	$I_{OL} = 1.6 \text{ mA}$
$V_{OH1}$	Output High Voltage ( $D_0$ - $D_7$ )	2.4		V	$I_{OH} = -400 \mu\text{A}$
$V_{OH2}$	Output High Voltage (All Other Outputs)	2.4		V	$I_{OH} = -50 \mu\text{A}$
$I_{IL}$	Input Leakage Current (COUNT, $\overline{\text{IFCL}}, \overline{\text{RD}}, \overline{\text{WR}}, \overline{\text{CS}}, A_0$ )		$\pm 10$	$\mu\text{A}$	$V_{SS} \leq V_{IN} \leq V_{CC}$
$I_{OZ}$	Output Leakage Current ( $D_0$ - $D_7$ , High Z State)		$\pm 10$	$\mu\text{A}$	$V_{SS} + 0.45 \leq V_{IN} \leq V_{CC}$
$I_{LI1}$	Low Input Load Current (Pins 21-24, 27-38)		0.5	mA	$V_{IL} = 0.8\text{V}$
$I_{LI2}$	Low Input Load Current ( $\overline{\text{RESET}}$ )		0.2	mA	$V_{IL} = 0.8\text{V}$
$I_{CC}$	Total Supply Current		125	mA	Typical = 65 mA
$I_{IH}$	Input High Leakage Current (Pins 21-24, 27-38)		100	$\mu\text{A}$	$V_{IN} = V_{CC}$
$C_{IN}$	Input Capacitance		10	pF	
$C_{I/O}$	I/O Capacitance		20	pF	

**A.C. CHARACTERISTICS** ( $T_A = 0^\circ\text{C}$  to  $70^\circ\text{C}$ ,  $V_{SS} = 0\text{V}$ ; 8292,  $V_{CC} = +5\text{V} \pm 10\%$ )

**DBB READ**

Symbol	Parameter	Min.	Max.	Unit	Test Conditions
$t_{AR}$	$\overline{\text{CS}}, A_0$ Setup to $\overline{\text{RD}}\downarrow$	0		ns	
$t_{RA}$	$\overline{\text{CS}}, A_0$ Hold After $\overline{\text{RD}}\uparrow$	0		ns	
$t_{RR}$	$\overline{\text{RD}}$ Pulse Width	250		ns	
$t_{AD}$	$\overline{\text{CS}}, A_0$ to Data Out Delay		225	ns	$C_L = 150 \text{ pF}$
$t_{RD}$	$\overline{\text{RD}}\downarrow$ to Data Out Delay		225	ns	$C_L = 150 \text{ pF}$
$t_{DF}$	$\overline{\text{RD}}\uparrow$ to Data Float Delay		100	ns	
$t_{CY}$	Cycle Time	2.5	15	$\mu\text{s}$	

**DBB WRITE**

Symbol	Parameter	Min.	Max.	Unit	Test Conditions
$t_{AW}$	$\text{CS}, A_0$ Setup to $\text{WR}\downarrow$	0		ns	
$t_{WA}$	$\text{CS}, A_0$ Hold After $\text{WR}\uparrow$	0		ns	
$t_{WW}$	$\text{WR}$ Pulse Width	250		ns	
$t_{DW}$	Data Setup to $\text{WR}\uparrow$	150		ns	
$t_{WD}$	Data Hold After $\text{WR}\downarrow$	0		ns	

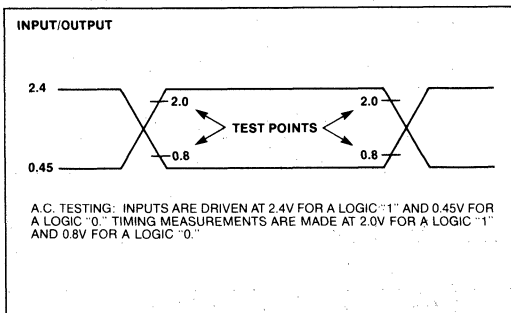
**COMMAND TIMINGS** <sup>[1,3]</sup>

Code	Name	Execution Time	IBFI†	TCI <sup>[2]</sup>	SPI	ATNO	CIC	IFC	REN	EOI	DAV	Comments
E1	WTOUT	63	24									
E2	WEVC	63	24									
E3	REVC	71	24	51								
E4	RERF	67	24	47								
E5	RINM	69	24	49								
E6	RCST	97	24	77								
E7	RBST	92	24	72								
E8												
E9	RTOUT	69	24	49								
EA	RERM	69	24	49								
F0	SPCNI	53	24									Count Stops After 39
F1	GIOL	88	24	70		†61	†61					
F2	RST	94	24		‡52							Not System Controller
F2	RST	214	24	192	‡52	‡179	‡174	‡101				System Controller
F3	RSTI	61	24									
F4	GSEC	125	24	107		†98						
F5	EXPP	75	24						‡53 ‡59	‡55 ‡57		
F6	GTSB	118	24	100		†91						
F7	SLOC	73	24	55				†46				
F8	SREM	91	24	73				‡64				
F9	ABORT	155	24	133		‡120	‡115	‡42				
FA	TCNTR	108	24	86		‡71	‡68					
FC	TCAS	92	24	67		‡55						
FD	TCSY	115	24	91		‡80						
FE	STCNI	59	24									Starts Count After 43
PIN	RESET	29	—	‡7	‡7							Not System Controller
X	IACK	116	—		‡73 ‡98							If Interrupt Pending

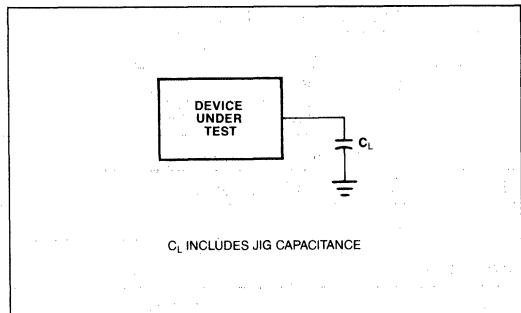
Notes:

1. All times are multiples of  $t_{CY}$  from the 8041A command interrupt.
2. TCI clears after 7  $t_{CY}$  on all commands.
3. † indicates a level transition from low to high, ‡ indicates a high to low transition.

**A.C. TESTING INPUT, OUTPUT WAVEFORM**

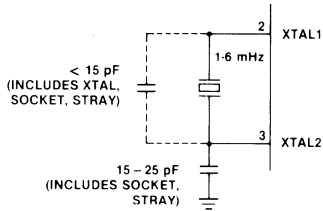


**A.C. TESTING LOAD CIRCUIT**



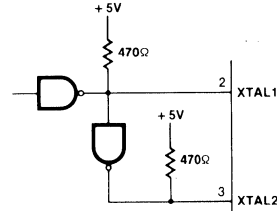
**CLOCK DRIVER CIRCUITS**

**CRYSTAL OSCILLATOR MODE**



CRYSTAL SERIES RESISTANCE SHOULD BE  
 $< 75\Omega$  AT 6 MHz;  $< 180\Omega$  AT 3.6 MHz.

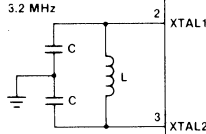
**DRIVING FROM EXTERNAL SOURCE**



BOTH XTAL1 AND XTAL2 SHOULD BE DRIVEN.  
 RESISTORS TO  $V_{CC}$  ARE NEEDED TO ENSURE  $V_{IH} = 3.8V$   
 IF TTL CIRCUITRY IS USED.

**LC OSCILLATOR MODE**

L	C	NOMINAL f
45 $\mu$ H	20 pF	5.2 MHz
120 $\mu$ H	20 pF	3.2 MHz



$$f \approx \frac{1}{2\pi\sqrt{LC^2}}$$

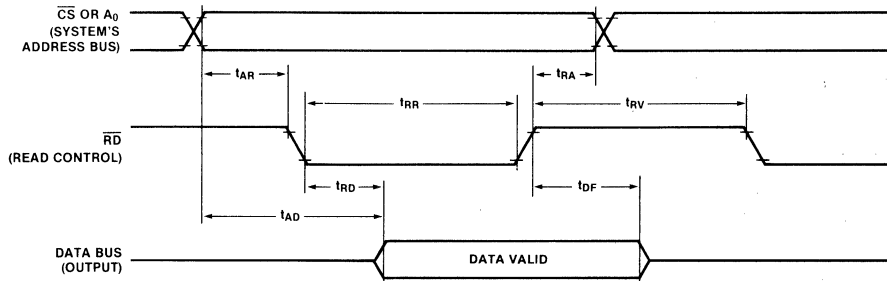
$$C' = \frac{C + 3C_{pp}}{2}$$

$C_{pp} \approx 5 - 10 \text{ pF}$  PIN-TO-PIN  
 CAPACITANCE

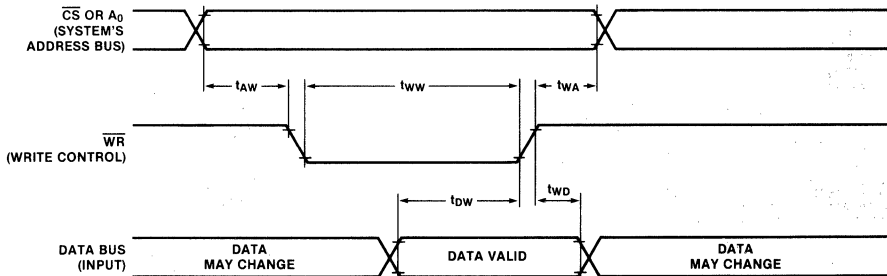
EACH C SHOULD BE APPROXIMATELY 20 pF, INCLUDING STRAY CAPACITANCE.

**WAVEFORMS**

**READ OPERATION—DATA BUS BUFFER REGISTER**



**WRITE OPERATION — DATA BUS BUFFER REGISTER**



APPENDIX

The following tables and state diagrams were taken from the IEEE Standard Digital Interface for Program-

mable Instrumentation, IEEE Std. 488-1978. This document is the official standard for the GPIB bus and can be purchased from IEEE, 345 East 47th St., New York, NY 10017.

C MNEMONICS

Messages	Interface States
pon = power on	CIDS = controller idle state
rsc = request system control	CADS = controller addressed state
rpp = request parallel poll	CTRS = controller transfer state
gts = go to standby	CACS = controller active state
tca = take control asynchronously	CPWS = controller parallel poll wait state
tcs = take control synchronously	CPPS = controller parallel poll state
sic = send interface clear	CSBS = controller standby state
sre = send remote enable	CSHS = controller standby hold state
IFC = interface clear	CAWS = controller active wait state
ATN = attention	CSWS = controller synchronous wait state
TCT = take control	CSRS = controller service requested state
	CSNS = controller service not requested state
	SNAS = system control not active state
	SACS = system control active state
	SRIS = system control remote enable idle state
	SRNS = system control remote enable not active state
	SRAS = system control remote enable active state
	SIIS = system control interface clear idle state
	SINS = system control interface clear not active state
	SIAS = system control interface clear active state
	(ACDS) = accept data state (AH function)
	(ANRS) = acceptor not ready state (AH function)
	(SDYS) = source delay state (SH function)
	(STRS) = source transfer state (SH function)
	(TADS) = talker addressed state (T function)

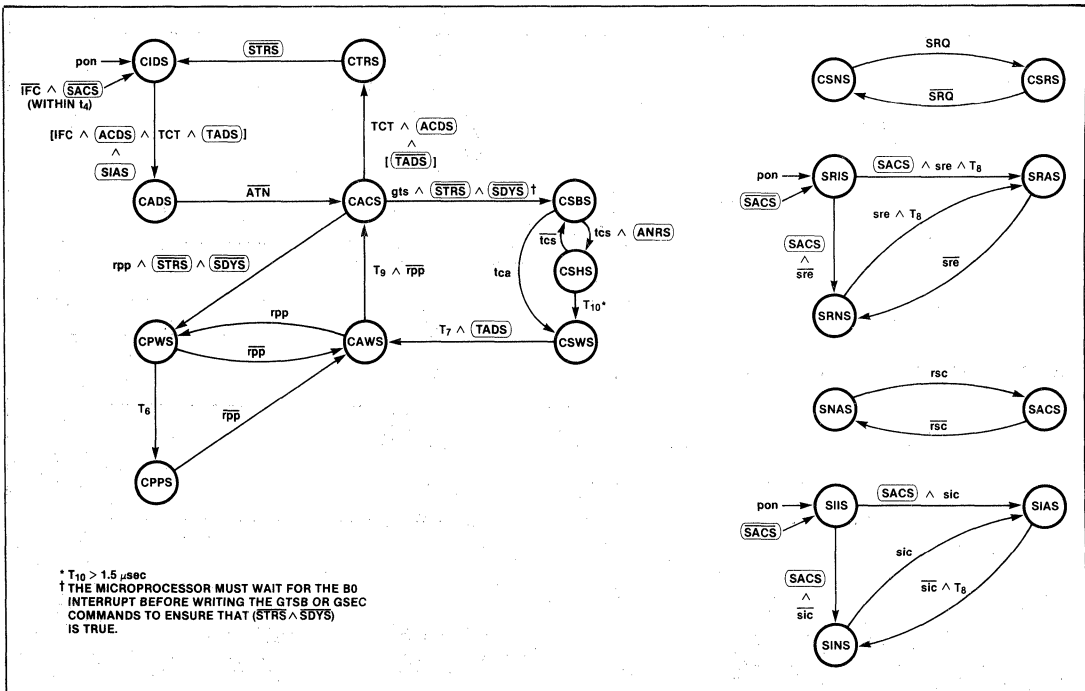


Figure A.1. C State Diagram

REMOTE MESSAGE CODING

Mnemonic	Message Name	T Y P E	C L A S S	Bus Signal Line(s) and Coding That Asserts the True Value of the Message															
				D I O 8	7	6	5	4	3	2	D I O 1	N N D R D A F A V D C	A T O R N I	E S R Q	I F C N	R E N			
ACG	Addressed Command Group	M	AC	Y	0	0	0	X	X	X	X	XXX	1	X	X	X	X		
ATN	Attention	U	UC	X	X	X	X	X	X	X	X	XXX	1	X	X	X	X		
DAB	Data Byte	M	DD	D	D	D	D	D	D	D	D	XXX	0	X	X	X	X		
	(Notes 1, 9)			8	7	6	5	4	3	2	1								
DAC	Data Accepted	U	HS	X	X	X	X	X	X	X	X	XX0	X	X	X	X	X		
DAV	Data Valid	U	HS	X	X	X	X	X	X	X	X	1XX	X	X	X	X	X		
DCL	Device Clear	M	UC	Y	0	0	1	0	1	0	0	XXX	1	X	X	X	X		
END	End	U	ST	X	X	X	X	X	X	X	X	XXX	0	1	X	X	X		
EOS	End of String	M	DD	E	E	E	E	E	E	E	E	XXX	0	X	X	X	X		
	(Notes 2, 9)			8	7	6	5	4	3	2	1								
GET	Group Execute Trigger	M	AC	Y	0	0	0	1	0	0	0	XXX	1	X	X	X	X		
GTL	Go to Local	M	AC	Y	0	0	0	0	0	0	1	XXX	1	X	X	X	X		
IDY	Identify	U	UC	X	X	X	X	X	X	X	X	XXX	X	1	X	X	X		
IFC	Interface Clear	U	UC	X	X	X	X	X	X	X	X	XXX	X	X	X	1	X		
LAG	Listen Address Group	M	AD	Y	0	1	X	X	X	X	X	XXX	1	X	X	X	X		
LLO	Local Lock Out	M	UC	Y	0	0	1	0	0	0	1	XXX	1	X	X	X	X		
MLA	My Listen Address	M	AD	Y	0	1	L	L	L	L	L	XXX	1	X	X	X	X		
	(Note 3)						5	4	3	2	1								
MTA	My Talk Address	M	AD	Y	1	0	T	T	T	T	T	XXX	1	X	X	X	X		
	(Note 4)						5	4	3	2	1								
MSA	My Secondary Address	M	SE	Y	1	1	S	S	S	S	S	XXX	1	X	X	X	X		
	(Note 5)						5	4	3	2	1								
NUL	Null Byte	M	DD	0	0	0	0	0	0	0	0	XXX	X	X	X	X	X		
OSA	Other Secondary Address	M	SE	(OSA = SCG ^ MSA)															
OTA	Other Talk Address	M	AD	(OTA = TAG ^ MTA)															
PCG	Primary Command Group	M	—	(PCG = ACG v UCG v LAG v TAG)															
PPC	Parallel Poll Configure	M	AC	Y	0	0	0	1	0	1	0	XXX	1	X	X	X	X		
PPE	Parallel Poll Enable	M	SE	Y	1	1	0	S	P	P	P	XXX	1	X	X	X	X		
	(Note 6)						3	2	1										
PPD	Parallel Poll Disable	M	SE	Y	1	1	1	D	D	D	D	XXX	1	X	X	X	X		
	(Note 7)						4	3	2	1									
PPR1	Parallel Poll Response 1	U	ST	X	X	X	X	X	X	X	X	XXX	1	1	X	X	X		
PPR2	Parallel Poll Response 2	U	ST	X	X	X	X	X	X	X	1	XXX	1	1	X	X	X		
PPR3	Parallel Poll Response 3	U	ST	X	X	X	X	X	1	X	X	XXX	1	1	X	X	X		
PPR4	Parallel Poll Response 4	U	ST	X	X	X	X	1	X	X	X	XXX	1	1	X	X	X		
PPR5	Parallel Poll Response 5	U	ST	X	X	X	1	X	X	X	X	XXX	1	1	X	X	X		
PPR6	Parallel Poll Response 6	U	ST	X	X	1	X	X	X	X	X	XXX	1	1	X	X	X		
PPR7	Parallel Poll Response 7	U	ST	X	1	X	X	X	X	X	X	XXX	1	1	X	X	X		
PPR8	Parallel Poll Response 8	U	ST	1	X	X	X	X	X	X	X	XXX	1	1	X	X	X		
PPU	Parallel Poll Unconfigure	M	UC	Y	0	0	1	0	1	0	1	XXX	1	X	X	X	X		
REN	Remote Enable	U	UC	X	X	X	X	X	X	X	X	XXX	X	X	X	X	1		
RFD	Ready for Data	U	HS	X	X	X	X	X	X	X	X	X0X	X	X	X	X	X		
RQS	Request Service	U	ST	X	1	X	X	X	X	X	X	XXX	0	X	X	X	X		
SCG	Secondary Command Group	M	SE	Y	1	1	X	X	X	X	X	XXX	1	X	X	X	X		
SDC	Selected Device Clear	M	AC	Y	0	0	0	1	0	0	0	XXX	1	X	X	X	X		
SPD	Serial Poll Disable	M	UC	Y	0	0	1	1	0	0	1	XXX	1	X	X	X	X		
SPE	Serial Poll Enable	M	UC	Y	0	0	1	1	0	0	0	XXX	1	X	X	X	X		
SRQ	Service Request	U	ST	X	X	X	X	X	X	X	X	XXX	X	1	X	X	X		
STB	Status Byte	M	ST	S	X	S	S	S	S	S	S	XXX	0	X	X	X	X		
	(Notes 8, 9)			8		6	5	4	3	2	1								
TCT	Take Control	M	AC	Y	0	0	0	1	0	0	1	XXX	1	X	X	X	X		
TAG	Talk Address Group	M	AD	Y	1	0	X	X	X	X	X	XXX	1	X	X	X	X		
UCG	Universal Command Group	M	UC	Y	0	0	1	X	X	X	X	XXX	1	X	X	X	X		
UNL	Unlisten	M	AD	Y	0	1	1	1	1	1	1	XXX	1	X	X	X	X		
UNT	Untalk	M	AD	Y	1	0	1	1	1	1	1	XXX	1	X	X	X	X		
	(Note 11)																		

The I/O coding on ATN when sent concurrent with multiline messages has been added to this revision for interpretive convenience.

## NOTES:

1. D1-D8 specify the device dependent data bits.
2. E1-E8 specify the device dependent code used to indicate the EOS message.
3. L1-L5 specify the device dependent bits of the device's listen address.
4. T1-T5 specify the device dependent bits of the device's talk address.
5. S1-S5 specify the device dependent bits of the device's secondary address.
6. S specifies the sense of the PPR.

$$\text{Response} = \overline{S} \oplus \text{lst}$$

P1-P3 specify the PPR message to be sent when a parallel poll is executed.

P3	P2	P1	PPR Message
0	0	0	PPR1
.	.	.	.
.	.	.	.
1	1	1	PPR8

7. D1-D4 specify don't-care bits that shall not be decoded by the receiving device. It is recommended that all zeroes be sent.
8. S1-S6, S8 specify the device dependent status. (DIO7 is used for the RQS message.)
9. The source of the message on the ATN line is always the C function, whereas the messages on the DIO and EOI lines are enabled by the T function.
10. The source of the messages on the ATN and EOI lines is always the C function, whereas the source of the messages on the DIO lines is always the PP function.
11. This code is provided for system use, see 6.3.

# 8293 GPIO TRANSCEIVER

- **Nine Open-collector or Three-state Line Drivers**
- **48 mA Sink Current Capability on Each Line Driver**
- **Nine Schmitt-type Line Receivers**
- **High Capacitance Load Drive Capability**
- **Single 5V Power Supply**
- **28-Pin Package**
- **Low Power HMOS Design**
- **On-chip Decoder for Mode Configuration**
- **Power Up/Power Down Protection to Prevent Disrupting the IEEE Bus**
- **Connects with the 8291A and 8292 to Form an IEEE Standard 488 Interface Talker/Listener/Controller with no Additional Components**
- **Only Two 8293's Required per GPIB Interface**
- **On-Chip IEEE-488 Bus Terminations**

The Intel® 8293 GPIB Transceiver is a high-current, non-inverting buffer chip designed to interface the 8291A GPIB Talker/Listener, or the 8291A/8292 GPIB Talker/Listener/Controller combination, to the IEEE Standard 488-1978 Instrumentation Interface Bus. Each GPIB interface would contain two 8293 Bus Transceivers. In addition, the 8293 can also be used as a general-purpose bus driver.

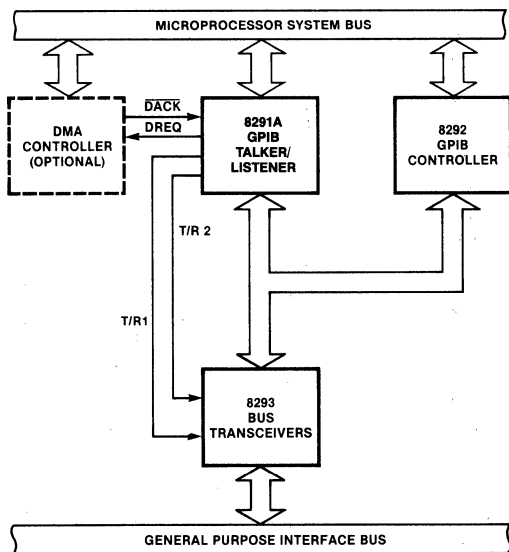


Figure 1. 8291A, 8292, 8293 Block Diagram

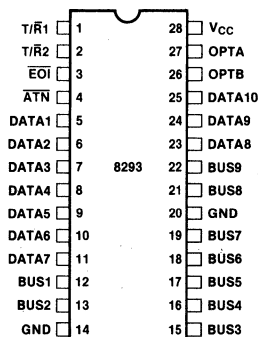


Figure 2. Pin Configuration



**Table 1. Pin Description**

Symbol	Pin No.	Type	Name and Function
BUS1-BUS9	12, 13, 15-19, 21, 22	I/O	<b>GPIB Lines, GPIB Side:</b> These are the IEEE-488 bus interface driver/receivers, or TTL-compatible inputs on the 8291A/8292 side, depending on the mode used. Their use is programmed by the two mode select pins, OPTA and OPTB.
DATA1-DATA10	5-11, 23-25	I/O	<b>GPIB Lines, 8291A/92 Side:</b> These are the pins to be connected to the 8291A and 8292 to interface with the GPIB. Their use is programmed by the two mode select pins, OPTA and OPTB. All these pins are TTL compatible.
T/R1	1	I	<b>Transmit Receive 1:</b> This pin controls the direction for NDAC, NRFD, DAV, and DIO1-DIO8. Input is TTL compatible.
T/R2	2	I	<b>Transmit Receive 2:</b> This pin controls the direction for EOI. Input is TTL compatible.

Symbol	Pin No.	Type	Name and Function
EOI	3	I/O	<b>End Or Identify:</b> This pin indicates the end of a multiple byte transfer or, in conjunction with ATN, addresses the device during a polling sequence. It connects to the 8291A and is switched between transmit and receive by T/R2. This pin is TTL compatible.
ATN	4	O	<b>Attention:</b> This pin is used by the 8291A to monitor the GPIB ATN control line. It specifies how data on the DIO lines is to be interpreted. This output is TTL compatible.
OPTA OPTB	27 26	I I	<b>Mode Select:</b> These two pins are to control the function of the 8293. A truth table of how they program the various modes is in Table 2.
VCC	28	P.S.	<b>Voltage:</b> Positive power supply (5V ± 10%).
GND	14, 20	P.S.	<b>Ground:</b> Circuit ground.

**Table 2. 8293 Mode Selection Pin Mapping**

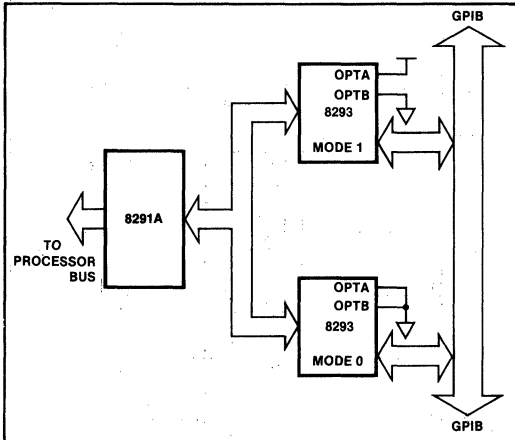
Pin Name	Pin No.	IEEE Implementation Name			
		Mode 0	Mode 1	Mode 2	Mode 3
OPTA	27	0	1	0	1
OPTB	26	0	0	1	1
DATA1	5	IFC	DIO8	IFC	DIO8
BUS1	12	IFC*	DIO8*	IFC*	DIO8*
DATA2	6	REN	DIO7	REN	DIO7
BUS2	13	REN*	DIO7*	REN*	DIO7*
DATA3	7	NC	DIO6	EOI2	DIO6
BUS3	15	EOI*	DIO6*	EOI*	DIO6*
DATA4	8	SRQ	DIO5	SRQ	DIO5
BUS4	16	SRQ*	DIO5*	SRQ*	DIO5*
DATA5	9	NRFD	DIO4	NRFD	DIO4
BUS5	17	NRFD*	DIO4*	NRFD*	DIO4*
DATA6	10	NDAC	DIO3	NDAC	DIO3
BUS6	18	NDAC*	DIO3*	NDAC*	DIO3*
DATA7	11	T/RIO1	NC	ATNI	ATNO
DATA8	23	T/RIO2	DIO2	ATNO	DIO2
BUS7	19	ATN*	DIO2*	ATN*	DIO2*
DATA9	24	GIO1	DAV	CIC	DAV
BUS8	21	GIO1*	DAV*	CLTH	DAV*
DATA10	25	GIO2	DIO1	IFCL	DIO1
BUS9	22	GIO2*	DIO1*	SYC	DIO1*
T/R1	1	T/R1	T/R1	T/R1	T/R1
T/R2	2	T/R2	NC	T/R2	IFCL
EOI	3	EOI	EOI	EOI	EOI
ATN	4	ATN	ATN	ATN	ATN

\*Note: These pins are the IEEE-488 bus non-inverting driver/receivers. They include all the bus terminations required by the Standard and may be connected directly to the GPIB bus connector.

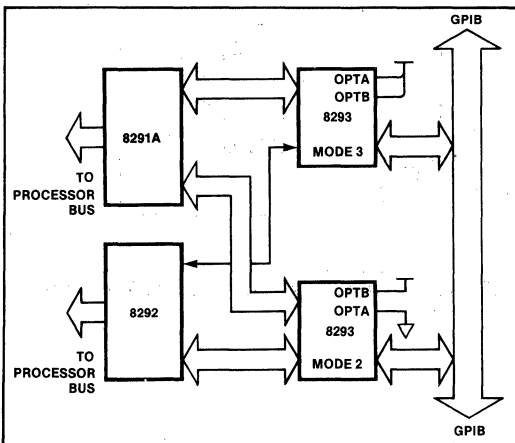
**GENERAL DESCRIPTION**

The 8293 is a bidirectional transceiver. It was designed to interface the Intel 8291A GPIB Talker/Listener and the Intel® 8292 GPIB Controller to the IEEE Standard 488-1978 Instrumentation Bus (also referred to as the GPIB). The Intel GPIB Transceiver meets or exceeds all of the electrical specifications defined in the IEEE Standard 488-1978, Section 3.3-3.5, including the bus termination specifications.

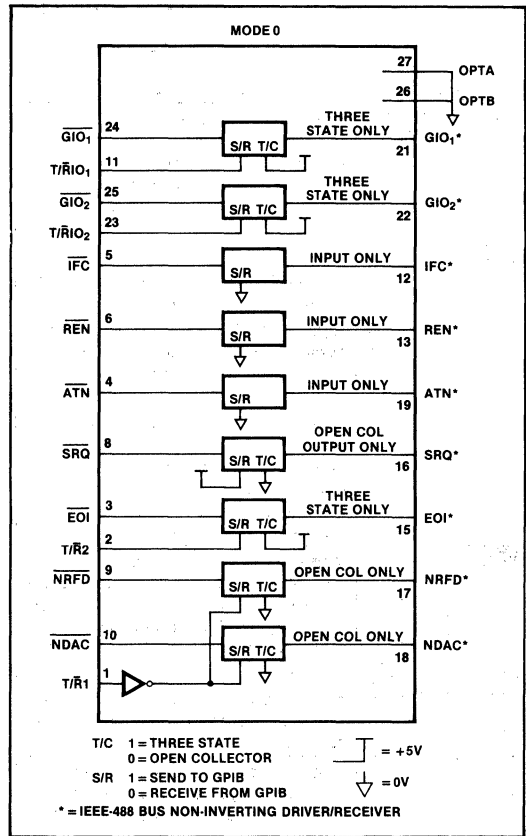
The 8293 can be hardware programmed to one of four modes of operation. These modes allow the 8293 to be configured to support both a Talker/Listener/Controller environment and a Talker/Listener environment. In addition, the 8293 can be used as a general-purpose, three-state (push-pull) or open-collector bus transceiver with nine receiver/drivers. Two modes each are used to support a Talker/Listener (see Figure 3) and a Talker/Listener/Controller environment (see Figure 4). Mode 1 is used in general-purpose environments.



**Figure 3. Talker/Listener Configuration**



**Figure 4. Talker/Listener/Controller Configuration**



**Figure 5. Talker/Listener Control Configuration**

**Table 3. Mode 0 Pin Description**

Symbol	Pin No.	Type	Name and Function
T/ $\bar{R}$ 1	1	I	<b>Transmit Receive 1</b> Direction control for NDAC and NRFD. If T/R1 is high, then NDAC* and NRFD* are receiving. Input is TTL compatible.
NDAC	10	I/O	<b>Not Data Accepted:</b> Processor GPIB bus handshake control line; used to indicate the condition of acceptance of data by device(s). It is TTL compatible.
NDAC*	18	I/O	<b>Not Data Accepted:</b> IEEE GPIB bus handshake control line. When an input, it is a TTL compatible Schmitt-trigger. When an output, it is an open-collector driver with 48 mA sinking capability.
NRFD	9	I/O	<b>Not Ready For Data:</b> Processor GPIB bus handshake control line; used to indicate the condition of readiness of device(s) to accept data. This pin is TTL compatible.

Table 3. Mode 0 Pin Description (Continued)

Symbol	Pin No.	Type	Name and Function
NRFD*	17	I/O	<b>Not Ready For Data:</b> IEEE GPIB bus handshake control line. When an input, it is a TTL compatible Schmitt-trigger. When an output, it is an open-collector driver with a 48 mA current sinking capability.
T/R <sub>2</sub>	2	I	<b>Transmit Receive 2:</b> Direction control for EOI. If T/R <sub>2</sub> is high, EOI* is sending. Input is TTL compatible.
EOI	3	I/O	<b>End Or Identify:</b> Processor GPIB bus control line; is used by a talker to indicate the end of a multiple byte transfer. This pin is TTL compatible.
EOI*	15	I/O	<b>End Or Identify:</b> IEEE GPIB bus control line; is used by a talker to indicate the end of a multiple byte transfer. This pin is a three-state (push-pull) driver capable of sinking 48 mA and a TTL compatible receiver with hysteresis.
SRQ	8	I	<b>Service Request:</b> Processor GPIB bus control line; used by a device to indicate the need for service and to request an interruption of the current sequence of events on the GPIB. It is a TTL compatible input.
SRQ*	16	O	<b>Service Request:</b> IEEE GPIB bus control line; it is an open collector driver capable of sinking 48 mA.
REN	6	O	<b>Remote Enable:</b> Processor GPIB bus control line; used by a controller (in conjunction with other messages) to select between two alternate sources of device programming data (remote or local control). This output is TTL compatible.
REN*	13	I	<b>Remote Enable:</b> IEEE GPIB bus control line. This input is a TTL compatible Schmitt-trigger.
ATN	4	O	<b>Attention:</b> Processor GPIB bus control line; used by the 8291 to determine how data on the DIO signal lines are to be interpreted. This is a TTL compatible output.
ATN*	19	I	<b>Attention:</b> IEEE GPIB bus control line; this input is a TTL compatible Schmitt-trigger.
IFC	5	O	<b>Interface Clear:</b> Processor GPIB bus control line; used by a controller to place the interface system into a known quiescent state. It is a TTL compatible output.

Symbol	Pin No.	Type	Name and Function
IFC*	12	I	<b>Interface Clear:</b> IEEE GPIB bus control line. This input is a TTL compatible Schmitt-trigger.
T/R <sub>101</sub> T/R <sub>102</sub>	11 23	I I	<b>Transmit Receive General IO:</b> Direction control for the two spare transceivers. These pins are TTL compatible.
GIO <sub>1</sub> GIO <sub>2</sub>	24 25	I/O I/O	<b>General IO:</b> This is the TTL side of the two spare transceivers. These pins are TTL compatible.
GIO1* GIO2*	21 22	I/O I/O	<b>General IO:</b> These are spare three-state (push-pull) drivers/Schmitt-trigger receivers. The drivers can sink 48 mA.

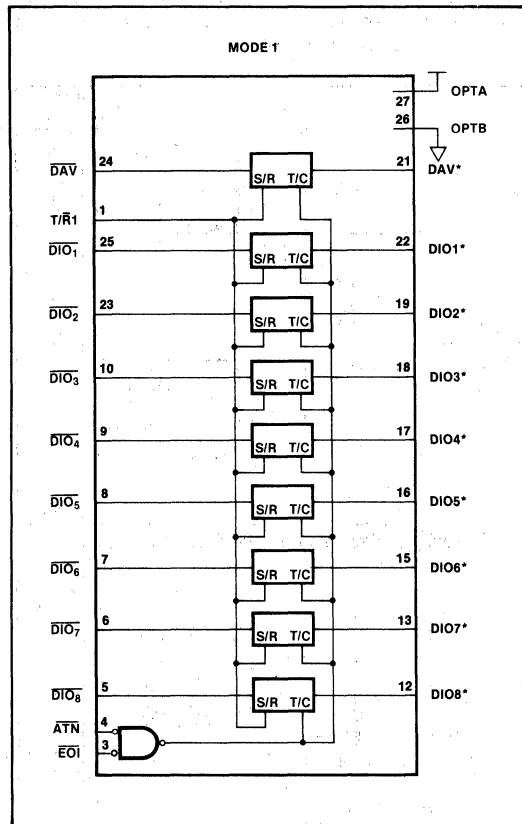


Figure 6. Talker/Listener Data Configuration

Table 4. Mode 1 Pin Description

Symbol	Pin No.	Type	Name and Function
T/R1	1	I	<b>Transmit Receive 1:</b> Controls the direction for DAV and the DIO lines. If T/R1 is high, then all these lines are sending information to the IEEE GPIB lines. This input is TTL compatible.
EOI ATN	3 4	I	<b>End Of Sequence And Attention:</b> Processor GPIB control lines. These two control signals are ANDed together to determine whether all the transceivers in the 8293 are three-state (push-pull) or open-collector. When both signals are low (true), then the controller is performing a parallel poll and the transceivers are all open-collector. These inputs are TTL compatible.
DAV	24	I/O	<b>Data Valid:</b> Processor GPIB bus handshake control line; used to indicate the condition (availability and validity) of information on the DIO lines. It is TTL compatible.
DAV*	21	I/O	<b>Data Valid:</b> IEEE GPIB bus handshake control line. When an input, it is a TTL compatible Schmitt-trigger. When DAV* is an output, it can sink 48 mA.
DIO1- DIO8	25, 23, 10, 9, 8, 7, 6, 5	I/O	<b>Data Input/Output:</b> Processor GPIB bus data lines; used to carry message and data bytes in a bit-parallel byte-serial form controlled by the three handshake signals. These lines are TTL compatible.
DIO1* DIO8*	22, 19, 18, 17, 16, 15, 13, 12	I/O	<b>Data Input/Output:</b> IEEE GPIB bus data lines. They are TTL compatible Schmitt-triggers when used for input and can sink 48 mA when used for output. See ATN and EOI description for output mode.

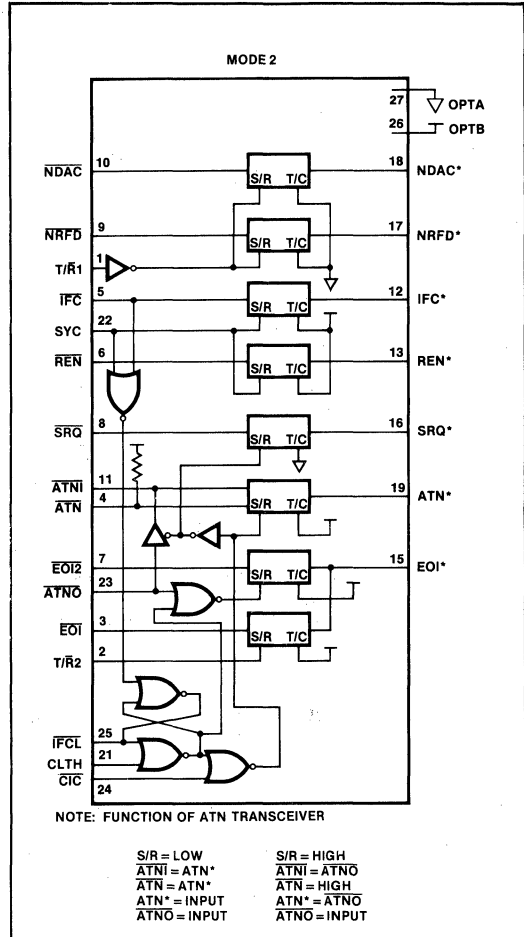


Figure 7. Talker/Listener/Controller Control Configuration

Table 5. Mode 2 Pin Description

Symbol	Pin No.	Type	Name and Function
$\overline{T/R1}$	1	I	<b>Transmit Receive 1:</b> Direction control for NDAC and NRFD. If $\overline{T/R1}$ is high, then NDAC and NRFD are receiving. Input is TTL compatible.
NDAC	10	I/O	<b>Not Data Accepted:</b> Processor GPIB bus handshake control line; used to indicate the condition of acceptance of data by device(s). This pin is TTL compatible.
NDAC*	18	I/O	<b>Not Data Accepted:</b> IEEE GPIB bus handshake control line. It is a TTL compatible Schmitt-trigger when used for input and an open-collector driver with a 48 mA current sink capability when used for output.
NRFD	9	I/O	<b>Not Ready For Data:</b> Processor GPIB bus handshake control line; used to indicate the condition of readiness of device(s) to accept data. This pin is TTL compatible.
NRFD*	17	I/O	<b>Not Ready For Data:</b> IEEE GPIB bus handshake control line. It is a TTL compatible Schmitt-trigger when used for input and an open-collector driver with a 48 mA current sink capability when used for output.
SYC <sup>1</sup>	22	I	<b>System Controller:</b> Used to monitor the system controller switch and control the direction for IFC and REN. This pin is a TTL compatible input.
REN	6	I/O	<b>Remote Enable:</b> Processor GPIB control line; used by the active controller (in conjunction with other messages) to select between two alternate sources of device programming data (remote or local control). This pin is TTL compatible.
REN*	13	I/O	<b>Remote Enable:</b> IEEE GPIB bus control line. When used as an input, this is a TTL compatible Schmitt-trigger. When an output, it is a three-state driver with a 48 mA current sinking capability.
IFC	5	I/O	<b>Interface Clear:</b> Processor GPIB bus control line; used by the active controller to place the interface system into a known quiescent state. This pin is TTL compatible.
IFC*	12	I/O	<b>Interface Clear:</b> IEEE GPIB control line. This is a TTL compatible Schmitt-trigger when used for input and a three-state driver capable of sinking 48 mA current when used for output.
$\overline{CIC}$	24	I	<b>Controller In Charge:</b> Used to control the direction of the SRQ and to indicate that the 8292 is in charge of the bus. $\overline{CIC}$ is a TTL compatible input.

Symbol	Pin No.	Type	Name and Function
CLTH <sup>1</sup>	21	I	<b>Clear Latch:</b> Used to clear the IFC Received latch after it has been recognized by the 8292. Normally low (except after a hardware reset). It will be pulsed high when IFC Received is recognized by the 8292. This input is TTL compatible.
$\overline{IFCL}$	25	O	<b>IFC Received Latch:</b> The 8292 monitors the IFC line when it is not the active controller through this pin.
$\overline{SRQ}$	8	I/O	<b>Service Request:</b> Processor GPIB control line; indicates the need for attention and requests the active controller to interrupt the current sequence of events on the GPIB bus. This pin is TTL compatible.
SRQ*	16	I/O	<b>Service Request:</b> IEEE GPIB bus control line. When used as an input, this pin is a TTL compatible Schmitt-trigger. When used as an output, it is an open-collector driver with a 48 mA current sinking capability.
$\overline{T/R2}$	2	I	<b>Transmit Receive 2:</b> Controls the direction for EOI. This input is TTL compatible.
$\overline{ATNO}$	23	I	<b>Attention Out:</b> Processor GPIB bus control line; used by the 8292 for ATN control of the IEEE bus during "take control synchronously" operations. A low on this input causes ATN to be asserted if $\overline{CIC}$ indicates that this 8292 is in charge. $\overline{ATNO}$ is a TTL compatible input.
$\overline{ATNI}$	11	O	<b>Attention In:</b> Processor GPIB bus control line; used by the 8292 to monitor the ATN line. This output is TTL compatible.
$\overline{ATN}$	4	O	<b>Attention:</b> Processor GPIB bus control line; used by the 8292 to monitor the ATN line. This output is TTL compatible.
ATN*	19	I/O	<b>Attention:</b> IEEE GPIB bus control line; used by a controller to specify how data on the DIO signal lines are to be interpreted and which devices must respond to data. When used as an output, this pin is a three-state driver capable of sinking 48 mA current. As an input, it is a TTL compatible Schmitt-trigger.
$\overline{EOI2}$	7	I/O	<b>End Or Identify 2:</b> Processor GPIB bus control line; used in conjunction with ATN by the active controller (the 8292) to execute a polling sequence. This pin is TTL compatible.
$\overline{EOI}$	3	I/O	<b>End Or Identify:</b> Processor GPIB bus control line; used by a talker to indicate the end of a multiple byte transfer sequence. This pin is TTL compatible.

## NOTES:

- $V_{IL3}$  is guaranteed at 1.1V on these inputs to accommodate the high current-sourcing capability of these pins during a low input in Mode 2.

Table 5. Mode 2 Pin Description (Continued)

Symbol	Pin No.	Type	Name and Function
EOI*	15	I/O	<b>End Or Identify:</b> IEEE GPIB bus control line; used by a talker to indicate the end of a multiple byte transfer sequence or, by a controller in conjunction with ATN, to execute a polling sequence. When an output, this pin can sink 48 mA current. When an input, it is a TTL compatible Schmitt-trigger.

Table 6. Mode 3 Pin Description

Symbol	Pin No.	Type	Name and Function
T/R1	1	I	<b>Transmit Receive 1:</b> Controls the direction for DAV and the DIO lines. If T/R1 is high, then all these lines are sending information to the IEEE GPIB lines. This input is TTL compatible.
EOI ATN	3 4	I I	<b>End Of Sequence and Attention:</b> Processor GPIB control lines. These two control lines are ANDed together to determine whether all the transceivers in the 8293 are push-pull or open-collector. When both signals are low (true), then the controller is performing a parallel poll and the transceivers are all open-collector. These inputs are TTL compatible.
ATNO	11	I	<b>Attention Out:</b> Processor GPIB control line; used by the 8292 during "take control synchronously" operations. This pin is TTL compatible.
IFCL	2	I	<b>Interface Clear Latched:</b> Used to make DAV received after the system controller asserts IFC. This input is TTL compatible.
DAV	24	I/O	<b>Data Valid:</b> Processor GPIB handshake control line; used to indicate the condition (availability and validity) of information on the DIO signals. This pin is TTL compatible.
DAV*	21	I/O	<b>Data Valid:</b> IEEE GPIB handshake control line. When an input, this pin is a TTL compatible Schmitt-trigger. When DAV* is an output, it can sink 48 mA.
DIO1- DIO8	25, 23, 10, 9, 8, 7, 6, 5	I/O	<b>Data Input/Output:</b> Processor GPIB bus data lines; used to carry message and data bytes in a bit-parallel byte-serial from controlled by the three handshake signals. These lines are TTL compatible.
DIO1* DIO8*	22, 19, 18, 17, 16, 15, 13, 12	I/O	<b>Data Input/Output:</b> IEEE GPIB bus data lines. They are TTL compatible Schmitt-triggers when used for input and can sink 48 mA when used for output.

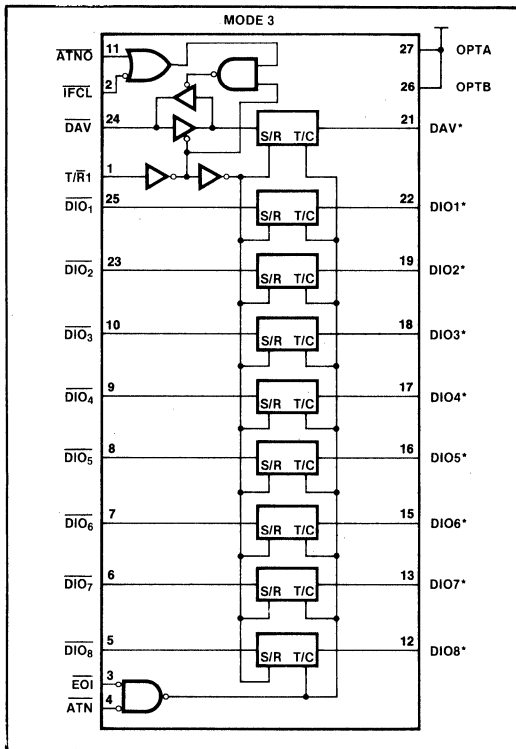


Figure 8. Talker/Listener/Controller Data Configuration

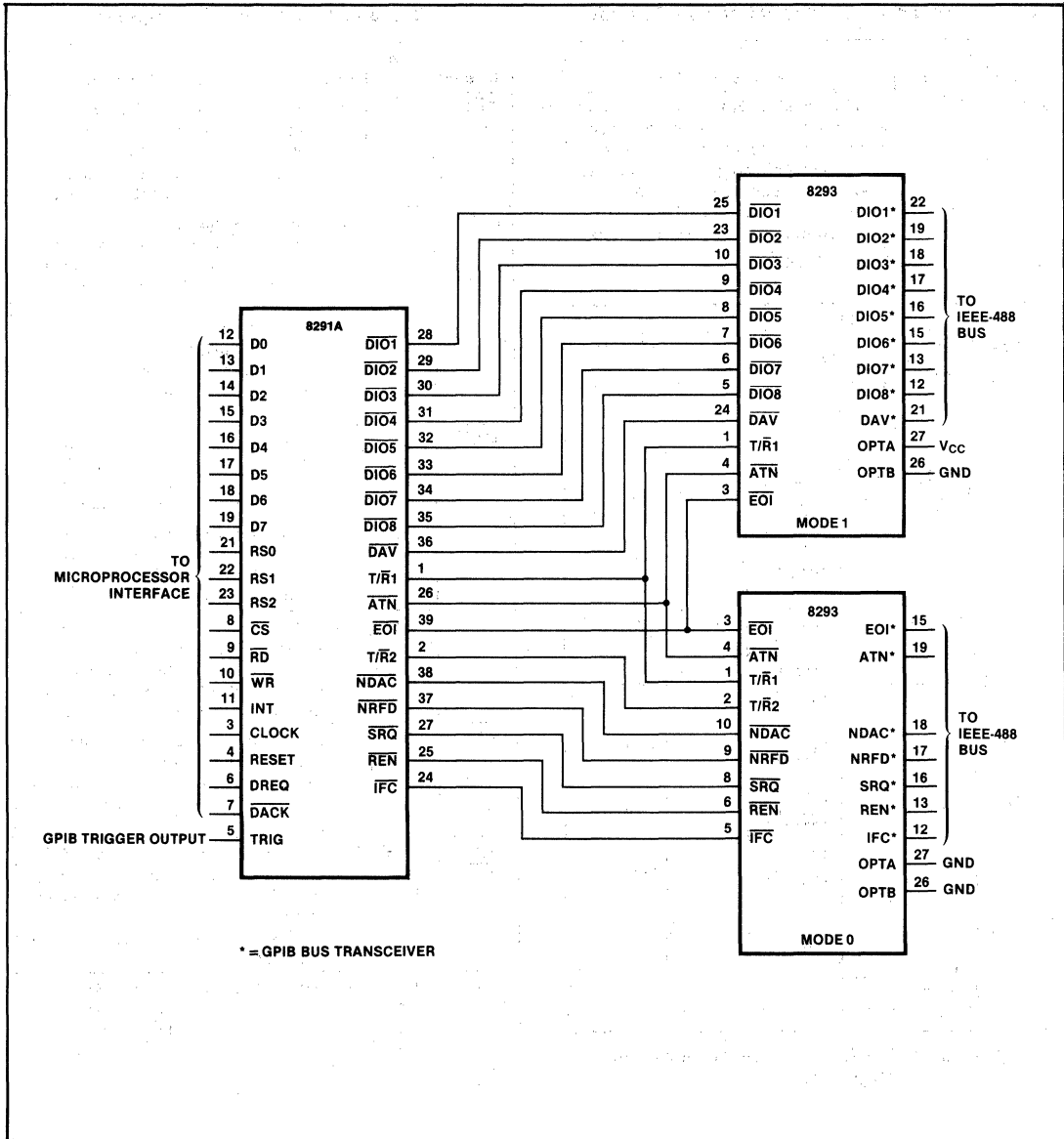


Figure 9. 8291A and 8293 System Configuration





**ABSOLUTE MAXIMUM RATINGS\***

Ambient Temperature Under Bias..... 0°C to 70°C  
 Storage Temperature..... - 65°C to + 150°C  
 Voltage on any Pin with  
     Respect to Ground..... - 1.0V to + 7V  
 Power Dissipation..... 1 Watt

*This is a stress rating only and functional operation of the device at these or any other conditions above those indicated in the operational sections of this specification is not implied. Exposure to absolute maximum rating conditions for extended periods may affect device reliability.*  
 2. All devices are guaranteed to operate within the minimum and maximum parameter limits specified below. Typical parameters however are not tested and are not guaranteed. Established statistically, they indicate the performance level expected in a typical device at room temperature ( $T_A = 25^\circ\text{C}$ ) and  $V_{CC} = 5\text{V}$ .

**\*NOTICE:**

1. Stresses above those listed under "Absolute Maximum Ratings" may cause permanent damage to the device.

**D.C. CHARACTERISTICS** ( $T_A = 0^\circ\text{C}$  to  $70^\circ\text{C}$ ,  $V_{CC} = 5.0\text{V} \pm 10\%$ ,  $\text{GND} = 0\text{V}$ )

Symbol	Parameter	Limits			Units	Test Conditions
		Min.	Typ.	Max.		
V <sub>IL1</sub>	Input Low Voltage ( GPIB Bus Pins)			0.8	V	
V <sub>IL2</sub>	Input Low Voltage (Option Pins)	-0.1		0.1	V	
V <sub>IL3</sub> <sup>1</sup>	Input Low Voltage (All Others)			0.8	V	
V <sub>IH1</sub>	Input High Voltage ( GPIB Bus Pins)	2.0		V <sub>CC</sub>	V	
V <sub>IH2</sub>	Input High Voltage (Option Pins)	4.5		V <sub>CC</sub>	V	
V <sub>IH3</sub>	Input High voltage (All Others)	2.0		V <sub>CC</sub>	V	
V <sub>IH4</sub>	Receiver Input Hysteresis	400			mV	
V <sub>OL1</sub>	Output Low Voltage ( GPIB Bus Pins)			0.5	V	I <sub>OL</sub> = 48 mA
V <sub>OL2</sub>	Output Low Voltage (All Others)			0.5	V	I <sub>OL</sub> = 16 mA
V <sub>OH1</sub>	Output High Voltage ( GPIB Bus Pins)	2.4			V	I <sub>OH</sub> = -5.2 mA
V <sub>OH2</sub>	Output High Voltage (All Others)	2.4			V	I <sub>OH</sub> = -800 $\mu\text{A}$
V <sub>IT</sub>	Receiver Input Threshold	High to Low	0.8		V	
		Low to High		2.0		
I <sub>LC</sub>	Input Load Current ( GPIB Pins)	See Bus Load Line Diagram				V <sub>CC</sub> = 5.0V $\pm$ 5%
I <sub>IL</sub>	Input Leakage Current (All Others)			10	$\mu\text{A}$	0.45 $\leq$ V <sub>IN</sub> $\leq$ V <sub>CC</sub>
I <sub>PD</sub>	Bus Power Down Leakage Current			40	$\mu\text{A}$	0.45V $\leq$ V <sub>BUS</sub> $\leq$ 2.7V
I <sub>CC</sub>	Power Supply Current		110	175	mA	

**NOTES:**

1. V<sub>IL3</sub> = 1.1V max on pins 21 and 22 in Mode 2 for the 8293-10.

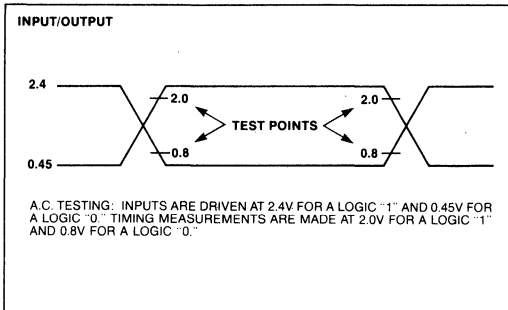
**CAPACITANCE**

Symbol	Parameter	Min.	Typ.	Max.	Units	Test Conditions
C <sub>IO1</sub>	I/O Capacitance ( GPIB Side)		50	80	pF	V <sub>IN</sub> = V <sub>CC</sub>
C <sub>IO2</sub>	I/O Capacitance (System Side)		35	50	pF	V <sub>IN</sub> = V <sub>CC</sub>
C <sub>ITR</sub>	Input Capacitance (T/R1, T/R2)		7	10	pF	V <sub>IN</sub> = V <sub>CC</sub>

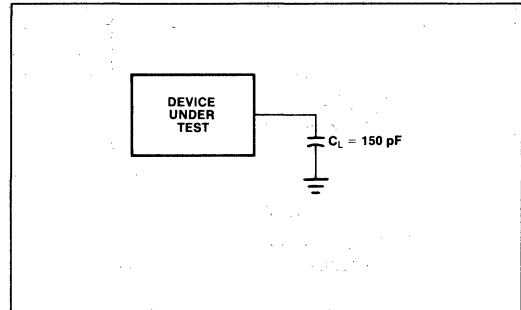
**A.C. CHARACTERISTICS** ( $T_A = 0^\circ\text{C}$  to  $70^\circ\text{C}$ ,  $V_{CC} = 5.0\text{V} \pm 10\%$ ,  $\text{GND} = 0\text{V}$ )

Symbol	Parameter	Max.	Units
$t_{p1}$	Transmitter Propagation Delay (All Lines)	30	ns
$t_{p2}$	Receiver Propagation Delay (EOI, ATN and Handshake Lines)	50	ns
$t_{p3}$	Receiver Propagation Delay (All Other Lines)	60	ns
$t_{pHZ1}$	Transmitter Disable Delay (High to 3-State)	40	ns
$t_{pZH1}$	Transmitter Enable Delay (3-state to High)	40	ns
$t_{pLZ1}$	Transmitter Disable Delay (Low to 3-State)	40	ns
$t_{pZL1}$	Transmitter Enable Delay (3-State to Low)	40	ns
$t_{pHZ2}$	Receiver Disable Delay (High to 3-State)	40	ns
$t_{pZH2}$	Receiver Enable Delay (3-State to High)	40	ns
$t_{pLZ2}$	Receiver Disable Delay (Low to 3-State)	40	ns
$t_{pZL2}$	Receiver Enable Delay (3-State to Low)	40	ns
$t_{MS}$	Mode Switch Delay	10	$\mu\text{s}$

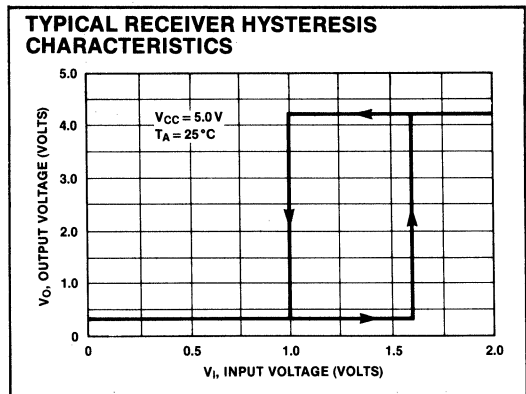
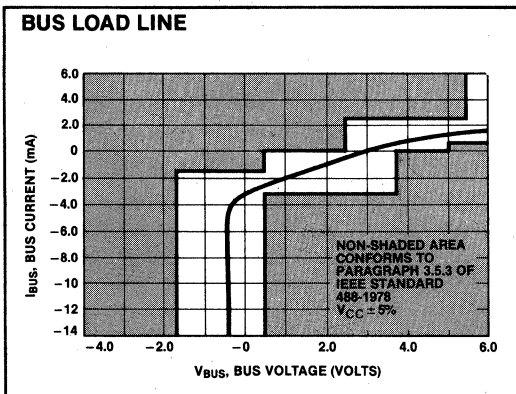
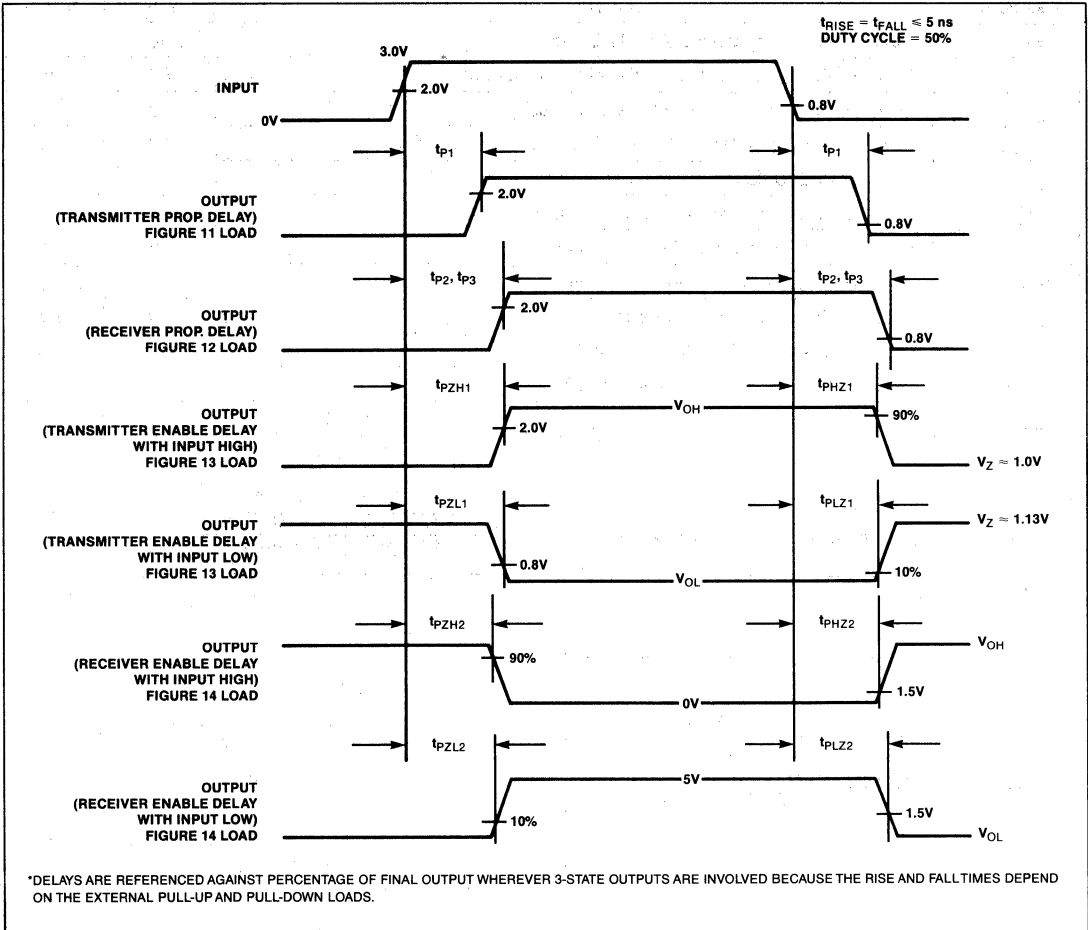
**A.C. TESTING INPUT, OUTPUT WAVEFORM**



**A.C. TESTING LOAD CIRCUIT FOR PROPAGATION DELAYS**



WAVEFORMS



---

# Controllers

... ..

---

... ..

... ..

... ..

... ..

... ..

... ..

... ..



# 8253/8253-5 PROGRAMMABLE INTERVAL TIMER

- MCS-85™ Compatible 8253-5
  - 3 Independent 16-Bit Counters
  - DC to 2 MHz
  - Programmable Counter Modes
- Count Binary or BCD
  - Single +5V Supply
  - 24-Pin Dual In-Line Package

The Intel® 8253 is a programmable counter/timer chip designed for use as an Intel microcomputer peripheral. It uses nMOS technology with a single +5V supply and is packaged in a 24-pin plastic DIP. It is organized as 3 independent 16-bit counters, each with a count rate of up to 2 MHz. All modes of operation are software programmable.

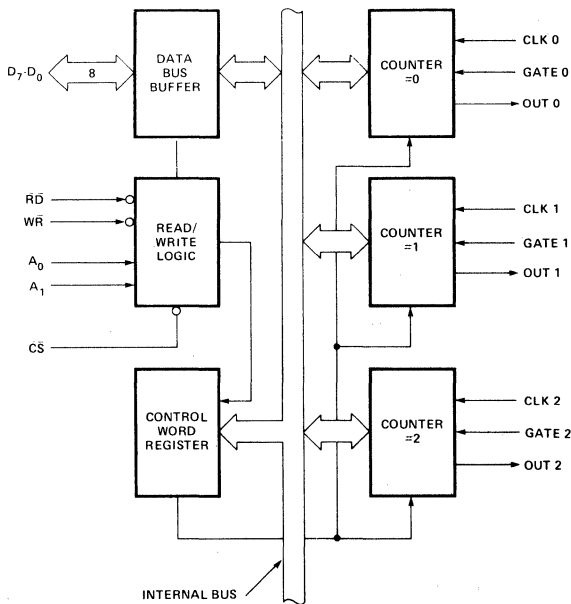


Figure 1. Block Diagram

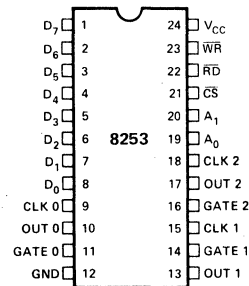


Figure 2. Pin Configuration

## FUNCTIONAL DESCRIPTION

### General

The 8253 is a programmable interval timer/counter specifically designed for use with the Intel™ Micro-computer systems. Its function is that of a general purpose, multi-timing element that can be treated as an array of I/O ports in the system software.

The 8253 solves one of the most common problems in any microcomputer system, the generation of accurate time delays under software control. Instead of setting up timing loops in systems software, the programmer configures the 8253 to match his requirements, initializes one of the counters of the 8253 with the desired quantity, then upon command the 8253 will count out the delay and interrupt the CPU when it has completed its tasks. It is easy to see that the software overhead is minimal and that multiple delays can easily be maintained by assignment of priority levels.

Other counter/timer functions that are non-delay in nature but also common to most microcomputers can be implemented with the 8253.

- Programmable Rate Generator
- Event Counter
- Binary Rate Multiplier
- Real Time Clock
- Digital One-Shot
- Complex Motor Controller

### Data Bus Buffer

This 3-state, bi-directional, 8-bit buffer is used to interface the 8253 to the system data bus. Data is transmitted or received by the buffer upon execution of INput or OUTput CPU instructions. The Data Bus Buffer has three basic functions.

1. Programming the MODES of the 8253.
2. Loading the count registers.
3. Reading the count values.

### Read/Write Logic

The Read/Write Logic accepts inputs from the system bus and in turn generates control signals for overall device operation. It is enabled or disabled by CS so that no operation can occur to change the function unless the device has been selected by the system logic.

### RD (Read)

A "low" on this input informs the 8253 that the CPU is inputting data in the form of a counters value.

### WR (Write)

A "low" on this input informs the 8253 that the CPU is outputting data in the form of mode information or loading counters.

### A0, A1

These inputs are normally connected to the address bus. Their function is to select one of the three counters to be operated on and to address the control word register for mode selection.

### CS (Chip Select)

A "low" on this input enables the 8253. No reading or writing will occur unless the device is selected. The CS input has no effect upon the actual operation of the counters.

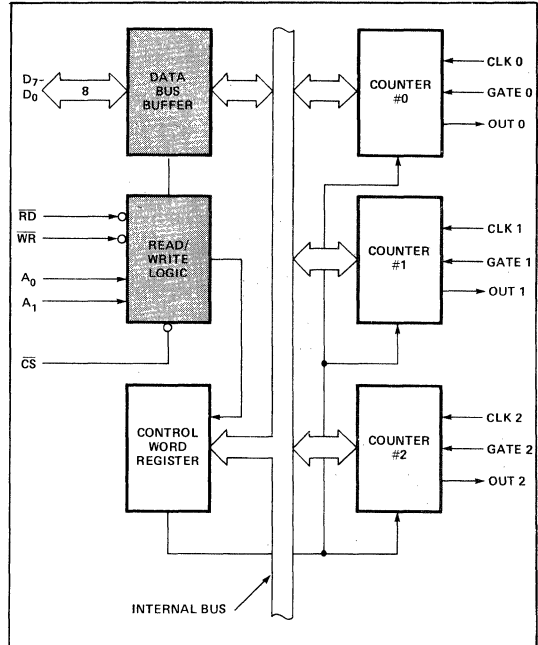


Figure 3. Block Diagram Showing Data Bus Buffer and Read/Write Logic Functions

CS	RD	WR	A <sub>1</sub>	A <sub>0</sub>	
0	1	0	0	0	Load Counter No. 0
0	1	0	0	1	Load Counter No. 1
0	1	0	1	0	Load Counter No. 2
0	1	0	1	1	Write Mode Word
0	0	1	0	0	Read Counter No. 0
0	0	1	0	1	Read Counter No. 1
0	0	1	1	0	Read Counter No. 2
0	0	1	1	1	No-Operation 3-State
1	X	X	X	X	Disable 3-State
0	1	1	X	X	No-Operation 3-State

**Control Word Register**

The Control Word Register is selected when A0, A1 are 11. It then accepts information from the data bus buffer and stores it in a register. The information stored in this register controls the operational MODE of each counter, selection of binary or BCD counting and the loading of each count register.

The Control Word Register can only be written into; no read operation of its contents is available.

**Counter #0, Counter #1, Counter #2**

These three functional blocks are identical in operation so only a single Counter will be described. Each Counter consists of a single, 16-bit, pre-settable, DOWN counter. The counter can operate in either binary or BCD and its input, gate and output are configured by the selection of MODES stored in the Control Word Register.

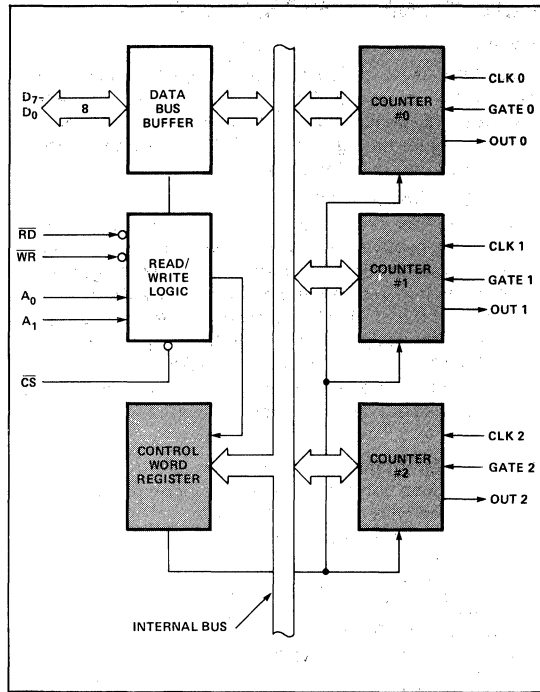
The counters are fully independent and each can have separate Mode configuration and counting operation, binary or BCD. Also, there are special features in the control word that handle the loading of the count value so that software overhead can be minimized for these functions.

The reading of the contents of each counter is available to the programmer with simple READ operations for event counting applications and special commands are included in the 8253 so that the contents of each counter can be read "on the fly" without having to inhibit the clock input.

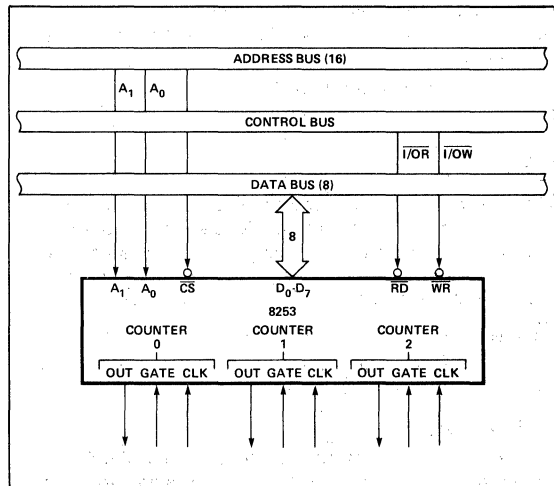
**8253 SYSTEM INTERFACE**

The 8253 is a component of the Intel™ Microcomputer Systems and interfaces in the same manner as all other peripherals of the family. It is treated by the systems software as an array of peripheral I/O ports; three are counters and the fourth is a control register for MODE programming.

Basically, the select inputs A0, A1 connect to the A0, A1 address bus signals of the CPU. The CS can be derived directly from the address bus using a linear select method. Or it can be connected to the output of a decoder, such as an Intel® 8205 for larger systems.



**Figure 4. Block Diagram Showing Control Word Register and Counter Functions**



**Figure 5. 8253 System Interface**

## OPERATIONAL DESCRIPTION

### General

The complete functional definition of the 8253 is programmed by the systems software. A set of control words must be sent out by the CPU to initialize each counter of the 8253 with the desired MODE and quantity information. Prior to initialization, the MODE, count, and output of all counters is undefined. These control words program the MODE, Loading sequence and selection of binary or BCD counting.

Once programmed, the 8253 is ready to perform whatever timing tasks it is assigned to accomplish.

The actual counting operation of each counter is completely independent and additional logic is provided on-chip so that the usual problems associated with efficient monitoring and management of external, asynchronous events or rates to the microcomputer system have been eliminated.

### Programming the 8253

All of the MODES for each counter are programmed by the systems software by simple I/O operations.

Each counter of the 8253 is individually programmed by writing a control word into the Control Word Register. (A0, A1 = 11)

### Control Word Format

D <sub>7</sub>	D <sub>6</sub>	D <sub>5</sub>	D <sub>4</sub>	D <sub>3</sub>	D <sub>2</sub>	D <sub>1</sub>	D <sub>0</sub>
SC1	SC0	RL1	RL0	M2	M1	M0	BCD

### Definition of Control

#### SC — Select Counter:

SC1		SC0		
0	0	0	0	Select Counter 0
0	0	0	1	Select Counter 1
1	0	0	0	Select Counter 2
1	0	0	1	Illegal

#### RL — Read/Load:

RL1	RL0	
0	0	Counter Latching operation (see READ/WRITE Procedure Section)
1	0	Read/Load most significant byte only.
0	1	Read/Load least significant byte only.
1	1	Read/Load least significant byte first, then most significant byte.

#### M — MODE:

M2	M1	M0	
0	0	0	Mode 0
0	0	1	Mode 1
X	1	0	Mode 2
X	1	1	Mode 3
1	0	0	Mode 4
1	0	1	Mode 5

#### BCD:

0	Binary Counter 16-bits
1	Binary Coded Decimal (BCD) Counter (4 Decades)

### Counter Loading

The count register is not loaded until the count value is written (one or two bytes, depending on the mode selected by the RL bits), followed by a rising edge and a falling edge of the clock. Any read of the counter prior to that falling clock edge may yield invalid data.

### MODE Definition

**MODE 0: Interrupt on Terminal Count.** The output will be initially low after the mode set operation. After the count is loaded into the selected count register, the output will remain low and the counter will count. When terminal count is reached the output will go high and remain high until the selected count register is reloaded with the mode or a new count is loaded. The counter continues to decrement after terminal count has been reached.

Rewriting a counter register during counting results in the following:

- (1) Write 1st byte stops the current counting.
- (2) Write 2nd byte starts the new count.

**MODE 1: Programmable One-Shot.** The output will go low on the count following the rising edge of the gate input.

The output will go high on the terminal count. If a new count value is loaded while the output is low it will not affect the duration of the one-shot pulse until the succeeding trigger. The current count can be read at any time without affecting the one-shot pulse.

The one-shot is retriggerable, hence the output will remain low for the full count after any rising edge of the gate input.



**MODE 2: Rate Generator.** Divide by N counter. The output will be low for one period of the input clock. The period from one output pulse to the next equals the number of input counts in the count register. If the count register is reloaded between output pulses the present period will not be affected, but the subsequent period will reflect the new value.

The gate input, when low, will force the output high. When the gate input goes high, the counter will start from the initial count. Thus, the gate input can be used to synchronize the counter.

When this mode is set, the output will remain high until after the count register is loaded. The output then can also be synchronized by software.

**MODE 3: Square Wave Rate Generator.** Similar to MODE 2 except that the output will remain high until one half the count has been completed (for even numbers) and go low for the other half of the count. This is accomplished by decrementing the counter by two on the falling edge of each clock pulse. When the counter reaches terminal count, the state of the output is changed and the counter is reloaded with the full count and the whole process is repeated.

If the count is odd and the output is high, the first clock pulse (after the count is loaded) decrements the count by 1. Subsequent clock pulses decrement the clock by 2. After timeout, the output goes low and the full count is reloaded. The first clock pulse (following the reload) decrements the counter by 3. Subsequent clock pulses decrement the count by 2 until timeout. Then the whole process is repeated. In this way, if the count is odd, the output will be high for  $(N + 1)/2$  counts and low for  $(N - 1)/2$  counts.

**MODE 4: Software Triggered Strobe.** After the mode is set, the output will be high. When the count is loaded, the counter will begin counting. On terminal count, the output will go low for one input clock period, then will go high again.

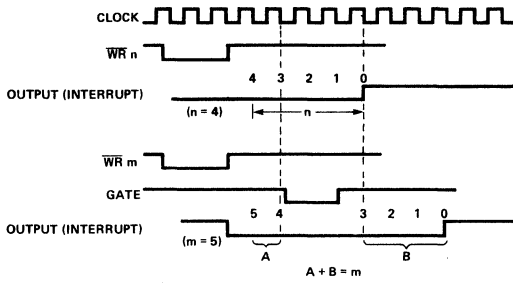
If the count register is reloaded between output pulses, counting will continue from the new value. The count will be inhibited while the gate input is low. Reloading the counter register will restart counting beginning with the new number.

**MODE 5: Hardware Triggered Strobe.** The counter will start counting after the rising edge of the trigger input and will go low for one clock period when the terminal count is reached. The counter is retriggerable. The output will not go low until the full count after the rising edge of any trigger.

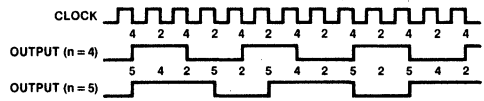
Modes	Signal Status	Low Or Going Low	Rising	High
0		Disables counting	---	Enables counting
1		---	1) Initiates counting 2) Resets output after next clock	---
2		1) Disables counting 2) Sets output immediately high	1) Reloads counter 2) Initiates counting	Enables counting
3		1) Disables counting 2) Sets output immediately high	Initiates counting	Enables counting
4		Disables counting	---	Enables counting
5		---	Initiates counting	---

Figure 6. Gate Pin Operations Summary

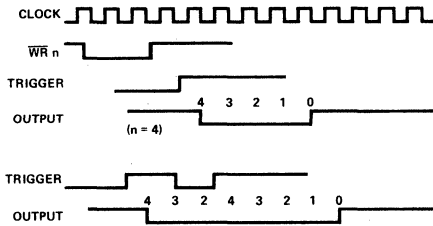
**MODE 0: Interrupt on Terminal Count**



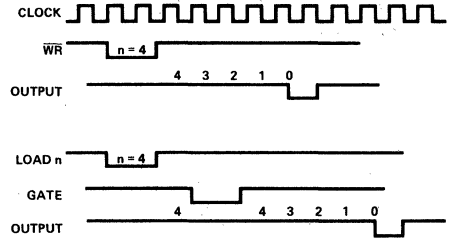
**MODE 3: Square Wave Generator**



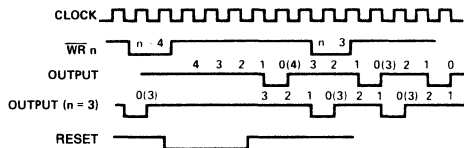
**MODE 1: Programmable One-Shot**



**MODE 4: Software Triggered Strobe**



**MODE 2: Rate Generator**



**MODE 5: Hardware Triggered Strobe**

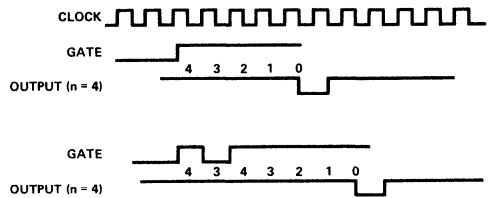


Figure 7. 8253 Timing Diagrams

## 8253 READ/WRITE PROCEDURE

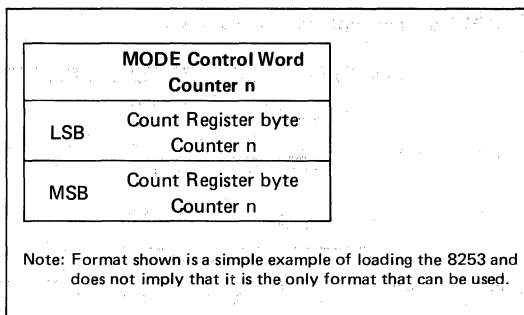
### Write Operations

The systems software must program each counter of the 8253 with the mode and quantity desired. The programmer must write out to the 8253 a MODE control word and the programmed number of count register bytes (1 or 2) prior to actually using the selected counter.

The actual order of the programming is quite flexible. Writing out of the MODE control word can be in any sequence of counter selection, e.g., counter #0 does not have to be first or counter #2 last. Each counter's MODE control word register has a separate address so that its loading is completely sequence independent. (SC0, SC1)

The loading of the Count Register with the actual count value, however, must be done in exactly the sequence programmed in the MODE control word (RL0, RL1). This loading of the counter's count register is still sequence independent like the MODE control word loading, but when a selected count register is to be loaded it must be loaded with the number of bytes programmed in the MODE control word (RL0, RL1). The one or two bytes to be loaded in the count register do not have to follow the associated MODE control word. They can be programmed at any time following the MODE control word loading as long as the correct number of bytes is loaded in order.

All counters are down counters. Thus, the value loaded into the count register will actually be decremented. Loading all zeroes into a count register will result in the maximum count ( $2^{16}$  for Binary or  $10^4$  for BCD). In MODE 0 the new count will not restart until the load has been completed. It will accept one of two bytes depending on how the MODE control words (RL0, RL1) are programmed. Then proceed with the restart operation.



**Figure 8. Programming Format**

		A1	A0
No. 1	MODE Control Word Counter 0	1	1
No. 2	MODE Control Word Counter 1	1	1
No. 3	MODE Control Word Counter 2	1	1
No. 4	LSB     Count Register Byte Counter 1	0	1
No. 5	MSB     Count Register Byte Counter 1	0	1
No. 6	LSB     Count Register Byte Counter 2	1	0
No. 7	MSB     Count Register Byte Counter 2	1	0
No. 8	LSB     Count Register Byte Counter 0	0	0
No. 9	MSB     Count Register Byte Counter 0	0	0

Note: The exclusive addresses of each counter's count register make the task of programming the 8253 a very simple matter, and maximum effective use of the device will result if this feature is fully utilized.

**Figure 9. Alternate Programming Formats**

**Read Operations**

In most counter applications it becomes necessary to read the value of the count in progress and make a computational decision based on this quantity. Event counters are probably the most common application that uses this function. The 8253 contains logic that will allow the programmer to easily read the contents of any of the three counters without disturbing the actual count in progress.

There are two methods that the programmer can use to read the value of the counters. The first method involves the use of simple I/O read operations of the selected counter. By controlling the A0, A1 inputs to the 8253 the programmer can select the counter to be read (remember that no read operation of the mode register is allowed A0, A1-11). The only requirement with this method is that in order to assure a stable count reading the actual operation of the selected counter must be inhibited either by controlling the Gate input or by external logic that inhibits the clock input. The contents of the counter selected will be available as follows:

- first I/O Read contains the least significant byte (LSB).
- second I/O Read contains the most significant byte (MSB).

Due to the internal logic of the 8253 it is absolutely necessary to complete the entire reading procedure. If two bytes are programmed to be read then two bytes must be read before any loading WR command can be sent to the same counter.

**Read Operation Chart**

A1	A0	RD	
0	0	0	Read Counter No. 0
0	1	0	Read Counter No. 1
1	0	0	Read Counter No. 2
1	1	0	Illegal

**Reading While Counting**

In order for the programmer to read the contents of any counter without effecting or disturbing the counting operation the 8253 has special internal logic that can be accessed using simple WR commands to the MODE register. Basically, when the programmer wishes to read the contents of a selected counter "on the fly" he loads the MODE register with a special code which latches the present count value into a storage register so that its contents contain an accurate, stable quantity. The programmer then issues a normal read command to the selected counter and the contents of the latched register is available.

**MODE Register for Latching Count**

A0, A1 = 11

D7	D6	D5	D4	D3	D2	D1	D0
SC1	SC0	0	0	X	X	X	X

- SC1, SC0 — specify counter to be latched.
- D5, D4 — 00 designates counter latching operation.
- X — don't care.

The same limitation applies to this mode of reading the counter as the previous method. That is, it is mandatory to complete the entire read operation as programmed. This command has no effect on the counter's mode.

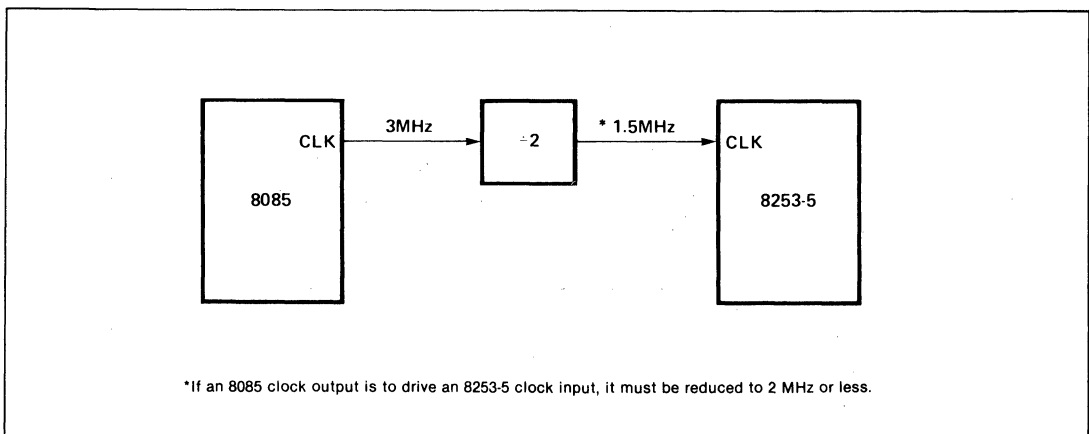


Figure 10. MCS-85™ Clock Interface\*

**ABSOLUTE MAXIMUM RATINGS\***

Ambient Temperature Under Bias	0° C to 70° C
Storage Temperature	-65° C to +150° C
Voltage On Any Pin	
With Respect to Ground	-0.5 V to +7 V
Power Dissipation	1 Watt

*\*NOTICE: Stresses above those listed under "Absolute Maximum Ratings" may cause permanent damage to the device. This is a stress rating only and functional operation of the device at these or any other conditions above those indicated in the operational sections of this specification is not implied. Exposure to absolute maximum rating conditions for extended periods may affect device reliability.*

**D.C. CHARACTERISTICS** ( $T_A = 0^\circ\text{C to } 70^\circ\text{C}$ ,  $V_{CC} = 5\text{V} \pm 10\%$ )

Symbol	Parameter	Min.	Max.	Unit	Test Conditions
$V_{IL}$	Input Low Voltage	-0.5	0.8	V	
$V_{IH}$	Input High Voltage	2.2	$V_{CC} + 5\text{V}$	V	
$V_{OL}$	Output Low Voltage		0.45	V	Note 1
$V_{OH}$	Output High Voltage	2.4		V	Note 2
$I_{IL}$	Input Load Current		$\pm 10$	$\mu\text{A}$	$V_{IN} = V_{CC}$ to 0V
$I_{OFL}$	Output Float Leakage		$\pm 10$	$\mu\text{A}$	$V_{OUT} = V_{CC}$ to 0V
$I_{CC}$	$V_{CC}$ Supply Current		140	mA	

**CAPACITANCE** ( $T_A = 25^\circ\text{C}$ ,  $V_{CC} = \text{GND} = 0\text{V}$ )

Symbol	Parameter	Min.	Typ.	Max.	Unit	Test Conditions
$C_{IN}$	Input Capacitance			10	pF	$f_c = 1\text{ MHz}$
$C_{I/O}$	I/O Capacitance			20	pF	Unmeasured pins returned to $V_{SS}$

**A.C. CHARACTERISTICS** ( $T_A = 0^\circ\text{C to } 70^\circ\text{C}$ ,  $V_{CC} = 5.0\text{V} \pm 5\%$ ,  $\text{GND} = 0\text{V}$ )

**Bus Parameters (Note 3)**
**READ CYCLE**

Symbol	Parameter	8253		8253-5		Unit
		Min.	Max.	Min.	Max.	
$t_{AR}$	Address Stable Before $\overline{\text{READ}}$	50		30		ns
$t_{RA}$	Address Hold Time for $\overline{\text{READ}}$	5		5		ns
$t_{RR}$	$\overline{\text{READ}}$ Pulse Width	400		300		ns
$t_{RD}$	Data Delay From $\overline{\text{READ}}^{[4]}$		300		200	ns
$t_{DF}$	$\overline{\text{READ}}$ to Data Floating	25	125	25	100	ns
$t_{RV}$	Recovery Time Between $\overline{\text{READ}}$ and Any Other Control Signal	1		1		$\mu\text{s}$

**A.C. CHARACTERISTICS (Continued)**

**WRITE CYCLE**

Symbol	Parameter	8253		8253-5		Unit
		Min.	Max.	Min.	Max.	
$t_{AW}$	Address Stable Before $\overline{WRITE}$	50		30		ns
$t_{WA}$	Address Hold Time for $\overline{WRITE}$	30		30		ns
$t_{WW}$	$\overline{WRITE}$ Pulse Width	400		300		ns
$t_{DW}$	Data Set Up Time for $\overline{WRITE}$	300		250		ns
$t_{WD}$	Data Hold Time for $\overline{WRITE}$	40		30		ns
$t_{RV}$	Recovery Time Between $\overline{WRITE}$ and Any Other Control Signal	1		1		$\mu$ s

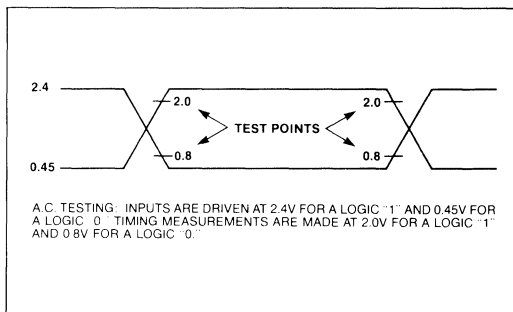
**CLOCK AND GATE TIMING**

Symbol	Parameter	8253		8253-5		Unit
		Min.	Max.	Min.	Max.	
$t_{CLK}$	Clock Period	380	dc	380	dc	ns
$t_{PWH}$	High Pulse Width	230		230		ns
$t_{PWL}$	Low Pulse Width	150		150		ns
$t_{GW}$	Gate Width High	150		150		ns
$t_{GL}$	Gate Width Low	100		100		ns
$t_{GS}$	Gate Set Up Time to $CLK\uparrow$	100		100		ns
$t_{GH}$	Gate Hold Time After $CLK\uparrow$	50		50		ns
$t_{OD}$	Output Delay From $CLK\downarrow$ [4]		400		400	ns
$t_{ODG}$	Output Delay From Gate $\downarrow$ [4]		300		300	ns

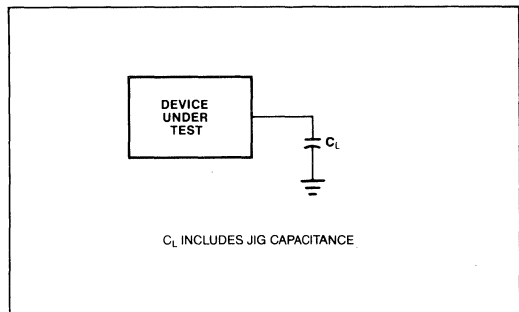
**NOTES:**

1.  $I_{OL} = 2.2$  mA.
2.  $I_{OH} = -400$   $\mu$ A.
3. AC timings measured at  $V_{OH} 2.2$ ,  $V_{OL} = 0.8$ .
4.  $C_L = 150$ pF.

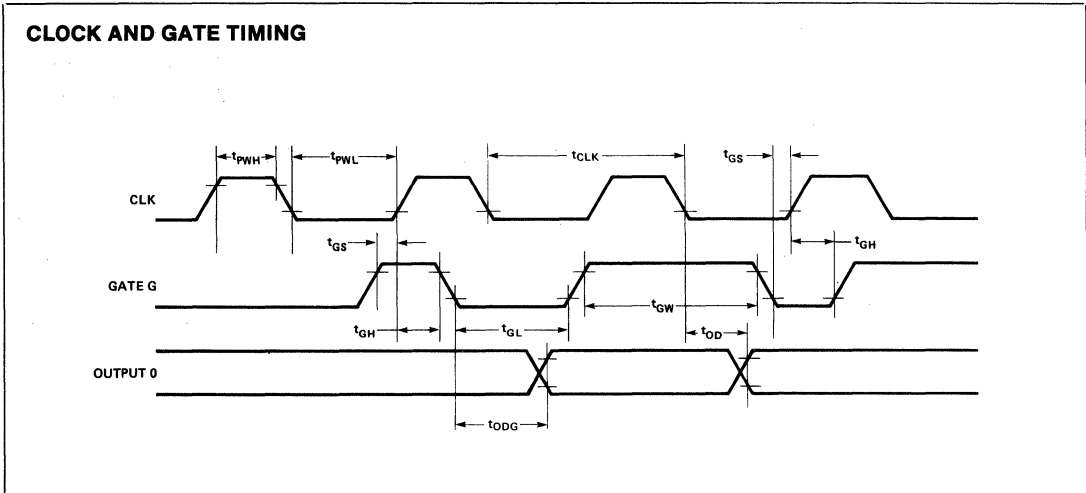
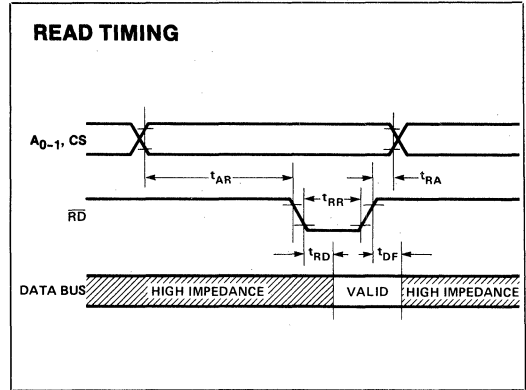
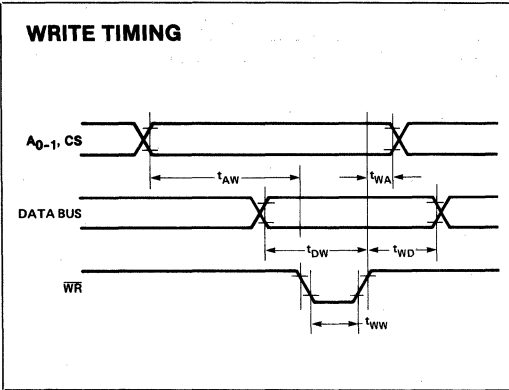
**A.C. TESTING INPUT, OUTPUT WAVEFORM**



**A.C. TESTING LOAD CIRCUIT**



WAVEFORMS



# 8254 Programmable Interval Timer

- Compatible with Most Micro-processors Including 8080A, 8085A, iAPX 88 and iAPX 86
- Three Independent 16-bit Counters
- Handles Inputs from DC to 5 MHz (10 MHz for 8254-2)
- Binary or BCD Counting
- Six Programmable Counter Modes
- Single +5V Supply
- Status Read-Back Command
- Uses HMOS Technology

The Intel® 8254 is a counter/timer device designed to solve the common timing control problems in microcomputer system design. It provides three independent 16-bit counters, each capable of handling clock inputs up to 10 MHz. All modes are software programmable.

The 8254 uses HMOS technology and comes in a 24-pin plastic or CERDIP package.

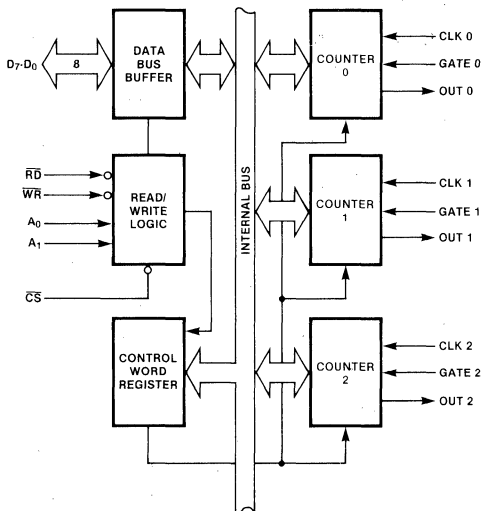


Figure 1. 8254 Block Diagram

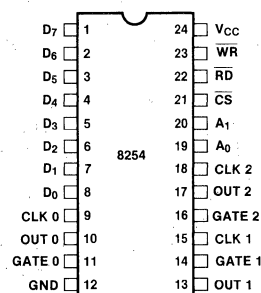


Figure 2. Pin Configuration



Table 1. Pin Description

Symbol	Pin No.	Type	Name and Function
D <sub>7</sub> -D <sub>0</sub>	1-8	I/O	<b>Data:</b> Bi-directional three state data bus lines, connected to system data bus.
CLK 0	9	I	<b>Clock 0:</b> Clock input of Counter 0.
OUT 0	10	O	<b>Output 0:</b> Output of Counter 0.
GATE 0	11	I	<b>Gate 0:</b> Gate input of Counter 0.
GND	12		<b>Ground:</b> Power supply connection.

Symbol	Pin No.	Type	Name and Function															
V <sub>CC</sub>	24		<b>Power:</b> +5V power supply connection.															
WR	23	I	<b>Write Control:</b> This input is low during CPU write operations.															
RD	22	I	<b>Read Control:</b> This input is low during CPU read operations.															
CS	21	I	<b>Chip Select:</b> A low on this input enables the 8254 to respond to RD and WR signals. RD and WR are ignored otherwise.															
A <sub>1</sub> , A <sub>0</sub>	20-19	I	<b>Address:</b> Used to select one of the three Counters or the Control Word Register for read or write operations. Normally connected to the system address bus.															
			<table border="1"> <thead> <tr> <th>A<sub>1</sub></th> <th>A<sub>0</sub></th> <th>Selects</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>0</td> <td>Counter 0</td> </tr> <tr> <td>0</td> <td>1</td> <td>Counter 1</td> </tr> <tr> <td>1</td> <td>0</td> <td>Counter 2</td> </tr> <tr> <td>1</td> <td>1</td> <td>Control Word Register</td> </tr> </tbody> </table>	A <sub>1</sub>	A <sub>0</sub>	Selects	0	0	Counter 0	0	1	Counter 1	1	0	Counter 2	1	1	Control Word Register
A <sub>1</sub>	A <sub>0</sub>	Selects																
0	0	Counter 0																
0	1	Counter 1																
1	0	Counter 2																
1	1	Control Word Register																
CLK 2	18	I	<b>Clock 2:</b> Clock input of Counter 2.															
OUT 2	17	O	<b>Out 2:</b> Output of Counter 2.															
GATE 2	16	I	<b>Gate 2:</b> Gate input of Counter 2.															
CLK 1	15	I	<b>Clock 1:</b> Clock input of Counter 1.															
GATE 1	14	I	<b>Gate 1:</b> Gate input of Counter 1.															
OUT 1	13	O	<b>Out 1:</b> Output of Counter 1.															

FUNCTIONAL DESCRIPTION

General

The 8254 is a programmable interval timer/counter designed for use with Intel microcomputer systems. It is a general purpose, multi-timing element that can be treated as an array of I/O ports in the system software.

The 8254 solves one of the most common problems in any microcomputer system, the generation of accurate time delays under software control. Instead of setting up timing loops in software, the programmer configures the 8254 to match his requirements and programs one of the counters for the desired delay. After the desired delay, the 8254 will interrupt the CPU. Software overhead is minimal and variable length delays can easily be accommodated.

Some of the other counter/timer functions common to microcomputers which can be implemented with the 8254 are:

- Real time clock
- Event counter
- Digital one-shot
- Programmable rate generator
- Square wave generator
- Binary rate multiplier
- Complex waveform generator
- Complex motor controller

Block Diagram

DATA BUS BUFFER

This 3-state, bi-directional, 8-bit buffer is used to interface the 8254 to the system bus (see Figure 3).

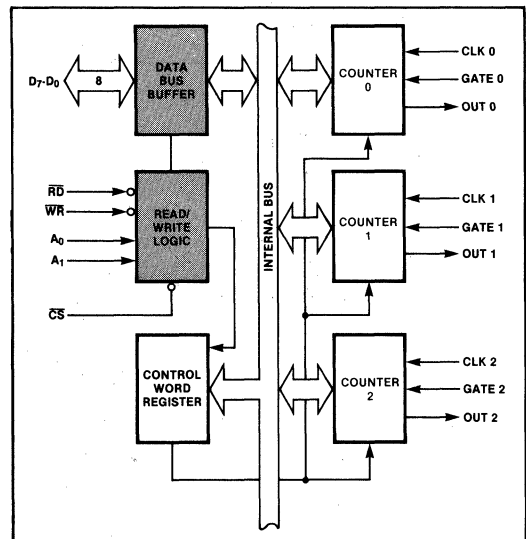


Figure 3. Block Diagram Showing Data Bus Buffer and Read/Write Logic Functions

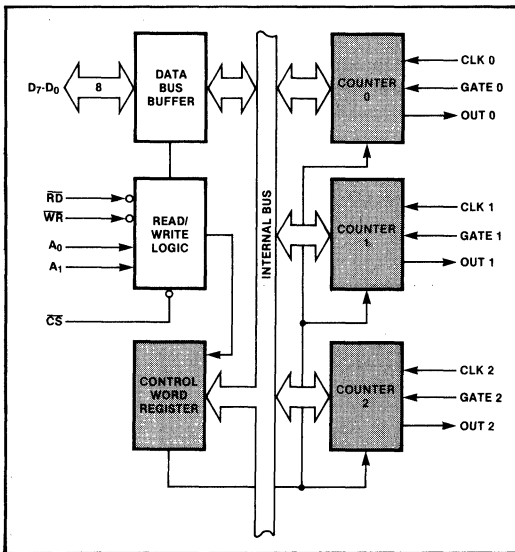
**READ/WRITE LOGIC**

The Read/Write Logic accepts inputs from the system bus and generates control signals for the other functional blocks of the 8254. A<sub>1</sub> and A<sub>0</sub> select one of the three counters or the Control Word Register to be read from/written into. A "low" on the  $\overline{RD}$  input tells the 8254 that the CPU is reading one of the counters. A "low" on the  $\overline{WR}$  input tells the 8254 that the CPU is writing either a Control Word or an initial count. Both  $\overline{RD}$  and  $\overline{WR}$  are qualified by  $\overline{CS}$ ;  $\overline{RD}$  and  $\overline{WR}$  are ignored unless the 8254 has been selected by holding  $\overline{CS}$  low.

**CONTROL WORD REGISTER**

The Control Word Register (see Figure 4) is selected by the Read/Write Logic when A<sub>1</sub>,A<sub>0</sub> = 11. If the CPU then does a write operation to the 8254, the data is stored in the Control Word Register and is interpreted as a Control Word used to define the operation of the Counters.

The Control Word Register can only be written to; status information is available with the Read-Back Command.



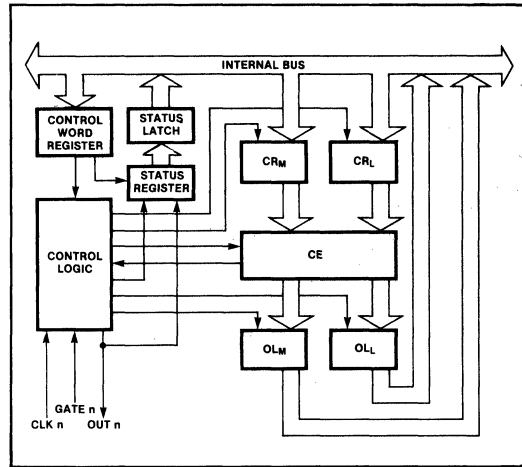
**Figure 4. Block Diagram Showing Control Word Register and Counter Functions**

**COUNTER 0, COUNTER 1, COUNTER 2**

These three functional blocks are identical in operation, so only a single Counter will be described. The internal block diagram of a single counter is shown in Figure 5.

The Counters are fully independent. Each Counter may operate in a different Mode.

The Control Word Register is shown in the figure; it is not part of the Counter itself, but its contents determine how the Counter operates.



**Figure 5. Internal Block Diagram of a Counter**

The status register, shown in the Figure, when latched, contains the current contents of the Control Word Register and status of the output and null count flag. (See detailed explanation of the Read-Back command.)

The actual counter is labelled CE (for "Counting Element"). It is a 16-bit presetable synchronous down counter.

OL<sub>M</sub> and OL<sub>L</sub> are two 8-bit latches. OL stands for "Output Latch"; the subscripts M and L stand for "Most significant byte" and "Least significant byte" respectively. Both are normally referred to as one unit and called just OL. These latches normally "follow" the CE, but if a suitable Counter Latch Command is sent to the 8254, the latches "latch" the present count until read by the CPU and then return to "following" the CE. One latch at a time is enabled by the counter's Control Logic to drive the internal bus. This is how the 16-bit Counter communicates over the 8-bit internal bus. Note that the CE itself cannot be read; whenever you read the count, it is the OL that is being read.

Similarly, there are two 8-bit registers called CR<sub>M</sub> and CR<sub>L</sub> (for "Count Register"). Both are normally referred to as one unit and called just CR. When a new count is written to the Counter, the count is stored in the CR and later transferred to the CE. The Control Logic allows one register at a time to be loaded from the internal bus. Both bytes are transferred to the CE simultaneously. CR<sub>M</sub> and CR<sub>L</sub> are cleared when the Counter is programmed. In this way, if the Counter has been programmed for one byte counts (either most significant byte only or least significant byte only) the other byte will be zero. Note that the CE cannot be written into; whenever a count is written, it is written into the CR.

The Control Logic is also shown in the diagram. CLK<sub>n</sub>, GATE<sub>n</sub>, and OUT<sub>n</sub> are all connected to the outside world through the Control Logic.

**8254 SYSTEM INTERFACE**

The 8254 is a component of the Intel Microcomputer Systems and interfaces in the same manner as all other peripherals of the family. It is treated by the systems software as an array of peripheral I/O ports; three are counters and the fourth is a control register for MODE programming.

Basically, the select inputs  $A_0$ ,  $A_1$  connect to the  $A_0$ ,  $A_1$  address bus signals of the CPU. The CS can be derived directly from the address bus using a linear select method. Or it can be connected to the output of a decoder, such as an Intel 8205 for larger systems.

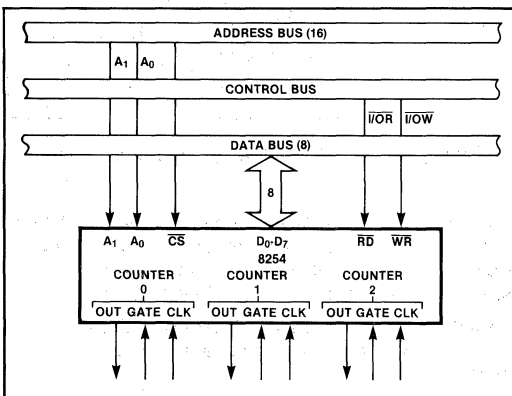


Figure 6. 8254 System Interface

**OPERATIONAL DESCRIPTION**

**General**

After power-up, the state of the 8254 is undefined. The Mode, count value, and output of all Counters are undefined.

How each Counter operates is determined when it is programmed. Each Counter must be programmed before it can be used. Unused counters need not be programmed.

**Programming the 8254**

Counters are programmed by writing a Control Word and then an initial count.

All Control Words are written into the Control Word Register, which is selected when  $A_1, A_0 = 11$ . The Control Word itself specifies which Counter is being programmed.

By contrast, initial counts are written into the Counters, not the Control Word Register. The  $A_1, A_0$  inputs are used to select the Counter to be written into. The format of the initial count is determined by the Control Word used.

**Control Word Format**

$A_1, A_0 = 11$   $\overline{CS} = 0$   $\overline{RD} = 1$   $\overline{WR} = 0$

	$D_7$	$D_6$	$D_5$	$D_4$	$D_3$	$D_2$	$D_1$	$D_0$
	SC1	SC0	RW1	RW0	M2	M1	M0	BCD

**SC — Select Counter:**

SC1	SC0	
0	0	Select Counter 0
0	1	Select Counter 1
1	0	Select Counter 2
1	1	Read-Back Command (See Read Operations)

**RW — Read/Write:**

RW1	RW0	
0	0	Counter Latch Command (see Read Operations)
0	1	Read/Write least significant byte only.
1	0	Read/Write most significant byte only.
1	1	Read/Write least significant byte first, then most significant byte.

**M — MODE:**

M2	M1	M0	
0	0	0	Mode 0
0	0	1	Mode 1
X	1	0	Mode 2
X	1	1	Mode 3
1	0	0	Mode 4
1	0	1	Mode 5

**BCD:**

0	Binary Counter 16-bits
1	Binary Coded Decimal (BCD) Counter (4 Decades)

**NOTE: DON'T CARE BITS (X) SHOULD BE 0 TO INSURE COMPATIBILITY WITH FUTURE INTEL PRODUCTS.**

Figure 7. Control Word Format

**Write Operations**

The programming procedure for the 8254 is very flexible. Only two conventions need to be remembered:

- 1) For each Counter, the Control Word must be written before the initial count is written.
- 2) The initial count must follow the count format specified in the Control Word (least significant byte only, most significant byte only, or least significant byte and then most significant byte).

Since the Control Word Register and the three Counters have separate addresses (selected by the A<sub>1</sub>,A<sub>0</sub> inputs), and each Control Word specifies the Counter it applies to (SC0,SC1 bits), no special instruction sequence is re-

quired. Any programming sequence that follows the conventions above is acceptable.

A new initial count may be written to a Counter at any time without affecting the Counter's programmed Mode in any way. Counting will be affected as described in the Mode definitions. The new count must follow the programmed count format.

If a Counter is programmed to read/write two-byte counts, the following precaution applies: A program must not transfer control between writing the first and second byte to another routine which also writes into that same Counter. Otherwise, the Counter will be loaded with an incorrect count.

		A <sub>1</sub>	A <sub>0</sub>			A <sub>1</sub>	A <sub>0</sub>
Control Word — Counter 0		1	1	Control Word — Counter 2		1	1
LSB of count — Counter 0		0	0	Control Word — Counter 1		1	1
MSB of count — Counter 0		0	0	Control Word — Counter 0		1	1
Control Word — Counter 1		1	1	LSB of count — Counter 2		1	0
LSB of count — Counter 1		0	1	MSB of count — Counter 2		1	0
MSB of count — Counter 1		0	1	LSB of count — Counter 1		0	1
Control Word — Counter 2		1	1	MSB of count — Counter 1		0	1
LSB of count — Counter 2		1	0	LSB of count — Counter 0		0	0
MSB of count — Counter 2		1	0	MSB of count — Counter 0		0	0
		A <sub>1</sub>	A <sub>0</sub>			A <sub>1</sub>	A <sub>0</sub>
Control Word — Counter 0		1	1	Control Word — Counter 1		1	1
Control Word — Counter 1		1	1	Control Word — Counter 0		1	1
Control Word — Counter 2		1	1	LSB of count — Counter 1		0	1
LSB of count — Counter 2		1	0	Control Word — Counter 2		1	1
LSB of count — Counter 1		0	1	LSB of count — Counter 0		0	0
LSB of count — Counter 0		0	0	MSB of count — Counter 1		0	1
MSB of count — Counter 0		0	0	LSB of count — Counter 2		1	0
MSB of count — Counter 1		0	1	MSB of count — Counter 0		0	0
MSB of count — Counter 2		1	0	MSB of count — Counter 2		1	0

NOTE: IN ALL FOUR EXAMPLES, ALL COUNTERS ARE PROGRAMMED TO READ/WRITE TWO-BYTE COUNTS. THESE ARE ONLY FOUR OF MANY POSSIBLE PROGRAMMING SEQUENCES.

**Figure 8. A Few Possible Programming Sequences**

**Read Operations**

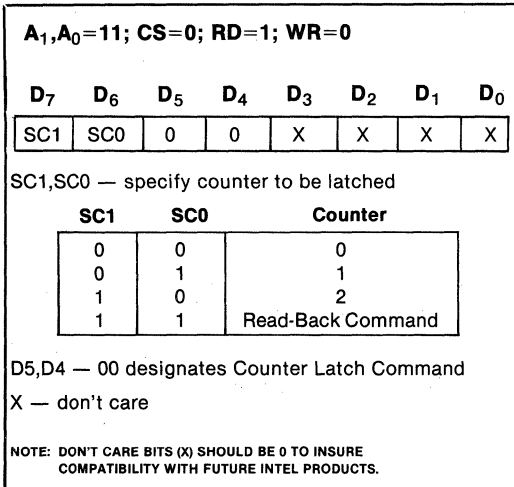
It is often desirable to read the value of a Counter without disturbing the count in progress. This is easily done in the 8254.

There are three possible methods for reading the Counters. The first is through the Read-Back command,

which is explained later. The second is a simple read operation of the Counter, which is selected with the A<sub>1</sub>,A<sub>0</sub> inputs. The only requirement is that the CLK input of the selected Counter must be inhibited by using either the GATE input or external logic. Otherwise, the count may be in process of changing when it is read, giving an undefined result.

**COUNTER LATCH COMMAND**

The other method involves a special software command called the "Counter Latch Command". Like a Control Word, this command is written to the Control Word Register, which is selected when  $A_1, A_0 = 11$ . Also like a Control Word, the SC0, SC1 bits select one of the three Counters, but two other bits, D5 and D4, distinguish this command from a Control Word.



**Figure 9. Counter Latching Command Format**

The selected Counter's output latch (OL) latches the count at the time the Counter Latch Command is received. This count is held in the latch until it is read by the CPU (or until the Counter is reprogrammed). The count is then unlatched automatically and the OL returns to "following" the counting element (CE). This allows reading the contents of the Counters "on the fly" without affecting counting in progress. Multiple Counter Latch Commands may be used to latch more than one Counter. Each latched Counter's OL holds its count until it is read. Counter Latch Commands do not affect the programmed Mode of the Counter in any way.

If a Counter is latched and then, some time later, latched again before the count is read, the second Counter Latch Command is ignored. The count read will be the count at the time the first Counter Latch Command was issued.

With either method, the count must be read according to the programmed format; specifically, if the Counter is programmed for two byte counts, two bytes must be read. The two bytes do not have to be read one right after the other; read or write or programming operations of other Counters may be inserted between them.

Another feature of the 8254 is that reads and writes of the same Counter may be interleaved; for example, if the Counter is programmed for two byte counts, the following sequence is valid.

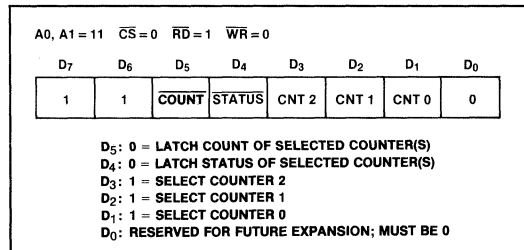
1. Read least significant byte.
2. Write new least significant byte.
3. Read most significant byte.
4. Write new most significant byte.

If a Counter is programmed to read/write two-byte counts, the following precaution applies: A program must not transfer control between reading the first and second byte to another routine which also reads from that same Counter. Otherwise, an incorrect count will be read.

**READ-BACK COMMAND**

The read-back command allows the user to check the count value, programmed Mode, and current state of the OUT pin and Null Count flag of the selected counter(s).

The command is written into the Control Word Register and has the format shown in Figure 10. The command applies to the counters selected by setting their corresponding bits D3, D2, D1=1.



**Figure 10. Read-Back Command Format**

The read-back command may be used to latch multiple counter output latches (OL) by setting the COUNT bit D5=0 and selecting the desired counter(s). This single command is functionally equivalent to several counter latch commands, one for each counter latched. Each counter's latched count is held until it is read (or the counter is reprogrammed). That counter is automatically unlatched when read, but other counters remain latched until they are read. If multiple count read-back commands are issued to the same counter without reading the count, all but the first are ignored; i.e., the count which will be read is the count at the time the first read-back command was issued.

The read-back command may also be used to latch status information of selected counter(s) by setting STATUS bit D4=0. Status must be latched to be read; status of a counter is accessed by a read from that counter.

The counter status format is shown in Figure 11. Bits D5 through D0 contain the counter's programmed Mode exactly as written in the last Mode Control Word. OUTPUT bit D7 contains the current state of the OUT pin. This allows the user to monitor the counter's output via software, possibly eliminating some hardware from a system.

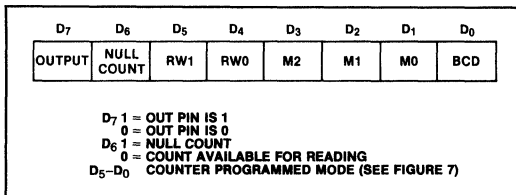


Figure 11. Status Byte

NULL COUNT bit D6 indicates when the last count written to the counter register (CR) has been loaded into the counting element (CE). The exact time this happens depends on the Mode of the counter and is described in the Mode Definitions, but until the count is loaded into the counting element (CE), it can't be read from the counter. If the count is latched or read before this time, the count value will not reflect the new count just written. The operation of Null Count is shown in Figure 12.

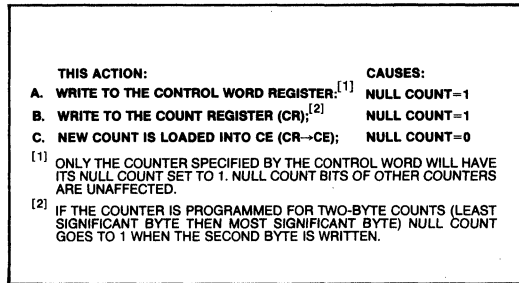


Figure 12. Null Count Operation

If multiple status latch operations of the counter(s) are performed without reading the status, all but the first are ignored; i.e., the status that will be read is the status of the counter at the time the first status read-back command was issued.

Both count and status of the selected counter(s) may be latched simultaneously by setting both **COUNT** and **STATUS** bits D5,D4=0. This is functionally the same as issuing two separate read-back commands at once, and the above discussions apply here also. Specifically, if multiple count and/or status read-back commands are issued to the same counter(s) without any intervening reads, all but the first are ignored. This is illustrated in Figure 13.

Command								Description	Result
D7	D6	D5	D4	D3	D2	D1	D0		
1	1	0	0	0	0	1	0	Read back count and status of Counter 0	Count and status latched for Counter 0
1	1	1	0	0	1	0	0	Read back status of Counter 1	Status latched for Counter 1
1	1	1	0	1	1	0	0	Read back status of Counters 2, 1	Status latched for Counter 2, but not Counter 1
1	1	0	1	1	0	0	0	Read back count of Counter 2	Count latched for Counter 2
1	1	0	0	0	1	0	0	Read back count and status of Counter 1	Count latched for Counter 1, but not status
1	1	1	0	0	0	1	0	Read back status of Counter 1	Command ignored, status already latched for Counter 1

Figure 13. Read-Back Command Example

If both count and status of a counter are latched, the first read operation of that counter will return latched status, regardless of which was latched first. The next one or two reads (depending on whether the counter is programmed for one or two type counts) return latched count. Subsequent reads return unlatched count.

CS	RD	WR	A <sub>1</sub>	A <sub>0</sub>	
0	1	0	0	0	Write into Counter 0
0	1	0	0	1	Write into Counter 1
0	1	0	1	0	Write into Counter 2
0	1	0	1	1	Write Control Word
0	0	1	0	0	Read from Counter 0
0	0	1	0	1	Read from Counter 1
0	0	1	1	0	Read from Counter 2
0	0	1	1	1	No-Operation (3-State)
1	X	X	X	X	No-Operation (3-State)
0	1	1	X	X	No-Operation (3-State)

Figure 14. Read/Write Operations Summary

**Mode Definitions**

The following are defined for use in describing the operation of the 8254.

CLK pulse: a rising edge, then a falling edge, in that order, of a Counter's CLK input.

trigger: a rising edge of a Counter's GATE input.

Counter loading: the transfer of a count from the CR to the CE (refer to the "Functional Description")

**MODE 0: INTERRUPT ON TERMINAL COUNT**

Mode 0 is typically used for event counting. After the Control Word is written, OUT is initially low, and will remain low until the Counter reaches zero. OUT then goes high and remains high until a new count or a new Mode 0 Control Word is written into the Counter.

GATE=1 enables counting; GATE=0 disables counting. GATE has no effect on OUT.

After the Control Word and initial count are written to a Counter, the initial count will be loaded on the next CLK pulse. This CLK pulse does not decrement the count, so for an initial count of N, OUT does not go high until N + 1 CLK pulses after the initial count is written.

If a new count is written to the Counter, it will be loaded on the next CLK pulse and counting will continue from the new count. If a two-byte count is written, the following happens:

- 1) Writing the first byte disables counting. OUT is set low immediately (no clock pulse required)
- 2) Writing the second byte allows the new count to be loaded on the next CLK pulse.

This allows the counting sequence to be synchronized by software. Again, OUT does not go high until N + 1 CLK pulses after the new count of N is written.

If an initial count is written while GATE = 0, it will still be loaded on the next CLK pulse. When GATE goes high, OUT will go high N CLK pulses later; no CLK pulse is needed to load the Counter as this has already been done.

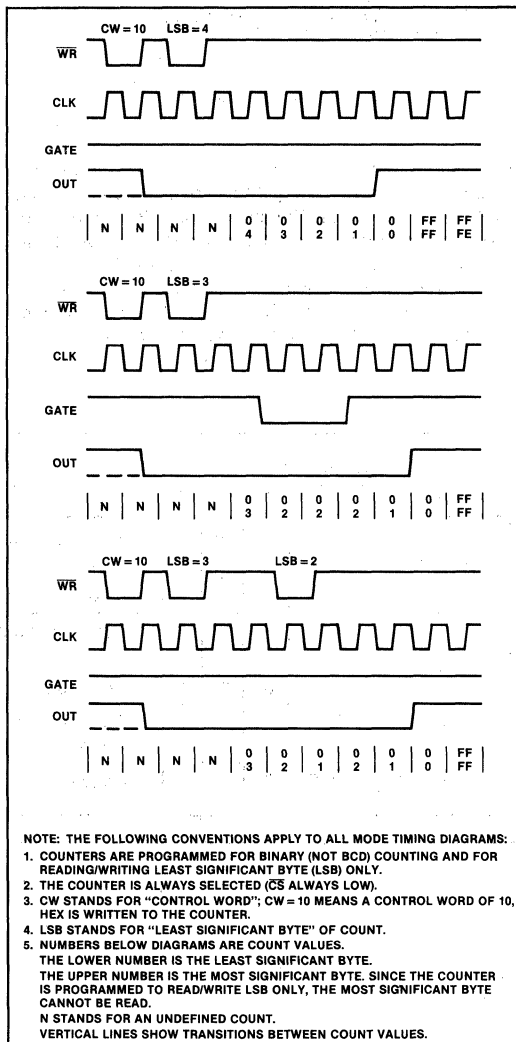


Figure 15. Mode 0

**MODE 1: HARDWARE RETRIGGERABLE ONE-SHOT**

OUT will be initially high. OUT will go low on the CLK pulse following a trigger to begin the one-shot pulse, and will remain low until the Counter reaches zero. OUT will then go high and remain high until the CLK pulse after the next trigger.

After writing the Control Word and initial count, the Counter is armed. A trigger results in loading the Counter and setting OUT low on the next CLK pulse, thus starting the one-shot pulse. An initial count of N will result in a one-shot pulse N CLK cycles in duration. The one-shot is retriggerable, hence OUT will remain low for N CLK pulses after any trigger. The one-shot pulse can be repeated without rewriting the same count into the counter. GATE has no effect on OUT.

If a new count is written to the Counter during a one-shot pulse, the current one-shot is not affected unless the Counter is retriggered. In that case, the Counter is loaded with the new count and the one-shot pulse continues until the new count expires.

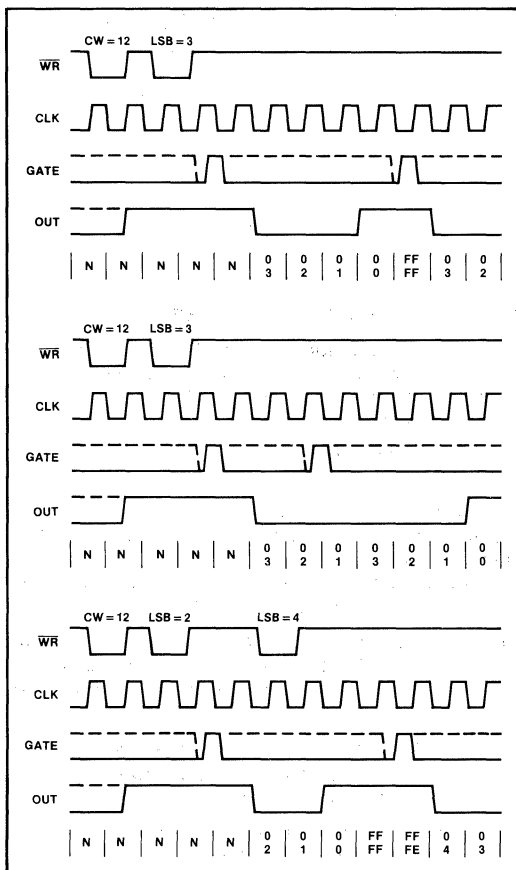


Figure 16. Mode 1

**MODE 2: RATE GENERATOR**

This Mode functions like a divide-by-N counter. It is typically used to generate a Real Time Clock interrupt. OUT will initially be high. When the initial count has decremented to 1, OUT goes low for one CLK pulse. OUT then goes high again, the Counter reloads the initial count and the process is repeated. Mode 2 is periodic; the same sequence is repeated indefinitely. For an initial count of N, the sequence repeats every N CLK cycles.

GATE = 1 enables counting; GATE = 0 disables counting. If GATE goes low during an output pulse, OUT is set high immediately. A trigger reloads the Counter with the initial count on the next CLK pulse; OUT goes low N CLK pulses after the trigger. Thus the GATE input can be used to synchronize the Counter.

After writing a Control Word and initial count, the Counter will be loaded on the next CLK pulse. OUT goes low N CLK Pulses after the initial count is written. This allows the Counter to be synchronized by software also.

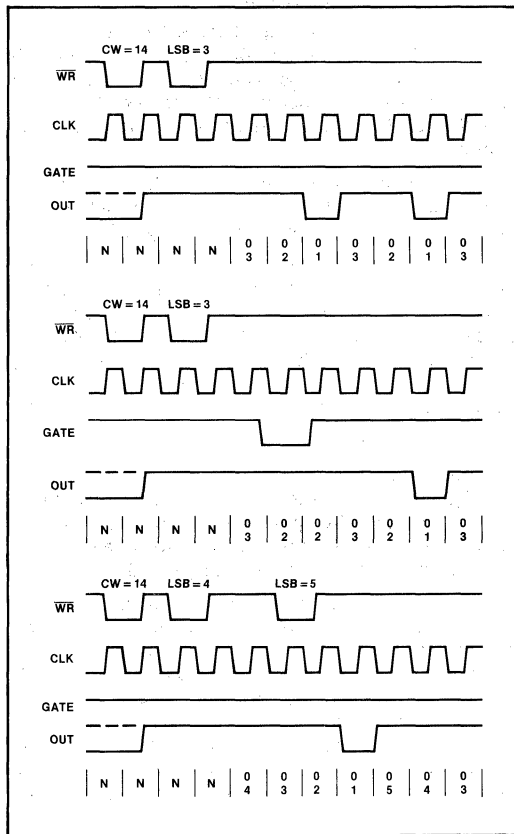


Figure 17. Mode 2



Writing a new count while counting does not affect the current counting sequence. If a trigger is received after writing a new count but before the end of the current period, the Counter will be loaded with the new count on the next CLK pulse and counting will continue from the new count. Otherwise, the new count will be loaded at the end of the current counting cycle.

**MODE 3: SQUARE WAVE MODE**

Mode 3 is typically used for Baud rate generation. Mode 3 is similar to Mode 2 except for the duty cycle of OUT. OUT will initially be high. When half the initial count has expired, OUT goes low for the remainder of the count. Mode 3 is periodic; the sequence above is repeated indefinitely. An initial count of N results in a square wave with a period of N CLK cycles.

GATE = 1 enables counting; GATE = 0 disables counting. If GATE goes low while OUT is low, OUT is set high immediately; no CLK pulse is required. A trigger reloads the Counter with the initial count on the next CLK pulse. Thus the GATE input can be used to synchronize the Counter.

After writing a Control Word and initial count, the Counter will be loaded on the next CLK pulse. This allows the Counter to be synchronized by software also.

Writing a new count while counting does not affect the current counting sequence. If a trigger is received after writing a new count but before the end of the current half-cycle of the square wave, the Counter will be loaded with the new count on the next CLK pulse and counting will continue from the new count. Otherwise, the new count will be loaded at the end of the current half-cycle.

Mode 3 is implemented as follows:

Even counts: OUT is initially high. The initial count is loaded on one CLK pulse and then is decremented by two on succeeding CLK pulses. When the count expires OUT changes value and the Counter is reloaded with the initial count. The above process is repeated indefinitely.

Odd counts: OUT is initially high. The initial count minus one (an even number) is loaded on one CLK pulse and then is decremented by two on succeeding CLK pulses. One CLK pulse after the count expires, OUT goes low and the Counter is reloaded with the initial count minus one. Succeeding CLK pulses decrement the count by two. When the count expires, OUT goes high again and the Counter is reloaded with the initial count minus one. The above process is repeated indefinitely. So for odd counts, OUT will be high for  $(N + 1)/2$  counts and low for  $(N - 1)/2$  counts.

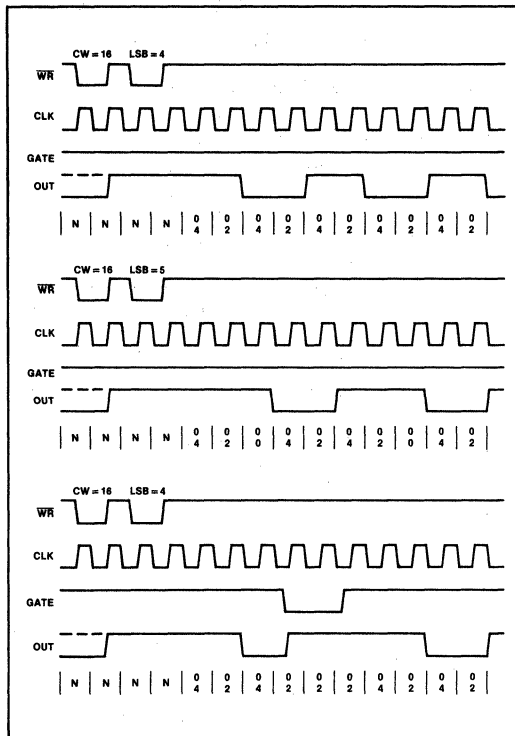


Figure 18. Mode 3

**MODE 4: SOFTWARE TRIGGERED STROBE**

OUT will be initially high. When the initial count expires, OUT will go low for one CLK pulse and then go high again. The counting sequence is "triggered" by writing the initial count.

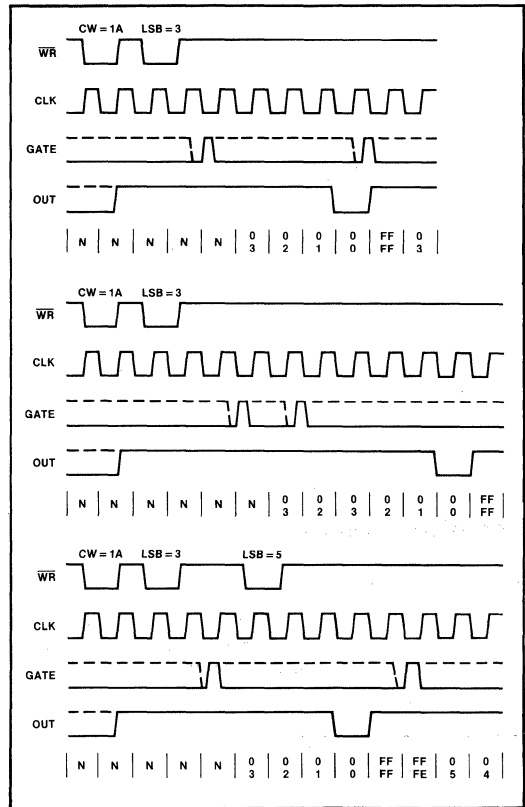
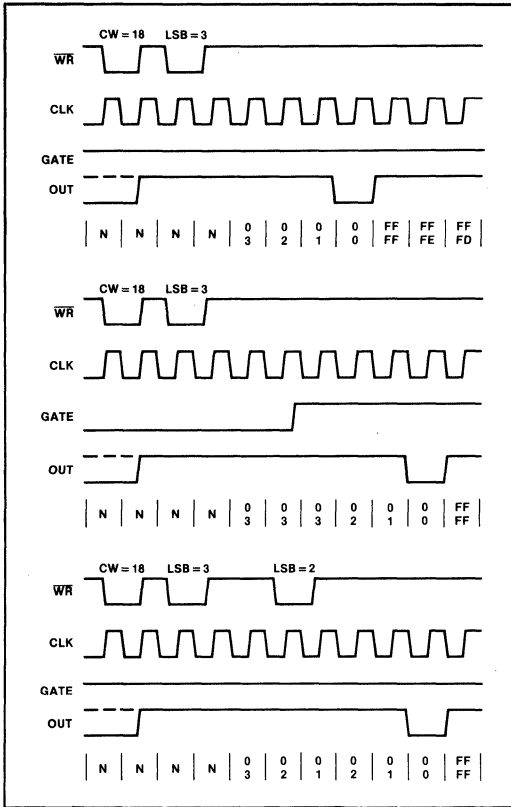
GATE = 1 enables counting; GATE = 0 disables counting. GATE has no effect on OUT.

After writing a Control Word and initial count, the Counter will be loaded on the next CLK pulse. This CLK pulse does not decrement the count, so for an initial count of N, OUT does not strobe low until N + 1 CLK pulses after the initial count is written.

If a new count is written during counting, it will be loaded on the next CLK pulse and counting will continue from the new count. If a two-byte count is written, the following happens:

- 1) Writing the first byte has no effect on counting.
- 2) Writing the second byte allows the new count to be loaded on the next CLK pulse.

This allows the sequence to be "retriggered" by software. OUT strobes low N + 1 CLK pulses after the new count of N is written.



**MODE 5: HARDWARE TRIGGERED STROBE (RETRIGGERABLE)**

OUT will initially be high. Counting is triggered by a rising edge of GATE. When the initial count has expired, OUT will go low for one CLK pulse and then go high again.

After writing the Control Word and initial count, the counter will not be loaded until the CLK pulse after a trigger. This CLK pulse does not decrement the count, so for an initial count of N, OUT does not strobe low until N + 1 CLK pulses after a trigger.

A trigger results in the Counter being loaded with the initial count on the next CLK pulse. The counting sequence is retriggerable. OUT will not strobe low for N + 1 CLK pulses after any trigger. GATE has no effect on OUT.

If a new count is written during counting, the current counting sequence will not be affected. If a trigger occurs after the new count is written but before the current count expires, the Counter will be loaded with the new count on the next CLK pulse and counting will continue from there.

Signal Status Modes	Low Or Going Low	Rising	High
0	Disables counting	---	Enables counting
1	---	1) Initiates counting 2) Resets output after next clock	---
2	1) Disables counting 2) Sets output immediately high	Initiates counting	Enables counting
3	1) Disables counting 2) Sets output immediately high	Initiates counting	Enables counting
4	Disables counting	---	Enables counting
5	---	Initiates counting	---

Figure 21. Gate Pin Operations Summary

Mode	Min Count	Max Count
0	1	0
1	1	0
2	2	0
3	2	0
4	1	0
5	1	0

NOTE: 0 IS EQUIVALENT TO  $2^{16}$  FOR BINARY COUNTING AND  $10^4$  FOR BCD COUNTING.

Figure 22. Minimum and Maximum Initial Counts

## Operation Common to All Modes

### PROGRAMMING

When a Control Word is written to a Counter, all Control Logic is immediately reset and OUT goes to a known initial state; no CLK pulses are required for this.

### GATE

The GATE input is always sampled on the rising edge of CLK. In Modes 0, 2, 3, and 4 the GATE input is level sensitive, and the logic level is sampled on the rising edge of CLK. In Modes 1, 2, 3, and 5 the GATE input is rising-edge sensitive. In these Modes, a rising edge of GATE (trigger) sets an edge-sensitive flip-flop in the Counter. This flip-flop is then sampled on the next rising edge of CLK; the flip-flop is reset immediately after it is sampled. In this way, a trigger will be detected no matter when it occurs—a high logic level does not have to be maintained until the next rising edge of CLK. Note that in Modes 2 and 3, the GATE input is both edge- and level-sensitive.

### COUNTER

New counts are loaded and Counters are decremented on the falling edge of CLK.

The largest possible initial count is 0; this is equivalent to  $2^{16}$  for binary counting and  $10^4$  for BCD counting.

The Counter does not stop when it reaches zero. In Modes 0, 1, 4, and 5 the Counter “wraps around” to the highest count, either FFFF hex for binary counting or 9999 for BCD counting, and continues counting. Modes 2 and 3 are periodic; the Counter reloads itself with the initial count and continues counting from there.

**ABSOLUTE MAXIMUM RATINGS\***

Ambient Temperature Under Bias ..... 0°C to 70°C  
 Storage Temperature ..... -65°C to +150°C  
 Voltage on Any Pin with  
     Respect to Ground ..... -0.5V to +7V  
 Power Dissipation ..... 1 Watt

*\*NOTICE: Stresses above those listed under "Absolute Maximum Ratings" may cause permanent damage to the device. This is a stress rating only and functional operation of the device at these or any other conditions above those indicated in the operational sections of this specification is not implied. Exposure to Absolute Maximum Rating conditions for extended periods may affect device reliability.*

**D.C. CHARACTERISTICS** ( $T_A = 0^\circ\text{C}$  to  $70^\circ\text{C}$ ,  $V_{CC} = 5V \pm 10\%$ )

Symbol	Parameter	Min.	Max.	Units	Test Conditions
$V_{IL}$	Input Low Voltage	-0.5	0.8	V	
$V_{IH}$	Input High Voltage	2.0	$V_{CC} + 0.5V$	V	
$V_{OL}$	Output Low Voltage		0.45	V	$I_{OL} = 2.0\text{ mA}$
$V_{OH}$	Output High Voltage	2.4		V	$I_{OH} = -400\mu\text{A}$
$I_{IL}$	Input Load Current		$\pm 10$	$\mu\text{A}$	$V_{IN} = V_{CC}$ to 0V
$I_{OFL}$	Output Float Leakage		$\pm 10$	$\mu\text{A}$	$V_{OUT} = V_{CC}$ to 0V
$I_{CC}$	$V_{CC}$ Supply Current		140	mA	

**CAPACITANCE** ( $T_A = 25^\circ\text{C}$ ,  $V_{CC} = \text{GND} = 0V$ )

Symbol	Parameter	Min.	Max.	Units	Test Conditions
$C_{IN}$	Input Capacitance		10	pF	$f_c = 1\text{ MHz}$
$C_{I/O}$	I/O Capacitance		20	pF	Unmeasured pins returned to $V_{SS}$

**A.C. CHARACTERISTICS** ( $T_A = 0^\circ\text{C}$  to  $70^\circ\text{C}$ ,  $V_{CC} = 5V \pm 10\%$ ,  $\text{GND} = 0V$ )

**Bus Parameters** (Note 1)

**READ CYCLE**

Symbol	Parameter	8254		8254-2		Unit
		Min.	Max.	Min.	Max.	
$t_{AR}$	Address Stable Before $\overline{RD}\downarrow$	30		25		ns
$t_{SR}$	$\overline{CS}$ Stable Before $\overline{RD}\downarrow$	0		0		ns
$t_{RA}$	Address Hold Time After $\overline{RD}\uparrow$	0		0		ns
$t_{RR}$	$\overline{RD}$ Pulse Width	150		95		ns
$t_{RD}$	Data Delay from $\overline{RD}\downarrow^{[2]}$		120		70	ns
$t_{DF}$	$\overline{RD}\uparrow$ to Data Floating	5	90	5	65	ns
$t_{RV}$	Command Recovery Time	200		95		ns

**Note 1:** AC timings measured at  $V_{OH} = 2.0V$ ,  $V_{OL} = 0.8V$ .

**Note 2:** Test Conditions:  $C_L = 150\text{ pF}$ .

**A.C. CHARACTERISTICS (Continued)**

**WRITE CYCLE**

Symbol	Parameter	8254		8254-2		Unit
		Min.	Max.	Min.	Max.	
t <sub>AW</sub>	Address Stable Before $\overline{WR}\downarrow$	0		0		ns
t <sub>SW</sub>	$\overline{CS}$ Stable Before $\overline{WR}\downarrow$	0		0		ns
t <sub>WA</sub>	Address Hold Time $\overline{WR}\uparrow$	0		0		ns
t <sub>WW</sub>	$\overline{WR}$ Pulse Width	150		95		ns
t <sub>DW</sub>	Data Setup Time Before $\overline{WR}\uparrow$	100		95		ns
t <sub>WD</sub>	Data Hold Time After $\overline{WR}\uparrow$	0		0		ns
t <sub>RV</sub>	Command Recovery Time	200		95		ns

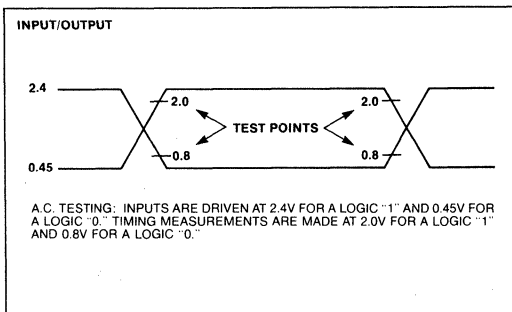
**CLOCK AND GATE** ( $T_A = 0^\circ\text{C}$  to  $70^\circ\text{C}$ ,  $V_{CC} = 5\text{V} \pm 10\%$ ,  $\text{GND} = 0\text{V}$ )

Symbol	Parameter	8254		8254-2		Unit
		Min.	Max.	Min.	Max.	
t <sub>CLK</sub>	Clock Period	200	DC	100	DC	ns
t <sub>PWH</sub> <sup>1</sup>	High Pulse Width	60		30		ns
t <sub>PWL</sub>	Low Pulse Width	60		40		ns
t <sub>R</sub>	Clock Rise Time		100		100	ns
t <sub>F</sub>	Clock Fall Time		100		100	ns
t <sub>GW</sub>	Gate Width High	50		50		ns
t <sub>GL</sub>	Gate Width Low	50		50		ns
t <sub>GS</sub> <sup>1</sup>	Gate Setup Time to CLK $\uparrow$	50		40		ns
t <sub>GH</sub>	Gate Hold Time After CLK $\uparrow$	50		50		ns
t <sub>OD</sub>	Output Delay from CLK $\downarrow$		150		100	ns
t <sub>ODG</sub>	Output Delay from Gate $\downarrow$		120		90	ns

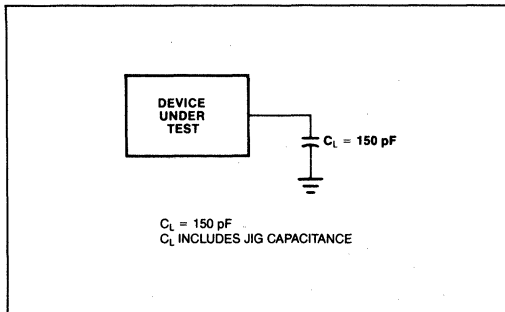
**NOTES:**

1. If the gate input is used asynchronously t<sub>PWH</sub> = 50 ns, t<sub>GS</sub> = 50 ns for 8254-2.

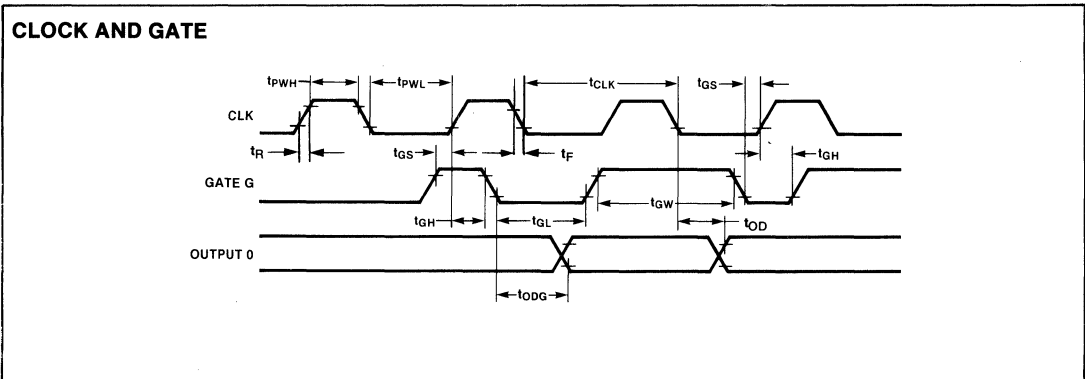
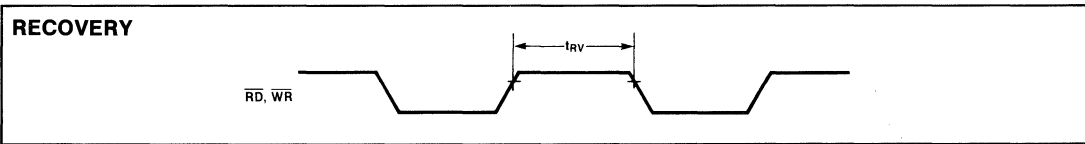
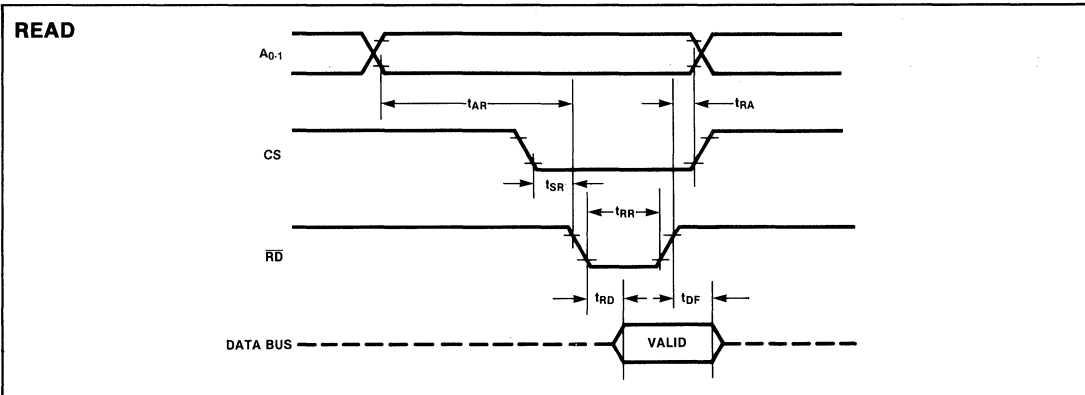
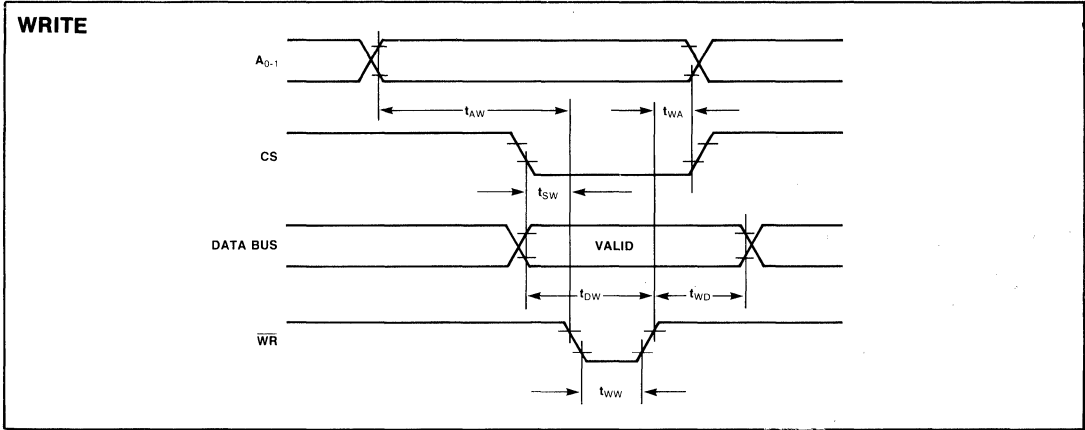
**A.C. TESTING INPUT, OUTPUT WAVEFORM**



**A.C. TESTING LOAD CIRCUIT**



WAVEFORMS





## 8255A/8255A-5 PROGRAMMABLE PERIPHERAL INTERFACE

- MCS-85™ Compatible 8255A-5
- 24 Programmable I/O Pins
- Completely TTL Compatible
- Fully Compatible with Intel® Micro-processor Families
- Improved Timing Characteristics
- Direct Bit Set/Reset Capability Easing Control Application Interface
- 40-Pin Dual In-Line Package
- Reduces System Package Count
- Improved DC Driving Capability

The Intel® 8255A is a general purpose programmable I/O device designed for use with Intel® microprocessors. It has 24 I/O pins which may be individually programmed in 2 groups of 12 and used in 3 major modes of operation. In the first mode (MODE 0), each group of 12 I/O pins may be programmed in sets of 4 to be input or output. In MODE 1, the second mode, each group may be programmed to have 8 lines of input or output. Of the remaining 4 pins, 3 are used for handshaking and interrupt control signals. The third mode of operation (MODE 2) is a bidirectional bus mode which uses 8 lines for a bidirectional bus, and 5 lines, borrowing one from the other group, for handshaking.

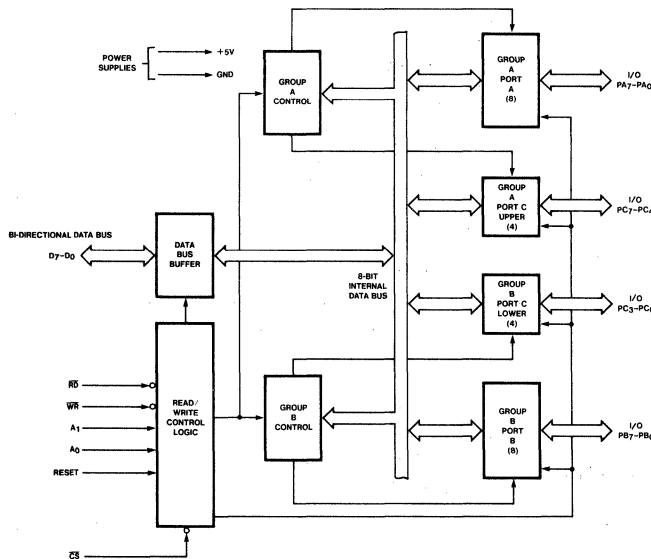


Figure 1. 8255A Block Diagram

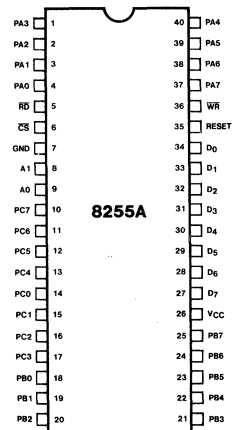


Figure 2. Pin Configuration

## 8255A FUNCTIONAL DESCRIPTION

### General

The 8255A is a programmable peripheral interface (PPI) device designed for use in Intel® microcomputer systems. Its function is that of a general purpose I/O component to interface peripheral equipment to the microcomputer system bus. The functional configuration of the 8255A is programmed by the system software so that normally no external logic is necessary to interface peripheral devices or structures.

### Data Bus Buffer

This 3-state bidirectional 8-bit buffer is used to interface the 8255A to the system data bus. Data is transmitted or received by the buffer upon execution of input or output instructions by the CPU. Control words and status information are also transferred through the data bus buffer.

### Read/Write and Control Logic

The function of this block is to manage all of the internal and external transfers of both Data and Control or Status words. It accepts inputs from the CPU Address and Control busses and in turn, issues commands to both of the Control Groups.

### (CS)

**Chip Select.** A "low" on this input pin enables the communication between the 8255A and the CPU.

### (RD)

**Read.** A "low" on this input pin enables the 8255A to send the data or status information to the CPU on the data bus. In essence, it allows the CPU to "read from" the 8255A.

### (WR)

**Write.** A "low" on this input pin enables the CPU to write data or control words into the 8255A.

### (A<sub>0</sub> and A<sub>1</sub>)

**Port Select 0 and Port Select 1.** These input signals, in conjunction with the RD and WR inputs, control the selection of one of the three ports or the control word registers. They are normally connected to the least significant bits of the address bus (A<sub>0</sub> and A<sub>1</sub>).

## 8255A BASIC OPERATION

A <sub>1</sub>	A <sub>0</sub>	RD	WR	CS	INPUT OPERATION (READ)
0	0	0	1	0	PORT A ⇒ DATA BUS
0	1	0	1	0	PORT B ⇒ DATA BUS
1	0	0	1	0	PORT C ⇒ DATA BUS
					<b>OUTPUT OPERATION (WRITE)</b>
0	0	1	0	0	DATA BUS ⇒ PORT A
0	1	1	0	0	DATA BUS ⇒ PORT B
1	0	1	0	0	DATA BUS ⇒ PORT C
1	1	1	0	0	DATA BUS ⇒ CONTROL
					<b>DISABLE FUNCTION</b>
X	X	X	X	1	DATA BUS ⇒ 3-STATE
1	1	0	1	0	ILLEGAL CONDITION
X	X	1	1	0	DATA BUS ⇒ 3-STATE

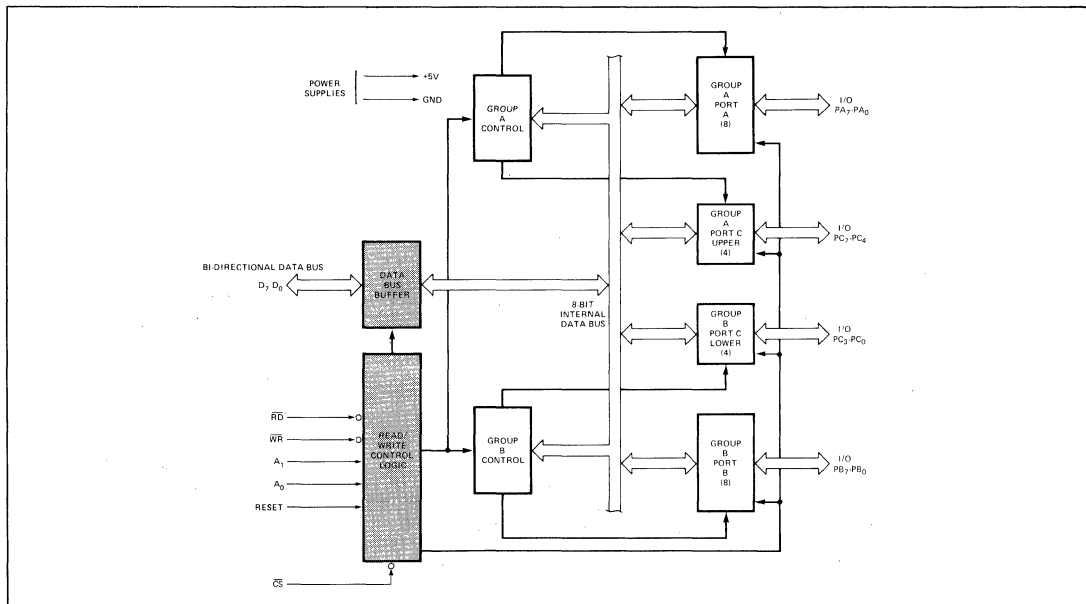


Figure 3. 8255A Block Diagram Showing Data Bus Buffer and Read/Write Control Logic Functions



**(RESET)**

**Reset.** A "high on this input clears the control register and all ports (A, C, C) are set to the input mode.

**Group A and Group B Controls**

The functional configuration of each port is programmed by the systems software. In essence, the CPU "outputs" a control word to the 8255A. The control word contains information such as "mode", "bit set", "bit reset", etc., that initializes the functional configuration of the 8255A.

Each of the Control blocks (Group A and Group B) accepts "commands" from the Read/Write Control Logic, receives "control words" from the internal data bus and issues the proper commands to its associated ports.

Control Group A – Port A and Port C upper (C7-C4)

Control Group B – Port B and Port C lower (C3-C0)

The Control Word Register can Only be written into. No Read operation of the Control Word Register is allowed.

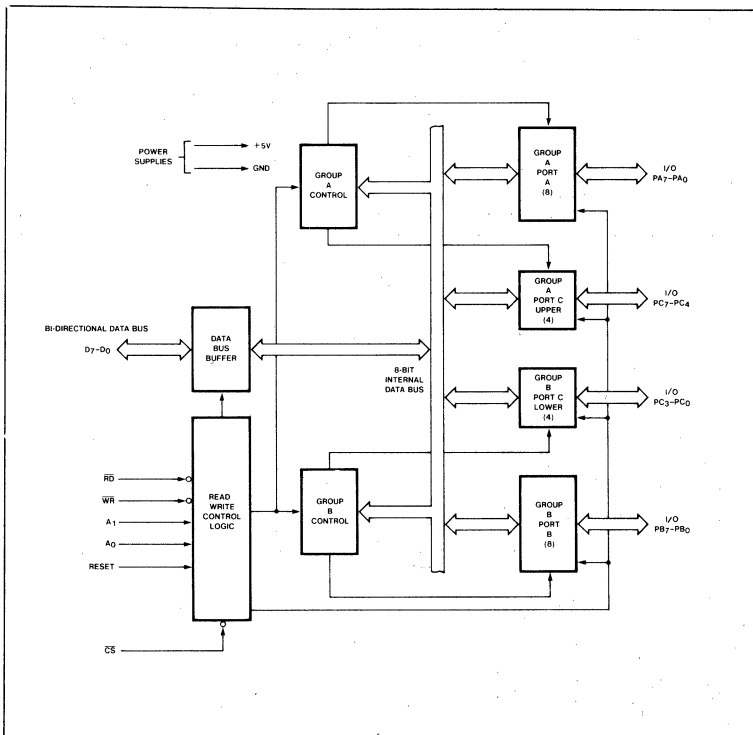
**Ports A, B, and C**

The 8255A contains three 8-bit ports (A, B, and C). All can be configured in a wide variety of functional characteristics by the system software but each has its own special features or "personality" to further enhance the power and flexibility of the 8255A.

**Port A.** One 8-bit data output latch/buffer and one 8-bit data input latch.

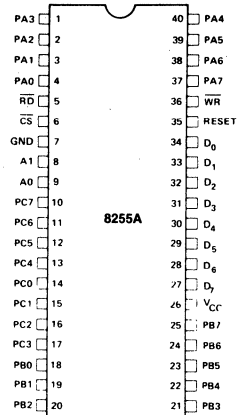
**Port B.** One 8-bit data input/output latch/buffer and one 8-bit data input buffer.

**Port C.** One 8-bit data output latch/buffer and one 8-bit data input buffer (no latch for input). This port can be divided into two 4-bit ports under the mode control. Each 4-bit port contains a 4-bit latch and it can be used for the control signal outputs and status signal inputs in conjunction with ports A and B.



**Figure 4. 8255A Block Diagram Showing Group A and Group B Control Functions**

**PIN CONFIGURATION**



**PIN NAMES**

D <sub>7</sub> D <sub>0</sub>	DATA BUS (BI-DIRECTIONAL)
RESET	RESET INPUT
CS	CHIP SELECT
RD	READ INPUT
WR	WRITE INPUT
A <sub>0</sub> , A <sub>1</sub>	PORT ADDRESS
PA <sub>7</sub> -PA <sub>0</sub>	PORT A (BIT)
PB <sub>7</sub> -PB <sub>0</sub>	PORT B (BIT)
PC <sub>7</sub> -PC <sub>0</sub>	PORT C (BIT)
V <sub>cc</sub>	+5 VOLTS
GND	0 VOLTS

## 8255A OPERATIONAL DESCRIPTION

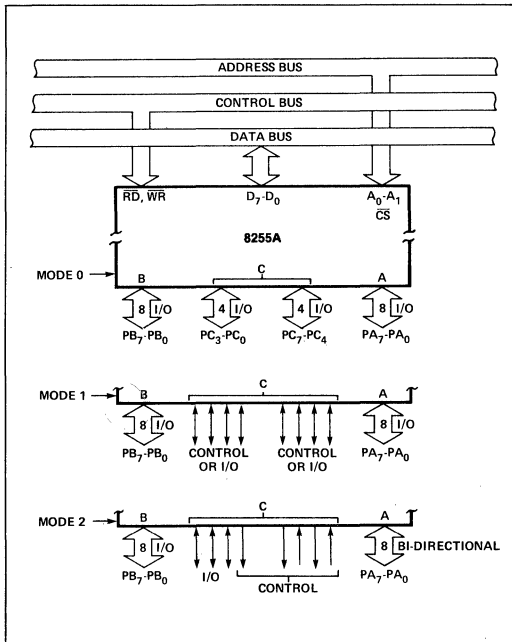
### Mode Selection

There are three basic modes of operation that can be selected by the system software:

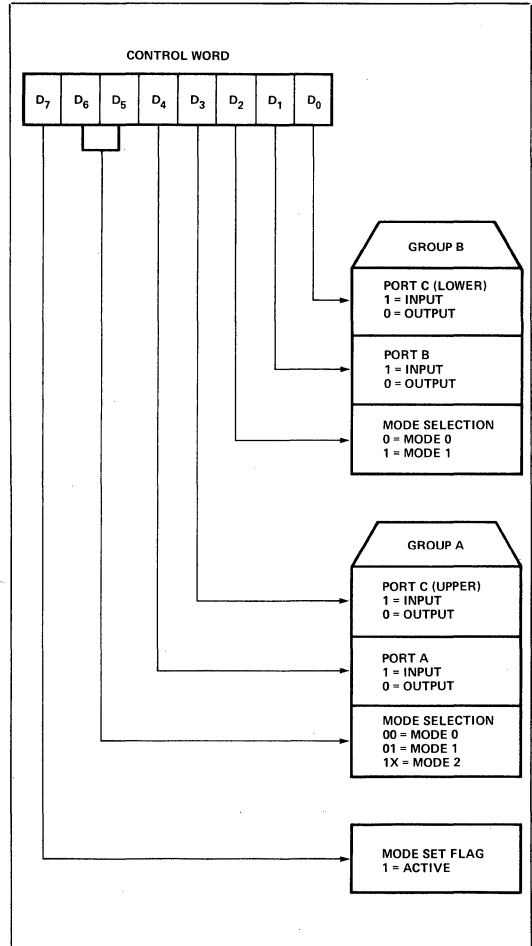
- Mode 0 – Basic Input/Output
- Mode 1 – Strobed Input/Output
- Mode 2 – Bi-Directional Bus

When the reset input goes “high” all ports will be set to the input mode (i.e., all 24 lines will be in the high impedance state). After the reset is removed the 8255A can remain in the input mode with no additional initialization required. During the execution of the system program any of the other modes may be selected using a single output instruction. This allows a single 8255A to service a variety of peripheral devices with a simple software maintenance routine.

The modes for Port A and Port B can be separately defined, while Port C is divided into two portions as required by the Port A and Port B definitions. All of the output registers, including the status flip-flops, will be reset whenever the mode is changed. Modes may be combined so that their functional definition can be “tailored” to almost any I/O structure. For instance; Group B can be programmed in Mode 0 to monitor simple switch closings or display computational results, Group A could be programmed in Mode 1 to monitor a keyboard or tape reader on an interrupt-driven basis.



**Figure 5. Basic Mode Definitions and Bus Interface**



**Figure 6. Mode Definition Format**

The mode definitions and possible mode combinations may seem confusing at first but after a cursory review of the complete device operation a simple, logical I/O approach will surface. The design of the 8255A has taken into account things such as efficient PC board layout, control signal definition vs PC layout and complete functional flexibility to support almost any peripheral device with no external logic. Such design represents the maximum use of the available pins.

### Single Bit Set/Reset Feature

Any of the eight bits of Port C can be Set or Reset using a single OUTput instruction. This feature reduces software requirements in Control-based applications.

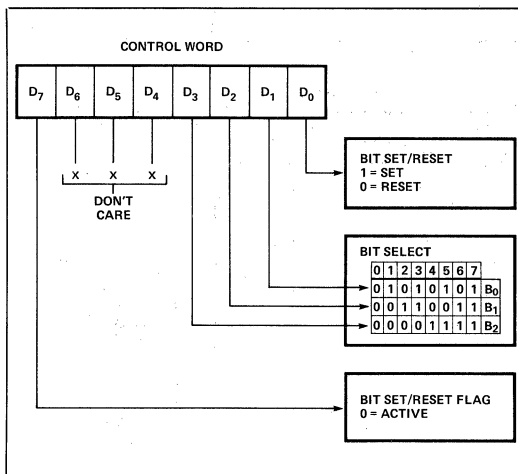


Figure 7. Bit Set/Reset Format

When Port C is being used as status/control for Port A or B, these bits can be set or reset by using the Bit Set/Reset operation just as if they were data output ports.

**Interrupt Control Functions**

When the 8255A is programmed to operate in mode 1 or mode 2, control signals are provided that can be used as interrupt request inputs to the CPU. The interrupt request signals, generated from port C, can be inhibited or enabled by setting or resetting the associated INTE flip-flop, using the bit set/reset function of port C.

This function allows the Programmer to disallow or allow a specific I/O device to interrupt the CPU without affecting any other device in the interrupt structure.

INTE flip-flop definition:

(BIT-SET) – INTE is SET – Interrupt enable

(BIT-RESET) – INTE is RESET – Interrupt disable

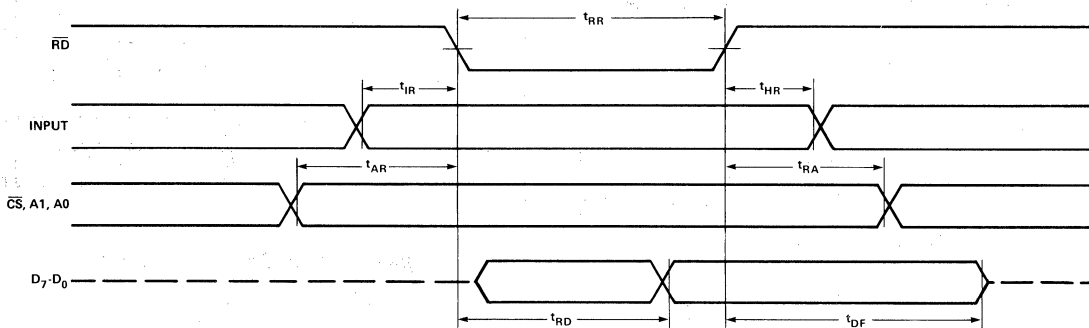
Note: All Mask flip-flops are automatically reset during mode selection and device Reset.

**Operating Modes**

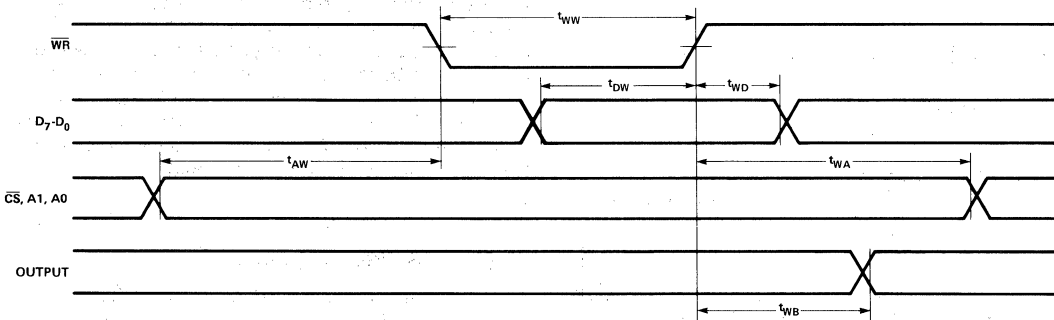
**MODE 0 (Basic Input/Output).** This functional configuration provides simple input and output operations for each of the three ports. No “handshaking” is required, data is simply written to or read from a specified port.

Mode 0 Basic Functional Definitions:

- Two 8-bit ports and two 4-bit ports.
- Any port can be input or output.
- Outputs are latched.
- Inputs are not latched.
- 16 different Input/Output configurations are possible in this Mode.



**MODE 0 (Basic Input)**



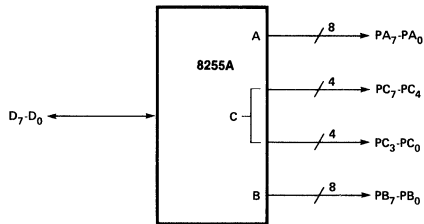
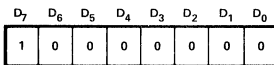
**MODE 0 (Basic Output)**

MODE 0 Port Definition

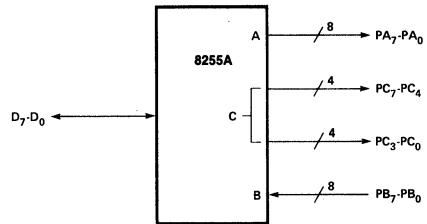
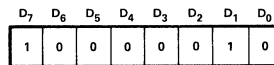
A		B		GROUP A			GROUP B		
D <sub>4</sub>	D <sub>3</sub>	D <sub>1</sub>	D <sub>0</sub>	PORT A	PORT C (UPPER)	#	PORT B	PORT C (LOWER)	
0	0	0	0	OUTPUT	OUTPUT	0	OUTPUT	OUTPUT	
0	0	0	1	OUTPUT	OUTPUT	1	OUTPUT	INPUT	
0	0	1	0	OUTPUT	OUTPUT	2	INPUT	OUTPUT	
0	0	1	1	OUTPUT	OUTPUT	3	INPUT	INPUT	
0	1	0	0	OUTPUT	INPUT	4	OUTPUT	OUTPUT	
0	1	0	1	OUTPUT	INPUT	5	OUTPUT	INPUT	
0	1	1	0	OUTPUT	INPUT	6	INPUT	OUTPUT	
0	1	1	1	OUTPUT	INPUT	7	INPUT	INPUT	
1	0	0	0	INPUT	OUTPUT	8	OUTPUT	OUTPUT	
1	0	0	1	INPUT	OUTPUT	9	OUTPUT	INPUT	
1	0	1	0	INPUT	OUTPUT	10	INPUT	OUTPUT	
1	0	1	1	INPUT	OUTPUT	11	INPUT	INPUT	
1	1	0	0	INPUT	INPUT	12	OUTPUT	OUTPUT	
1	1	0	1	INPUT	INPUT	13	OUTPUT	INPUT	
1	1	1	0	INPUT	INPUT	14	INPUT	OUTPUT	
1	1	1	1	INPUT	INPUT	15	INPUT	INPUT	

MODE 0 Configurations

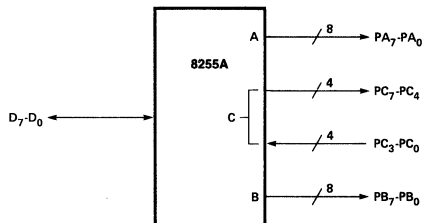
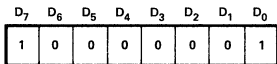
CONTROL WORD #0



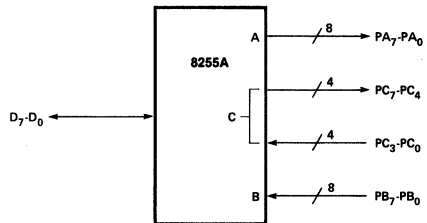
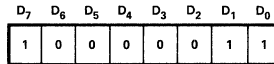
CONTROL WORD #2



CONTROL WORD #1

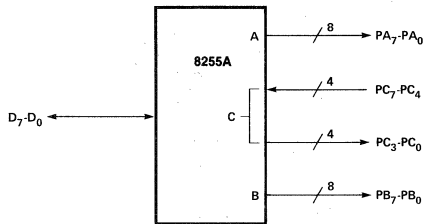


CONTROL WORD #3



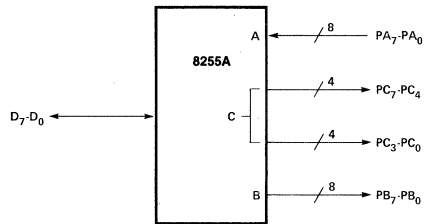
CONTROL WORD #4

D <sub>7</sub>	D <sub>6</sub>	D <sub>5</sub>	D <sub>4</sub>	D <sub>3</sub>	D <sub>2</sub>	D <sub>1</sub>	D <sub>0</sub>
1	0	0	0	1	0	0	0



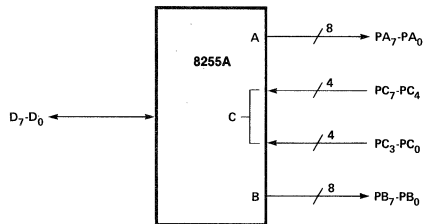
CONTROL WORD #8

D <sub>7</sub>	D <sub>6</sub>	D <sub>5</sub>	D <sub>4</sub>	D <sub>3</sub>	D <sub>2</sub>	D <sub>1</sub>	D <sub>0</sub>
1	0	0	1	0	0	0	0



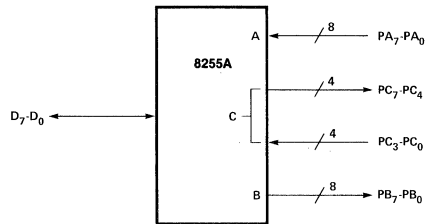
CONTROL WORD #5

D <sub>7</sub>	D <sub>6</sub>	D <sub>5</sub>	D <sub>4</sub>	D <sub>3</sub>	D <sub>2</sub>	D <sub>1</sub>	D <sub>0</sub>
1	0	0	0	1	0	0	1



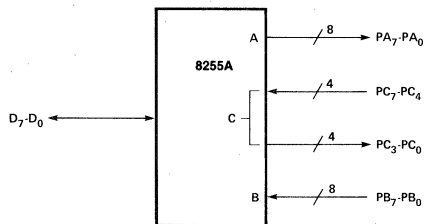
CONTROL WORD #9

D <sub>7</sub>	D <sub>6</sub>	D <sub>5</sub>	D <sub>4</sub>	D <sub>3</sub>	D <sub>2</sub>	D <sub>1</sub>	D <sub>0</sub>
1	0	0	1	0	0	0	1



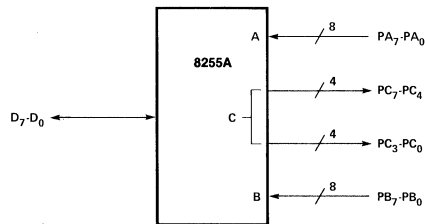
CONTROL WORD #6

D <sub>7</sub>	D <sub>6</sub>	D <sub>5</sub>	D <sub>4</sub>	D <sub>3</sub>	D <sub>2</sub>	D <sub>1</sub>	D <sub>0</sub>
1	0	0	0	1	0	1	0



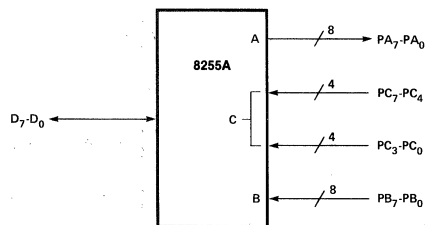
CONTROL WORD #10

D <sub>7</sub>	D <sub>6</sub>	D <sub>5</sub>	D <sub>4</sub>	D <sub>3</sub>	D <sub>2</sub>	D <sub>1</sub>	D <sub>0</sub>
1	0	0	1	0	0	1	0



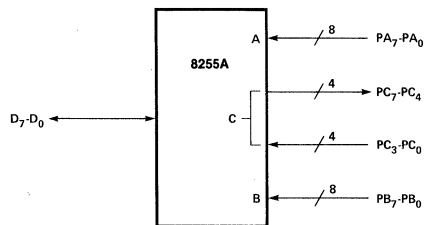
CONTROL WORD #7

D <sub>7</sub>	D <sub>6</sub>	D <sub>5</sub>	D <sub>4</sub>	D <sub>3</sub>	D <sub>2</sub>	D <sub>1</sub>	D <sub>0</sub>
1	0	0	0	1	0	1	1

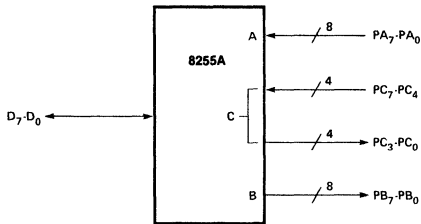
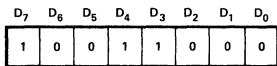


CONTROL WORD #11

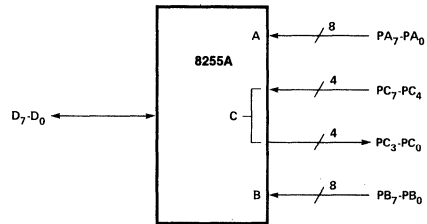
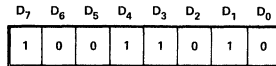
D <sub>7</sub>	D <sub>6</sub>	D <sub>5</sub>	D <sub>4</sub>	D <sub>3</sub>	D <sub>2</sub>	D <sub>1</sub>	D <sub>0</sub>
1	0	0	1	0	0	1	1



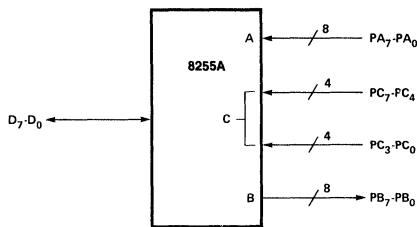
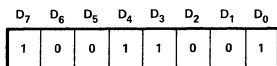
CONTROL WORD #12



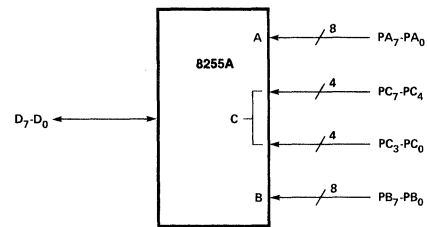
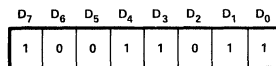
CONTROL WORD #14



CONTROL WORD #13



CONTROL WORD #15



### Operating Modes

**MODE 1 (Strobed Input/Output).** This functional configuration provides a means for transferring I/O data to or from a specified port in conjunction with strobes or "handshaking" signals. In mode 1, port A and Port B use the lines on port C to generate or accept these "handshaking" signals.

#### Mode 1 Basic Functional Definitions:

- Two Groups (Group A and Group B)
- Each group contains one 8-bit data port and one 4-bit control/data port.
- The 8-bit data port can be either input or output. Both inputs and outputs are latched.
- The 4-bit port is used for control and status of the 8-bit data port.

**Input Control Signal Definition**

**STB (Strobe Input).** A "low" on this input loads data into the input latch.

**IBF (Input Buffer Full F/F)**

A "high" on this output indicates that the data has been loaded into the input latch; in essence, an acknowledgement. IBF is set by STB input being low and is reset by the rising edge of the RD input.

**INTR (Interrupt Request)**

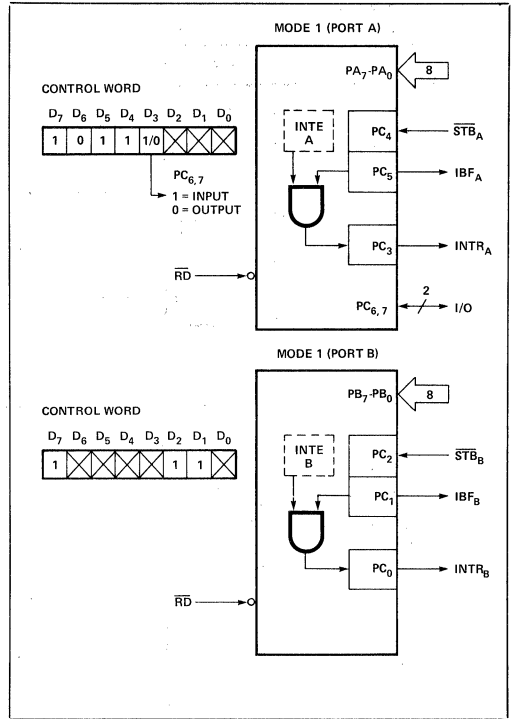
A "high" on this output can be used to interrupt the CPU when an input device is requesting service. INTR is set by the STB is a "one", IBF is a "one" and INTE is a "one". It is reset by the falling edge of RD. This procedure allows an input device to request service from the CPU by simply strobing its data into the port.

**INTE A**

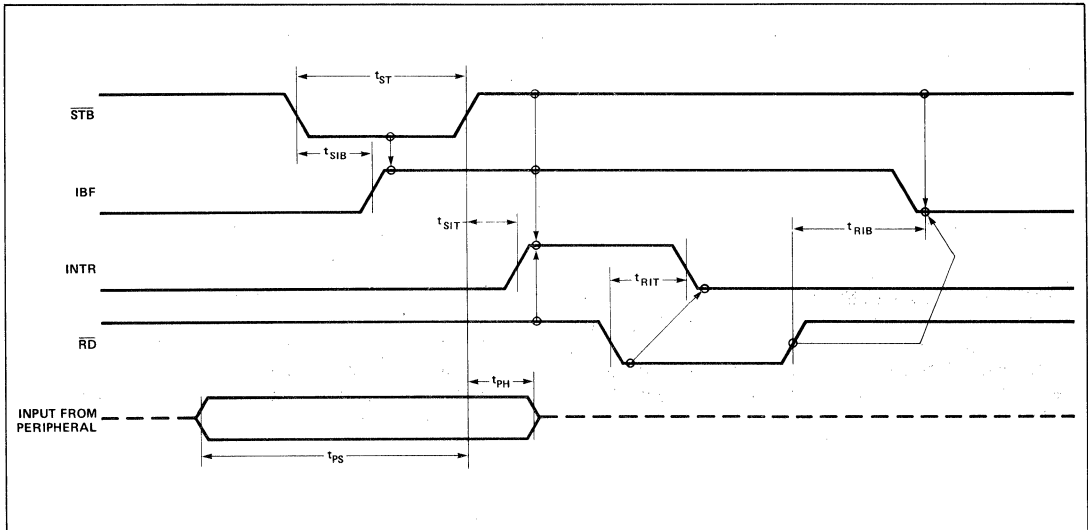
Controlled by bit set/reset of PC<sub>4</sub>.

**INTE B**

Controlled by bit set/reset of PC<sub>2</sub>.



**Figure 8. MODE 1 Input**



**Figure 9. MODE 1 (Strobed Input)**

**Output Control Signal Definition**

**$\overline{\text{OBF}}$  (Output Buffer Full F/F).** The  $\overline{\text{OBF}}$  output will go "low" to indicate that the CPU has written data out to the specified port. The  $\overline{\text{OBF}}$  F/F will be set by the rising edge of the  $\overline{\text{WR}}$  input and reset by  $\overline{\text{ACK}}$  Input being low.

**$\overline{\text{ACK}}$  (Acknowledge Input).** A "low" on this input informs the 8255A that the data from port A or port B has been accepted. In essence, a response from the peripheral device indicating that it has received the data output by the CPU.

**$\text{INTR}$  (Interrupt Request).** A "high" on this output can be used to interrupt the CPU when an output device has accepted data transmitted by the CPU.  $\text{INTR}$  is set when  $\overline{\text{ACK}}$  is a "one",  $\overline{\text{OBF}}$  is a "one" and  $\text{INTE}$  is a "one". It is reset by the falling edge of  $\overline{\text{WR}}$ .

**INTE A**

Controlled by bit set/reset of  $\text{PC}_6$ .

**INTE B**

Controlled by bit set/reset of  $\text{PC}_2$ .

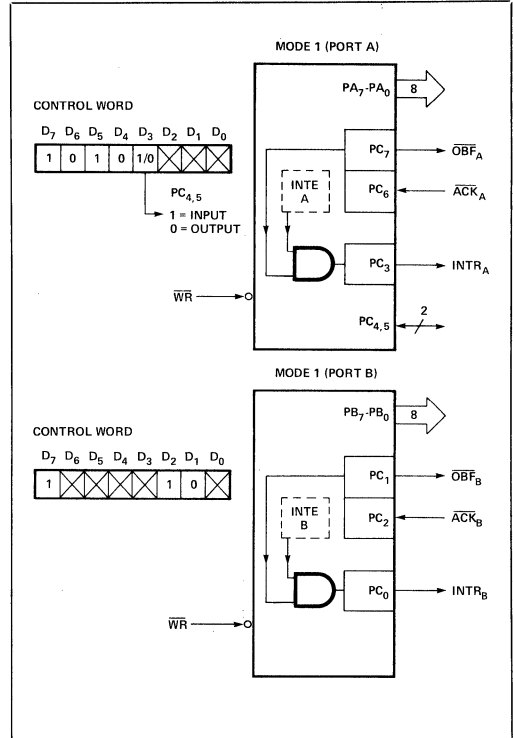


Figure 10. MODE 1 Output

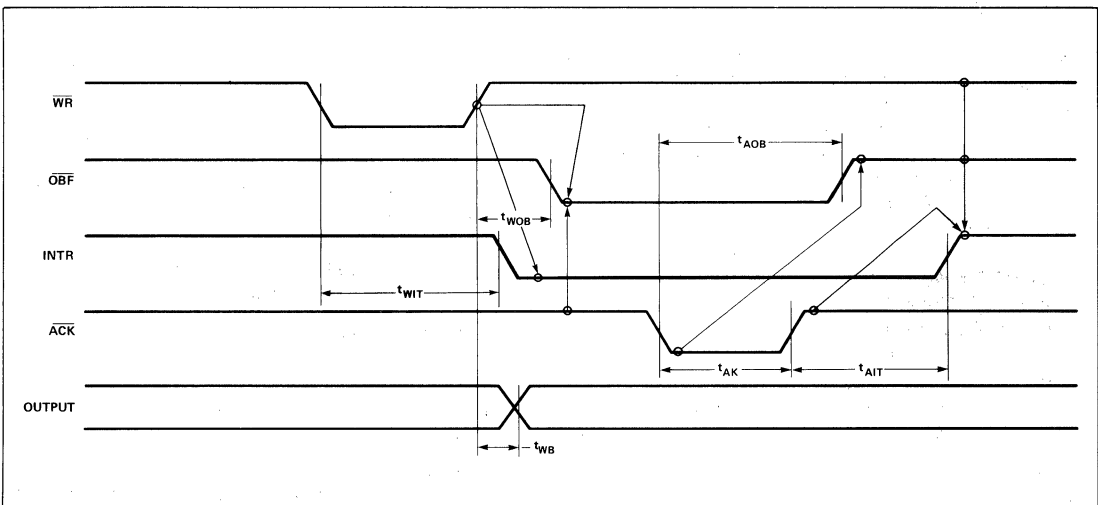
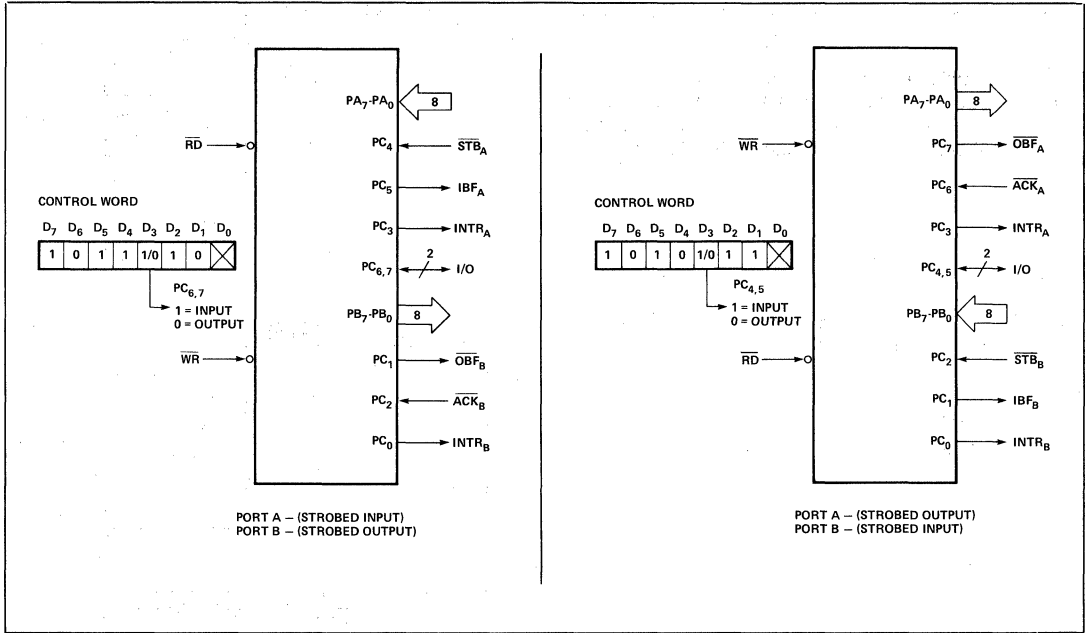


Figure 11. Mode 1 (Strobed Output)



**Combinations of MODE 1**

Port A and Port B can be individually defined as input or output in Mode 1 to support a wide variety of strobed I/O applications.



**Figure 12. Combinations of MODE 1**

**Operating Modes**

**MODE 2 (Strobed Bidirectional Bus I/O).** This functional configuration provides a means for communicating with a peripheral device or structure on a single 8-bit bus for both transmitting and receiving data (bidirectional bus I/O). "Handshaking" signals are provided to maintain proper bus flow discipline in a similar manner to MODE 1. Interrupt generation and enable/disable functions are also available.

**MODE 2 Basic Functional Definitions:**

- Used in Group A only.
- One 8-bit, bi-directional bus Port (Port A) and a 5-bit control Port (Port C).
- Both inputs and outputs are latched.
- The 5-bit control port (Port C) is used for control and status for the 8-bit, bi-directional bus port (Port A).

**Bidirectional Bus I/O Control Signal Definition**

**INTR (Interrupt Request).** A high on this output can be used to interrupt the CPU for both input or output operations.

**Output Operations**

**OBF (Output Buffer Full).** The OBF output will go "low" to indicate that the CPU has written data out to port A.

**ACK (Acknowledge).** A "low" on this input enables the tri-state output buffer of port A to send out the data. Otherwise, the output buffer will be in the high impedance state.

**INTE 1 (The INTE Flip-Flop Associated with OBF).** Controlled by bit set/reset of PC<sub>6</sub>.

**Input Operations**

**STB (Strobe Input)**

**STB (Strobe Input).** A "low" on this input loads data into the input latch.

**IBF (Input Buffer Full F/F).** A "high" on this output indicates that data has been loaded into the input latch.

**INTE 2 (The INTE Flip-Flop Associated with IBF).** Controlled by bit set/reset of PC<sub>4</sub>.

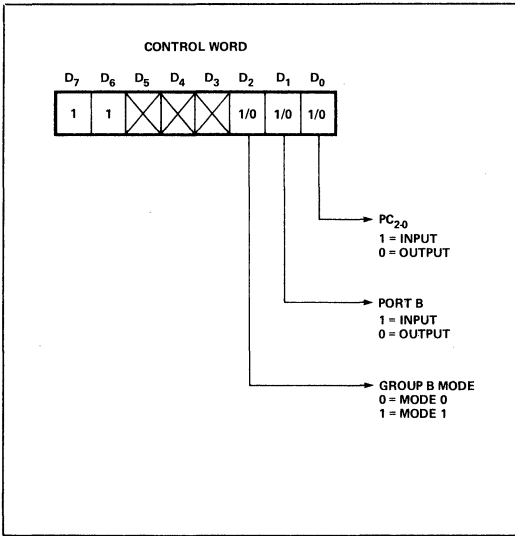


Figure 13. MODE Control Word

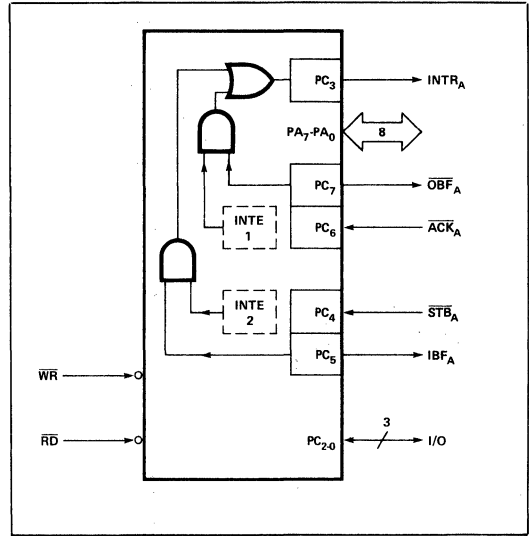


Figure 14. MODE 2

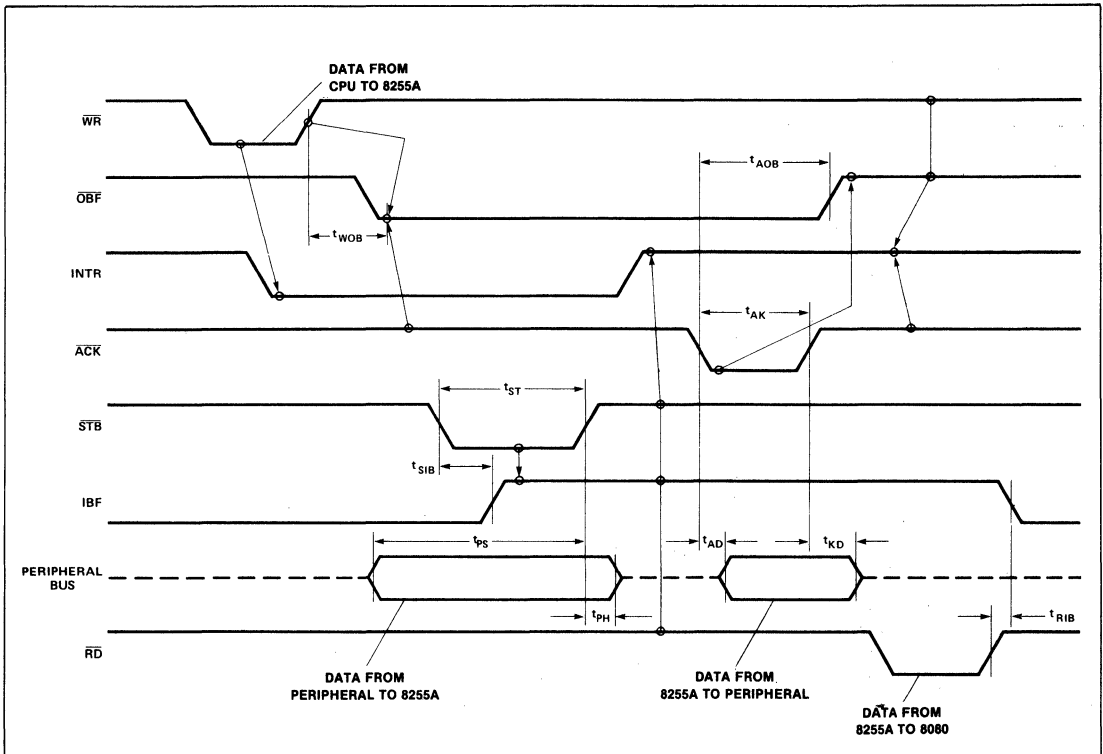


Figure 15. MODE 2 (Bidirectional)

NOTE: Any sequence where  $\overline{WR}$  occurs before  $\overline{ACK}$  and  $\overline{STB}$  occurs before  $\overline{RD}$  is permissible.  
 $(INTR = IBF \cdot MASK \cdot \overline{STB} \cdot \overline{RD} + OBF \cdot MASK \cdot \overline{ACK} \cdot \overline{WR})$

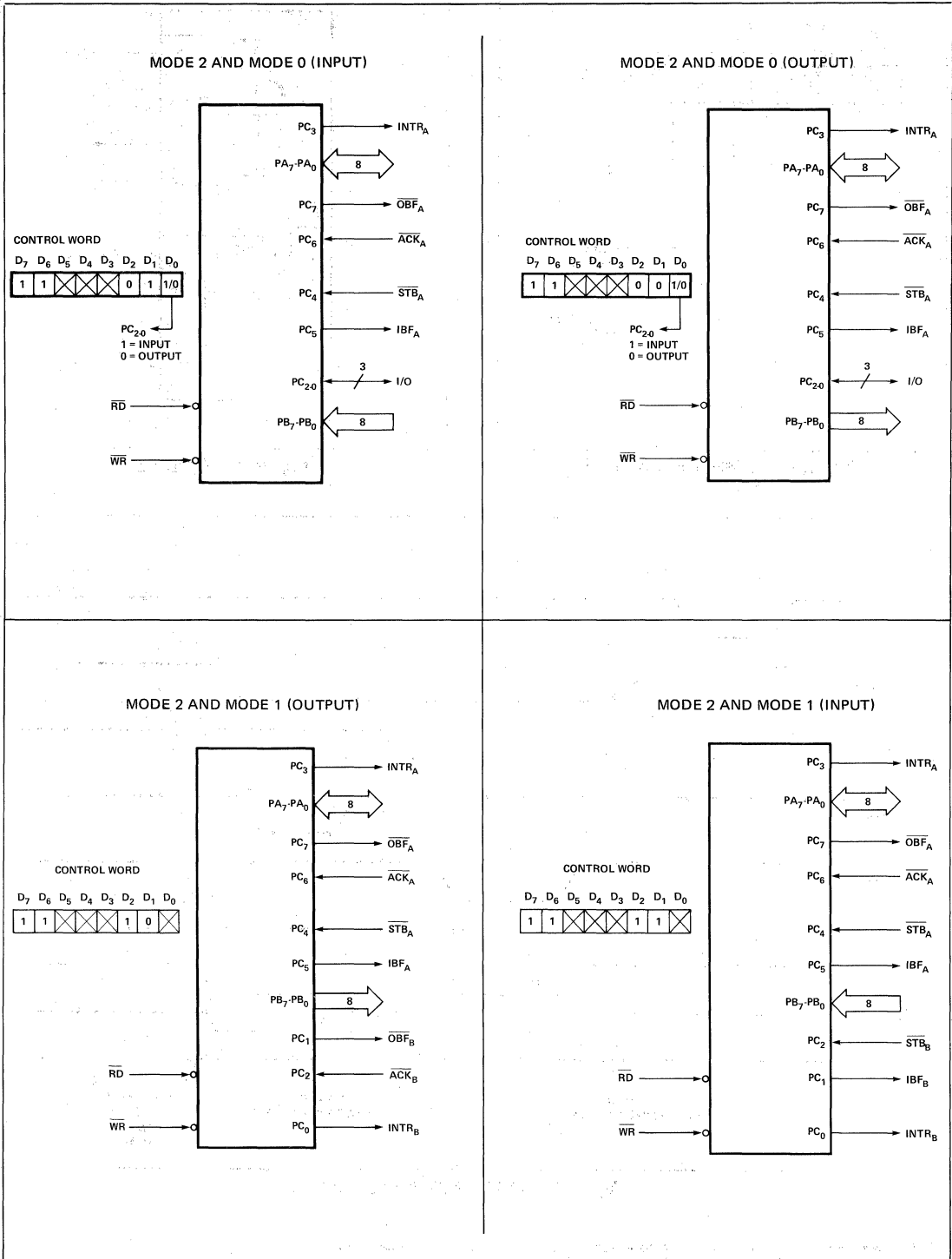


Figure 16. MODE ¼ Combinations

**Mode Definition Summary**

	MODE 0		MODE 1		MODE 2	
	IN	OUT	IN	OUT	GROUP A ONLY	
PA <sub>0</sub>	IN	OUT	IN	OUT	↔	
PA <sub>1</sub>	IN	OUT	IN	OUT	↔	
PA <sub>2</sub>	IN	OUT	IN	OUT	↔	
PA <sub>3</sub>	IN	OUT	IN	OUT	↔	
PA <sub>4</sub>	IN	OUT	IN	OUT	↔	
PA <sub>5</sub>	IN	OUT	IN	OUT	↔	
PA <sub>6</sub>	IN	OUT	IN	OUT	↔	
PA <sub>7</sub>	IN	OUT	IN	OUT	↔	
PB <sub>0</sub>	IN	OUT	IN	OUT	—	
PB <sub>1</sub>	IN	OUT	IN	OUT	—	
PB <sub>2</sub>	IN	OUT	IN	OUT	—	
PB <sub>3</sub>	IN	OUT	IN	OUT	—	
PB <sub>4</sub>	IN	OUT	IN	OUT	—	
PB <sub>5</sub>	IN	OUT	IN	OUT	—	
PB <sub>6</sub>	IN	OUT	IN	OUT	—	
PB <sub>7</sub>	IN	OUT	IN	OUT	—	
PC <sub>0</sub>	IN	OUT	INTR <sub>B</sub>	INTR <sub>B</sub>	I/O	
PC <sub>1</sub>	IN	OUT	IBF <sub>B</sub>	OBFB	I/O	
PC <sub>2</sub>	IN	OUT	STB <sub>B</sub>	ACK <sub>B</sub>	I/O	
PC <sub>3</sub>	IN	OUT	INTR <sub>A</sub>	INTR <sub>A</sub>	INTR <sub>A</sub>	
PC <sub>4</sub>	IN	OUT	STB <sub>A</sub>	I/O	STB <sub>A</sub>	
PC <sub>5</sub>	IN	OUT	IBF <sub>A</sub>	I/O	IBF <sub>A</sub>	
PC <sub>6</sub>	IN	OUT	I/O	ACK <sub>A</sub>	ACK <sub>A</sub>	
PC <sub>7</sub>	IN	OUT	I/O	OBFA	OBFA	

**Special Mode Combination Considerations**

There are several combinations of modes when not all of the bits in Port C are used for control or status. The remaining bits can be used as follows:

If Programmed as Inputs –

All input lines can be accessed during a normal Port C read.

If Programmed as Outputs –

Bits in C upper (PC<sub>7</sub>-PC<sub>4</sub>) must be individually accessed using the bit set/reset function.

Bits in C lower (PC<sub>3</sub>-PC<sub>0</sub>) can be accessed using the bit set/reset function or accessed as a threesome by writing into Port C.

**Source Current Capability on Port B and Port C**

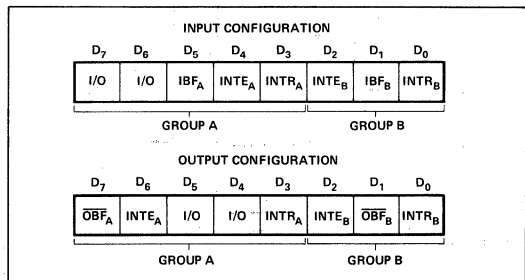
Any set of eight output buffers, selected randomly from Ports B and C can source 1mA at 1.5 volts. This feature allows the 8255 to directly drive Darlington type drivers and high-voltage displays that require such source current.

**Reading Port C Status**

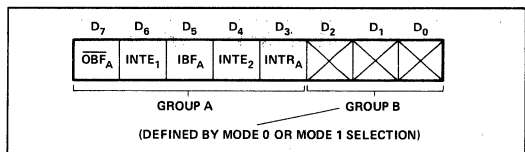
In Mode 0, Port C transfers data to or from the peripheral device. When the 8255 is programmed to function in Modes 1 or 2, Port C generates or accepts "hand-shaking" signals with the peripheral device. Reading the contents of Port C

allows the programmer to test or verify the "status" of each peripheral device and change the program flow accordingly.

There is no special instruction to read the status information from Port C. A normal read operation of Port C is executed to perform this function.



**Figure 17. MODE 1 Status Word Format**



**Figure 18. MODE 2 Status Word Format**

### APPLICATIONS OF THE 8255A

The 8255A is a very powerful tool for interfacing peripheral equipment to the microcomputer system. It represents the optimum use of available pins and is flexible enough to interface almost any I/O device without the need for additional external logic.

Each peripheral device in a microcomputer system usually has a "service routine" associated with it. The routine manages the software interface between the device and the CPU. The functional definition of the 8255A is programmed by the I/O service routine and becomes an extension of the system software. By examining the I/O devices interface characteristics for both data transfer and timing, and matching this information to the examples and tables in the detailed operational description, a control word can easily be developed to initialize the 8255A to exactly "fit" the application. Figures 19 through 25 present a few examples of typical applications of the 8255A.

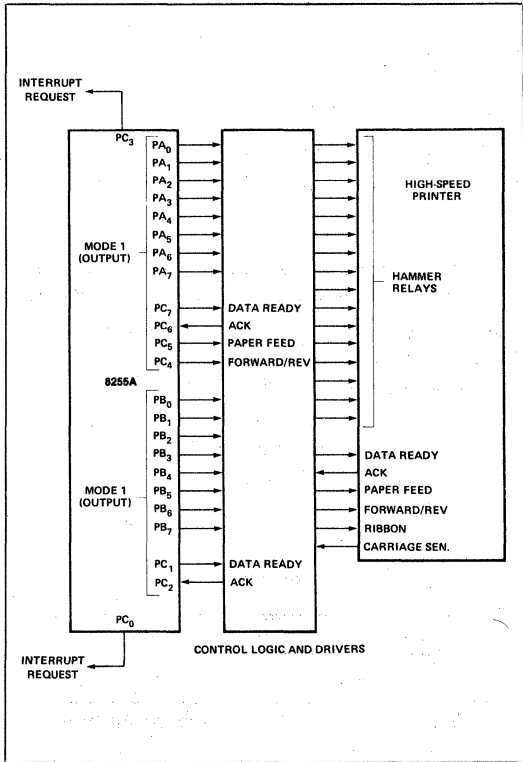


Figure 19. Printer Interface

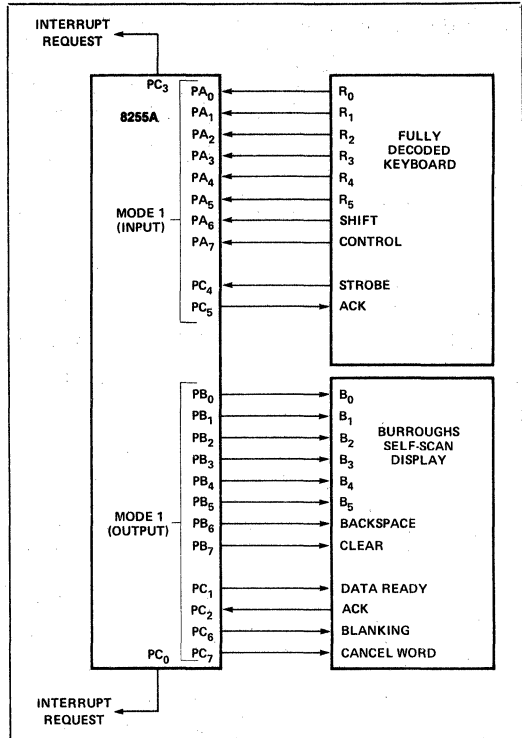


Figure 20. Keyboard and Display Interface

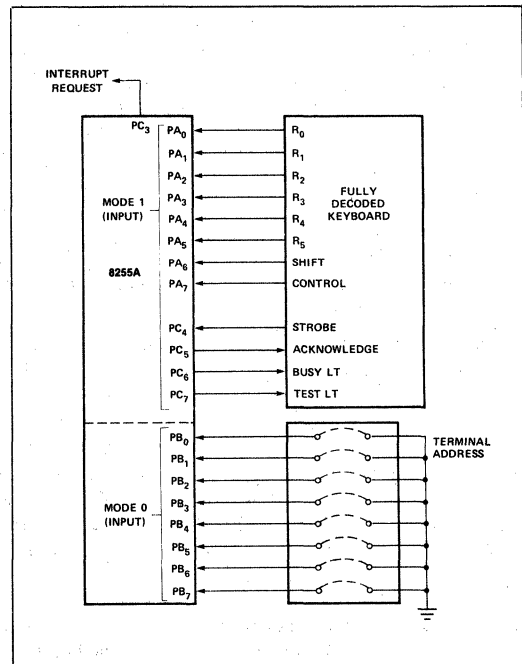


Figure 21. Keyboard and Terminal Address Interface

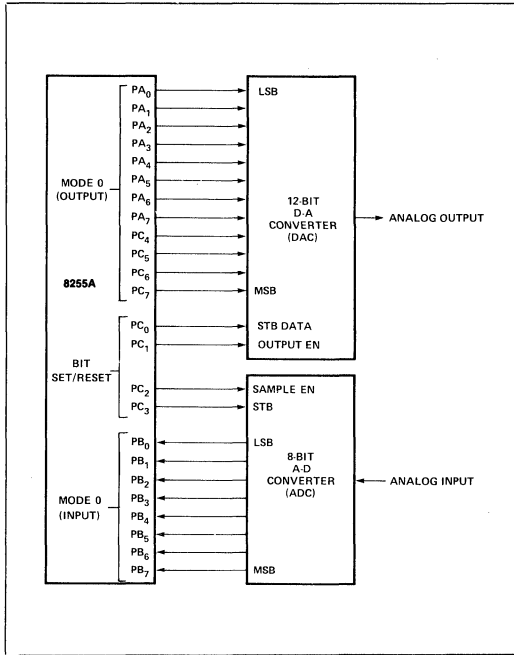


Figure 22. Digital to Analog, Analog to Digital

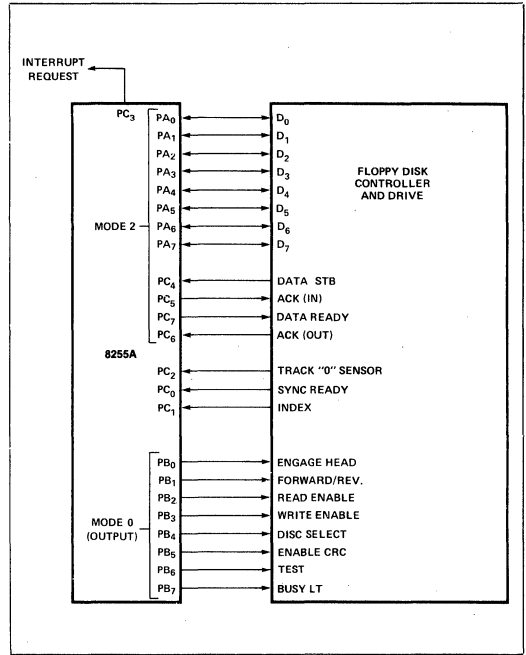


Figure 23. Basic CRT Controller Interface

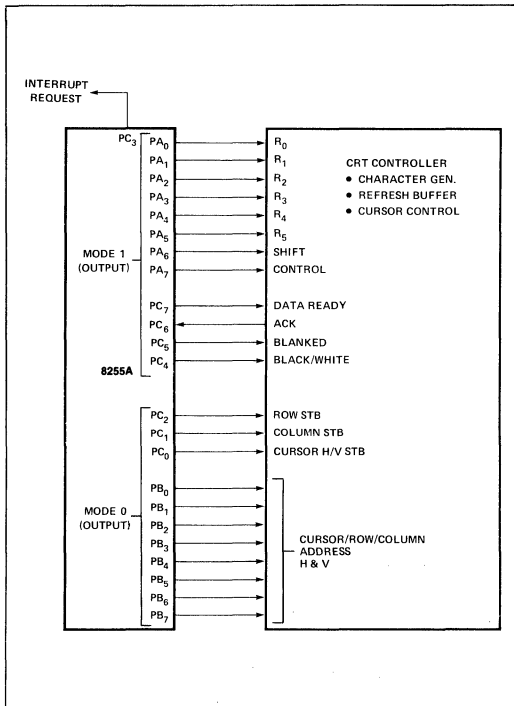


Figure 24. Basic Floppy Disc Interface

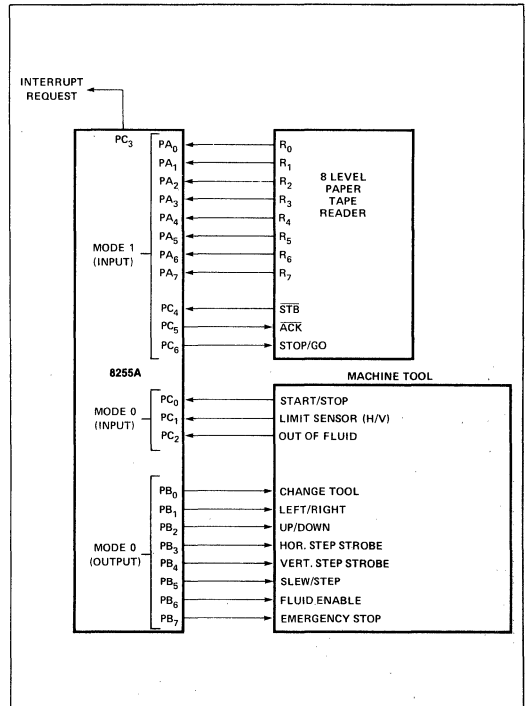


Figure 25. Machine Tool Controller Interface

**ABSOLUTE MAXIMUM RATINGS\***

Ambient Temperature Under Bias. . . . . 0°C to 70°C  
 Storage Temperature . . . . . -65°C to +150°C  
 Voltage on Any Pin  
     With Respect to Ground. . . . . -0.5V to +7V  
 Power Dissipation . . . . . 1 Watt

*\*NOTICE: Stresses above those listed under "Absolute Maximum Ratings" may cause permanent damage to the device. This is a stress rating only and functional operation of the device at these or any other conditions above those indicated in the operational sections of this specification is not implied. Exposure to absolute maximum rating conditions for extended periods may affect device reliability.*

**D.C. CHARACTERISTICS** (T<sub>A</sub> = 0°C to 70°C, V<sub>CC</sub> = +5V ± 5%, GND = 0V)

Symbol	Parameter	Min.	Max.	Unit	Test Conditions
V <sub>IL</sub>	Input Low Voltage	-0.5	0.8	V	
V <sub>IH</sub>	Input High Voltage	2.0	V <sub>CC</sub>	V	
V <sub>OL</sub> (DB)	Output Low Voltage (Data Bus)		0.45	V	I <sub>OL</sub> = 2.5mA
V <sub>OL</sub> (PER)	Output Low Voltage (Peripheral Port)		0.45	V	I <sub>OL</sub> = 1.7mA
V <sub>OH</sub> (DB)	Output High Voltage (Data Bus)	2.4		V	I <sub>OH</sub> = -400μA
V <sub>OH</sub> (PER)	Output High Voltage (Peripheral Port)	2.4		V	I <sub>OH</sub> = -200μA
I <sub>DAR</sub> [1]	Darlington Drive Current	-1.0	-4.0	mA	R <sub>EXT</sub> = 750Ω; V <sub>EXT</sub> = 1.5V
I <sub>CC</sub>	Power Supply Current		120	mA	
I <sub>IL</sub>	Input Load Current		±10	μA	V <sub>IN</sub> = V <sub>CC</sub> to 0V
I <sub>OFL</sub>	Output Float Leakage		±10	μA	V <sub>OUT</sub> = V <sub>CC</sub> to 0V

**NOTE:**

1. Available on any 8 pins from Port B and C.

**CAPACITANCE** (T<sub>A</sub> = 25°C, V<sub>CC</sub> = GND = 0V)

Symbol	Parameter	Min.	Typ.	Max.	Unit	Test Conditions
C <sub>IN</sub>	Input Capacitance			10	pF	f <sub>c</sub> = 1MHz
C <sub>I/O</sub>	I/O Capacitance			20	pF	Unmeasured pins returned to GND

**A.C. CHARACTERISTICS** (T<sub>A</sub> = 0°C to 70°C, V<sub>CC</sub> = +5V ± 5%, GND = 0V)

**Bus Parameters**
**READ**

Symbol	Parameter	8255A		8255A-5		Unit
		Min.	Max.	Min.	Max.	
t <sub>AR</sub>	Address Stable Before READ	0		0		ns
t <sub>RA</sub>	Address Stable After READ	0		0		ns
t <sub>RR</sub>	READ Pulse Width	300		300		ns
t <sub>RD</sub>	Data Valid From READ <sup>[1]</sup>		250		200	ns
t <sub>DF</sub>	Data Float After READ	10	150	10	100	ns
t <sub>RV</sub>	Time Between READs and/or WRITEs	850		850		ns

**A.C. CHARACTERISTICS (Continued)**

**WRITE**

Symbol	Parameter	8255A		8255A-5		Unit
		Min.	Max.	Min.	Max.	
$t_{AW}$	Address Stable Before WRITE	0		0		ns
$t_{WA}$	Address Stable After WRITE	20		20		ns
$t_{WW}$	WRITE Pulse Width	400		300		ns
$t_{DW}$	Data Valid to WRITE (T.E.)	100		100		ns
$t_{WD}$	Data Valid After WRITE	30		30		ns

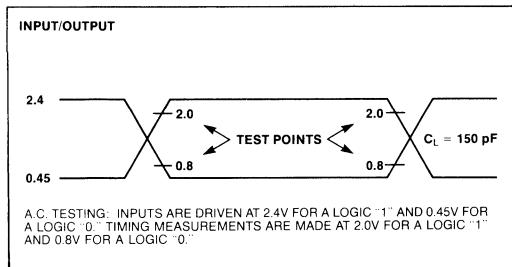
**OTHER TIMINGS**

Symbol	Parameter	8255A		8255A-5		Unit
		Min.	Max.	Min.	Max.	
$t_{WB}$	WR = 1 to Output <sup>[1]</sup>		350		350	ns
$t_{IR}$	Peripheral Data Before RD	0		0		ns
$t_{HR}$	Peripheral Data After RD	0		0		ns
$t_{AK}$	ACK Pulse Width	300		300		ns
$t_{ST}$	STB Pulse Width	500		500		ns
$t_{PS}$	Per. Data Before T.E. of STB	0		0		ns
$t_{PH}$	Per. Data After T.E. of STB	180		180		ns
$t_{AD}$	ACK = 0 to Output <sup>[1]</sup>		300		300	ns
$t_{KD}$	ACK = 1 to Output Float	20	250	20	250	ns
$t_{WOB}$	WR = 1 to OBF = 0 <sup>[1]</sup>		650		650	ns
$t_{AOB}$	ACK = 0 to OBF = 1 <sup>[1]</sup>		350		350	ns
$t_{SIB}$	STB = 0 to IBF = 1 <sup>[1]</sup>		300		300	ns
$t_{RIB}$	RD = 1 to IBF = 0 <sup>[1]</sup>		300		300	ns
$t_{RIT}$	RD = 0 to INTR = 0 <sup>[1]</sup>		400		400	ns
$t_{SIT}$	STB = 1 to INTR = 1 <sup>[1]</sup>		300		300	ns
$t_{AIT}$	ACK = 1 to INTR = 1 <sup>[1]</sup>		350		350	ns
$t_{WIT}$	WR = 0 to INTR = 0 <sup>[1,3]</sup>		450		450	ns

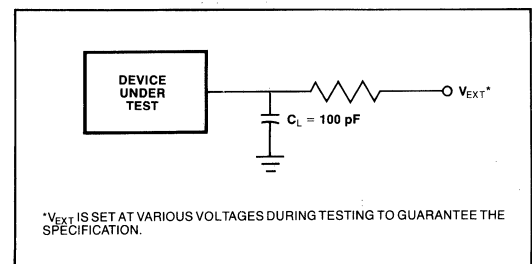
**NOTES:**

1. Test Conditions: 8255A:  $C_L = 100\text{pF}$ ; 8255A-5:  $C_L = 150\text{pF}$ .
2. Period of Reset pulse must be at least  $50\mu\text{s}$  during or after power on. Subsequent Reset pulse can be 500 ns min.
3. INTR $\uparrow$  may occur as early as WR $\downarrow$ .

**A.C. TESTING INPUT, OUTPUT WAVEFORM**

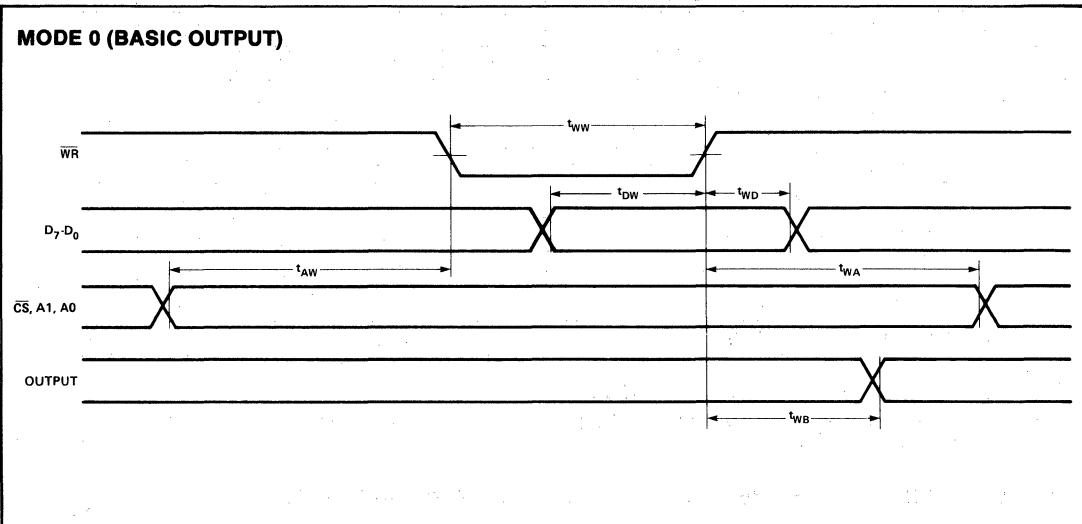
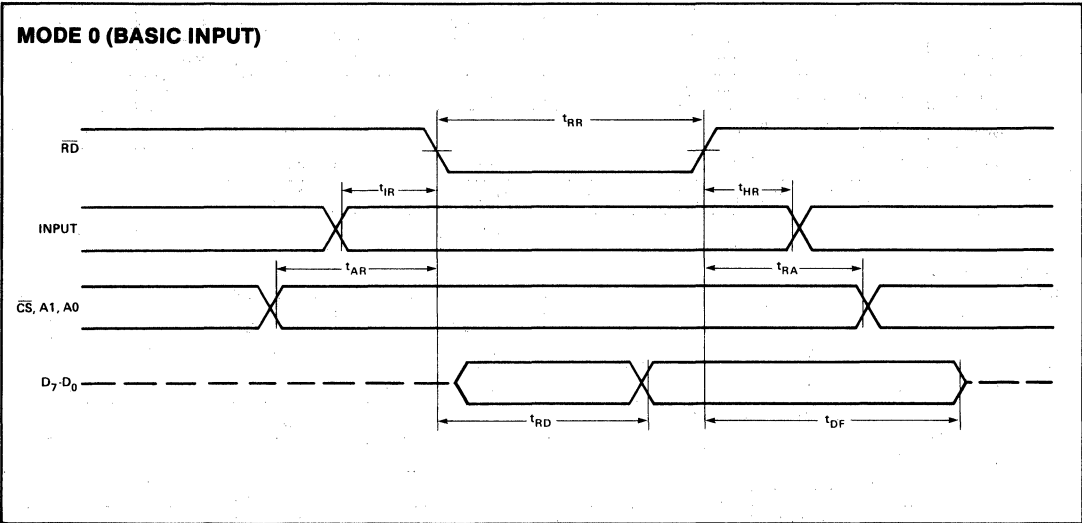


**A.C. TESTING LOAD CIRCUIT**

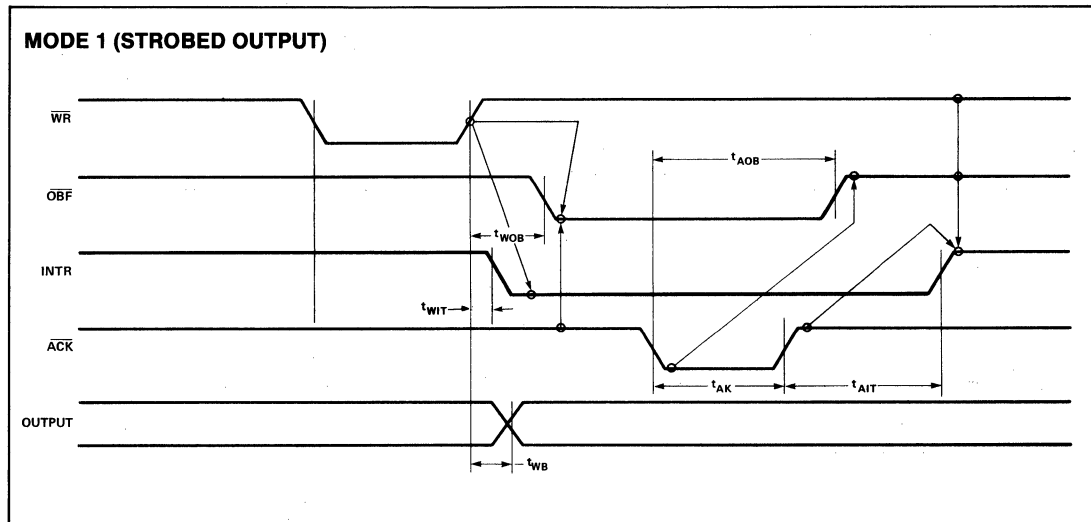
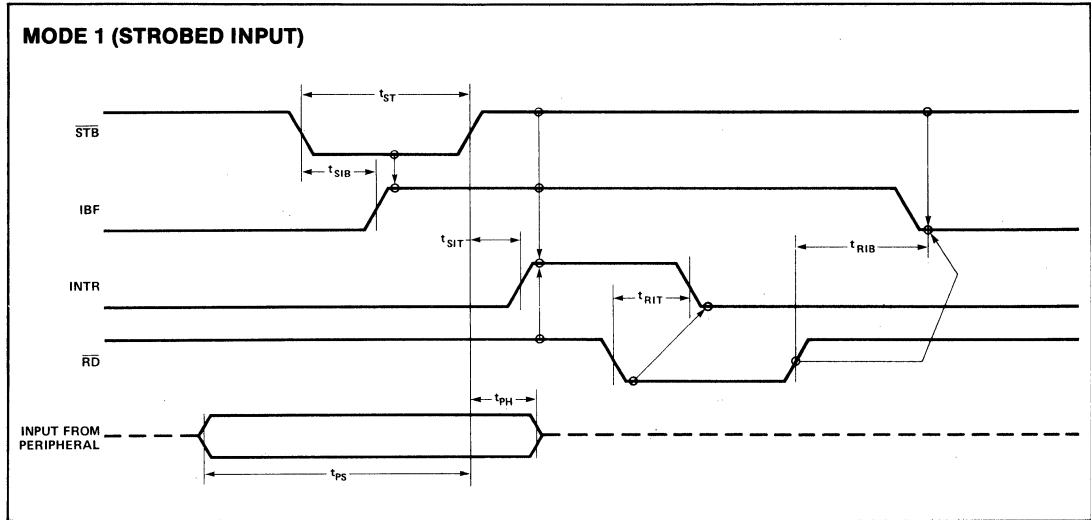




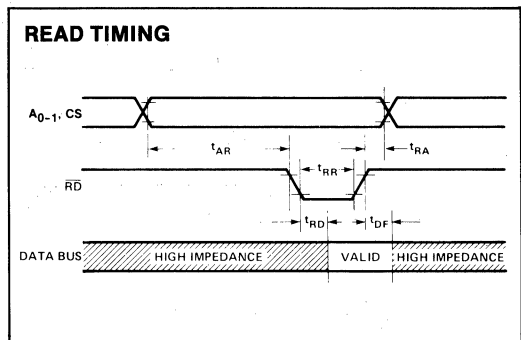
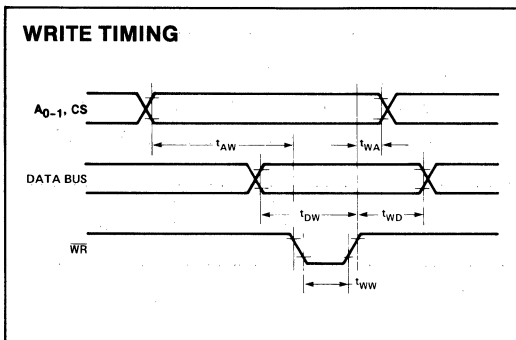
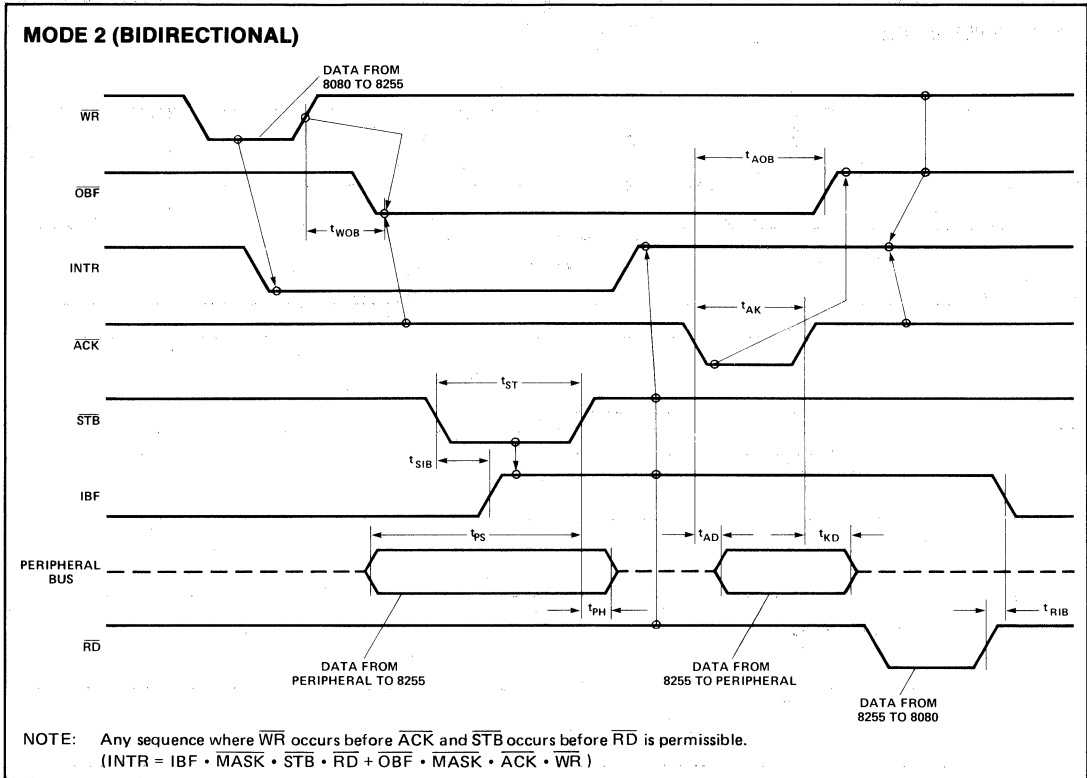
WAVEFORMS



WAVEFORMS (Continued)



WAVEFORMS (Continued)





# 8275 PROGRAMMABLE CRT CONTROLLER

- Programmable Screen and Character Format
- Fully MCS-80™ and MCS-85™ Compatible
- 6 Independent Visual Field Attributes
- Dual Row Buffers
- 11 Visual Character Attributes (Graphic Capability)
- Programmable DMA Burst Mode
- Cursor Control (4 Types)
- Single +5V Supply
- Light Pen Detection and Registers
- 40-Pin Package

The Intel® 8275 Programmable CRT Controller is a single chip device to interface CRT raster scan displays with Intel® microcomputer systems. Its primary function is to refresh the display by buffering the information from main memory and keeping track of the display position of the screen. The flexibility designed into the 8275 will allow simple interface to almost any raster scan CRT display with a minimum of external hardware and software overhead.

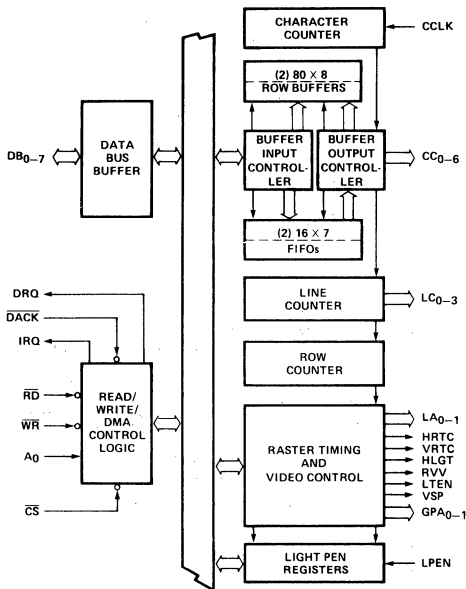


Figure 1. Block Diagram

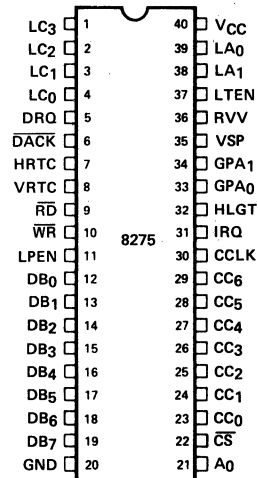


Figure 2. Pin Configuration

**Table 1. Pin Descriptions**

Symbol	Pin No.	Type	Name and Function
LC <sub>3</sub>	1	O	<b>Line Count:</b> Output from the line counter which is used to address the character generator for the line positions on the screen.
LC <sub>2</sub>	2		
LC <sub>1</sub>	3		
LC <sub>0</sub>	4		
DRQ	5	O	<b>DMA Request:</b> Output signal to the 8257 DMA controller requesting a DMA cycle.
DACK	6	I	<b>DMA Acknowledge:</b> Input signal from the 8257 DMA controller acknowledging that the requested DMA cycle has been granted.
HRTC	7	O	<b>Horizontal Retrace:</b> Output signal which is active during the programmed horizontal retrace interval. During this period the VSP output is high and the LTEN output is low.
VRTC	8	O	<b>Vertical Retrace:</b> Output signal which is active during the programmed vertical retrace interval. During this period the VSP output is high and the LTEN output is low.
$\overline{\text{RD}}$	9	I	<b>Read Input:</b> A control signal to read registers.
$\overline{\text{WR}}$	10	I	<b>Write Input:</b> A control signal to write commands into the control registers or write data into the row buffers during a DMA cycle.
LPEN	11	I	<b>Light Pen:</b> Input signal from the CRT system signifying that a light pen signal has been detected.
DB <sub>0</sub>	12	I/O	<b>Bi-Directional Three-State Data Bus Lines:</b> The outputs are enabled during a read of the C or P ports.
DB <sub>1</sub>	13		
DB <sub>2</sub>	14		
DB <sub>3</sub>	15		
DB <sub>4</sub>	16		
DB <sub>5</sub>	17		
DB <sub>6</sub>	18		
DB <sub>7</sub>	19		
Ground	20		<b>Ground.</b>

Symbol	Pin No.	Type	Name and Function
V <sub>CC</sub>	40		<b>+5V Power Supply.</b>
LA <sub>0</sub>	39	O	<b>Line Attribute Codes:</b> These attribute codes have to be decoded externally by the dot/timing logic to generate the horizontal and vertical line combinations for the graphic displays specified by the character attribute codes.
LA <sub>1</sub>	38		
LTEN	37	O	<b>Light Enable:</b> Output signal used to enable the video signal to the CRT. This output is active at the programmed underline cursor position, and at positions specified by attribute codes.
RVV	36	O	<b>Reverse Video:</b> Output signal used to indicate the CRT circuitry to reverse the video signal. This output is active at the cursor position if a reverse video block cursor is programmed or at the positions specified by the field attribute codes.
VSP	35	O	<b>Video Suppression:</b> Output signal used to blank the video signal to the CRT. This output is active: <ul style="list-style-type: none"> <li>—during the horizontal and vertical retrace intervals.</li> <li>—at the top and bottom lines of rows if underline is programmed to be number 8 or greater.</li> <li>—when an end of row or end of screen code is detected.</li> <li>—when a DMA underrun occurs.</li> <li>—at regular intervals (1/16 frame frequency for cursor, 1/32 frame frequency for character and field attributes)—to create blinking displays as specified by cursor, character attribute, or field attribute programming.</li> </ul>
GPA <sub>1</sub>	34	O	<b>General Purpose Attribute Codes:</b> Outputs which are enabled by the general purpose field attribute codes.
GPA <sub>0</sub>	33		
HLGT	32	O	<b>Highlight:</b> Output signal used to intensify the display at particular positions on the screen as specified by the character attribute codes or field attribute codes.
IRQ	31	O	<b>Interrupt Request.</b>
CCLK	30	I	<b>Character Clock (from dot/timing logic).</b>
CC <sub>6</sub>	29	O	<b>Character Codes:</b> Output from the row buffers used for character selection in the character generator.
CC <sub>5</sub>	28		
CC <sub>4</sub>	27		
CC <sub>3</sub>	26		
CC <sub>2</sub>	25		
CC <sub>1</sub>	24		
CC <sub>0</sub>	23		
CS	22	I	<b>Chip Select:</b> The read and write are enabled by CS.
A <sub>0</sub>	21	I	<b>Port Address:</b> A high input on A <sub>0</sub> selects the "C" port or command registers and a low input selects the "P" port or parameter registers.

## FUNCTIONAL DESCRIPTION

### Data Bus Buffer

This 3-state, bidirectional, 8-bit buffer is used to interface the 8275 to the system Data Bus.

This functional block accepts inputs from the System Control Bus and generates control signals for overall device operation. It contains the Command, Parameter, and Status Registers that store the various control formats for the device functional definition.

A <sub>0</sub>	OPERATION	REGISTER
0	Read	PREG
0	Write	PREG
1	Read	SREG
1	Write	CREG

### RD (Read)

A "low" on this input informs the 8275 that the CPU is reading data or status information from the 8275.

### WR (Write)

A "low" on this input informs the 8275 that the CPU is writing data or control words to the 8275.

### CS (Chip Select)

A "low" on this input selects the 8275. No reading or writing will occur unless the device is selected. When CS is high, the Data Bus in the float state and RD and WR will have no effect on the chip.

### DRQ (DMA Request)

A "high" on this output informs the DMA Controller that the 8275 desires a DMA transfer.

### DACK (DMA Acknowledge)

A "low" on this input informs the 8275 that a DMA cycle is in progress.

### IRQ (Interrupt Request)

A "high" on this output informs the CPU that the 8275 desires interrupt service.

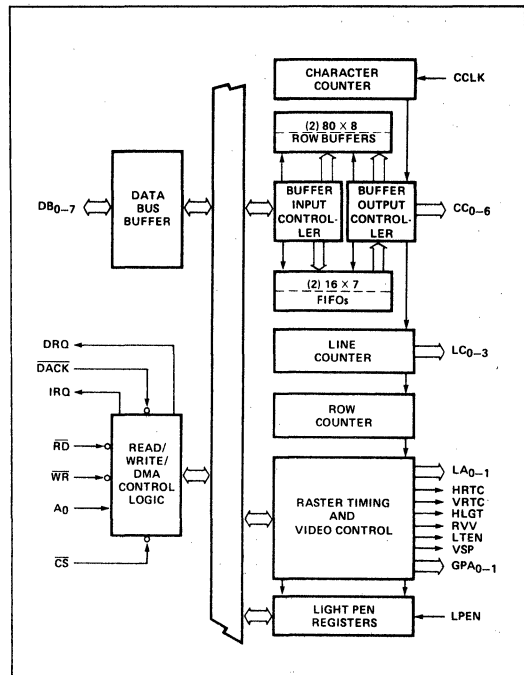


Figure 3. 8275 Block Diagram Showing Data Bus Buffer and Read/Write Functions

A <sub>0</sub>	RD	WR	CS	Function
0	0	1	0	Write 8275 Parameter
0	1	0	0	Read 8275 Parameter
1	0	1	0	Write 8275 Command
1	1	0	0	Read 8275 Status
X	1	1	0	Three-State
X	X	X	1	Three-state

### Character Counter

The Character Counter is a programmable counter that is used to determine the number of characters to be displayed per row and the length of the horizontal retrace interval. It is driven by the CCLK (Character Clock) input, which should be a derivative of the external dot clock.

### Line Counter

The Line Counter is a programmable counter that is used to determine the number of horizontal lines (Sweeps) per character row. Its outputs are used to address the external character generator ROM.

### Row Counter

The Row Counter is a programmable counter that is used to determine the number of character rows to be displayed per frame and length of the vertical retrace interval.

### Light Pen Registers

The Light Pen Registers are two registers that store the contents of the character counter and the row counter whenever there is a rising edge on the LPEN (Light Pen) input.

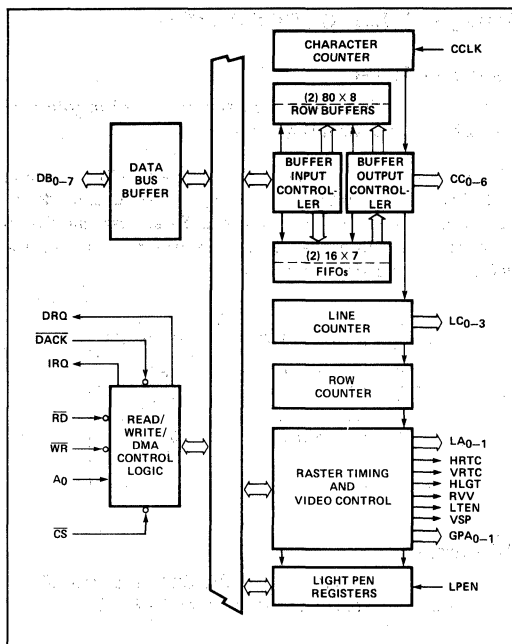
**Note:** Software correction is required.

### Raster Timing and Video Controls

The Raster Timing circuitry controls the timing of the HRTC (Horizontal Retrace) and VRTC (Vertical Retrace) outputs. The Video Control circuitry controls the generation of LA<sub>0-1</sub> (Line Attribute), HGLT (Highlight), RVV (Reverse Video), LTEN (Light Enable), VSP (Video Suppress), and GPA<sub>0-1</sub> (General Purpose Attribute) outputs.

### Row Buffers

The Row Buffers are two 80 character buffers. They are filled from the microcomputer system memory with the character codes to be displayed. While one row buffer is displaying a row of characters, the other is being filled with the next row of characters.



**Figure 4. 8275 Block Diagram Showing Counter and Register Functions**

### FIFOs

There are two 16 character FIFOs in the 8275. They are used to provide extra row buffer length in the Transparent Attribute Mode (see Detailed Operation section).

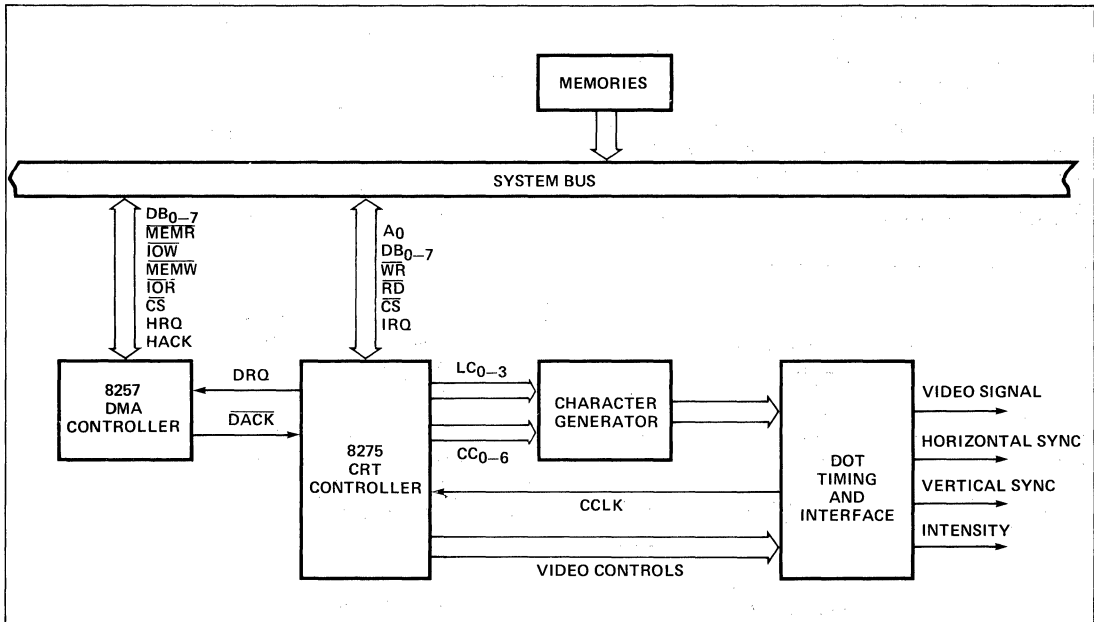
### Buffer Input/Output Controllers

The Buffer Input/Output Controllers decode the characters being placed in the row buffers. If the character is a character attribute, field attribute or special code, these controllers control the appropriate action. (Examples: An "End of Screen—Stop DMA" special code will cause the Buffer Input Controller to stop further DMA requests. A "Highlight" field attribute will cause the Buffer Output Controller to activate the HGLT output.)

**SYSTEM OPERATION**

The 8275 is programmable to a large number of different display formats. It provides raster timing, display row buffering, visual attribute decoding, cursor timing, and light pen detection.

It is designed to interface with the 8257 DMA Controller and standard character generator ROMs for dot matrix decoding. Dot level timing must be provided by external circuitry.



**Figure 5. 8275 Systems Block Diagram Showing Systems Operation**



### General Systems Operational Description

The 8275 provides a "window" into the microcomputer system memory.

Display characters are retrieved from memory and displayed on a row by row basis. The 8275 has two row buffers. While one row buffer is being used for display, the other is being filled with the next row of characters to be displayed. The number of display characters per row and the number of character rows per frame are software programmable, providing easy interface to most CRT displays. (See Programming Section.)

The 8275 requests DMA to fill the row buffer that is not being used for display. DMA burst length and spacing is programmable. (See Programming Section.)

The 8275 displays character rows one line at a time.

The number of lines per character row, the underline position, and blanking of top and bottom lines are programmable. (See Programming Section.)

The 8275 provides special Control Codes which can be used to minimize DMA or software overhead. It also provides Visual Attribute Codes to cause special action or symbols on the screen without the use of the character generator (see Visual Attributes Section).

The 8275 also controls raster timing. This is done by generating Horizontal Retrace (HRTC) and Vertical Retrace (VRTC) signals. The timing of these signals is programmable.

The 8275 can generate a cursor. Cursor location and format are programmable. (See Programming Section.)

The 8275 has a light pen input and registers. The light pen input is used to load the registers. Light pen registers can be read on command. (See Programming Section.)

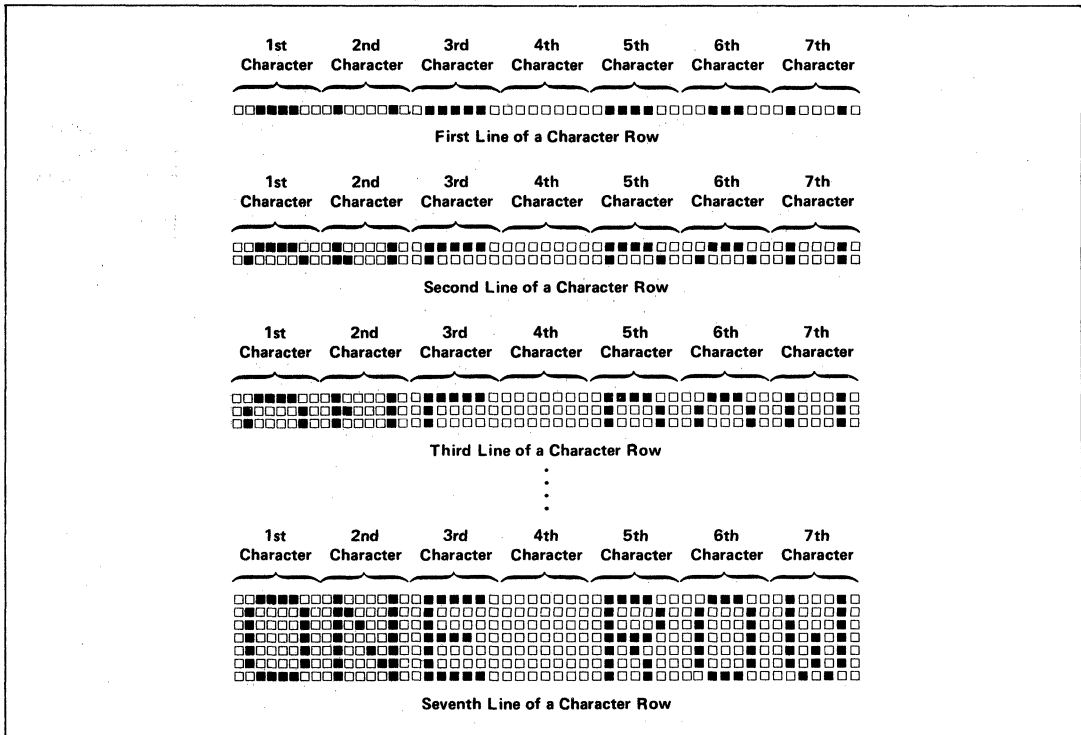
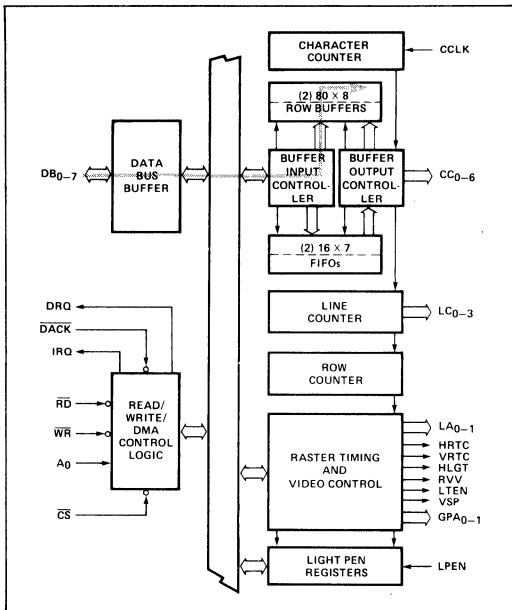


Figure 6. Display of a Character Row

**Display Row Buffering**

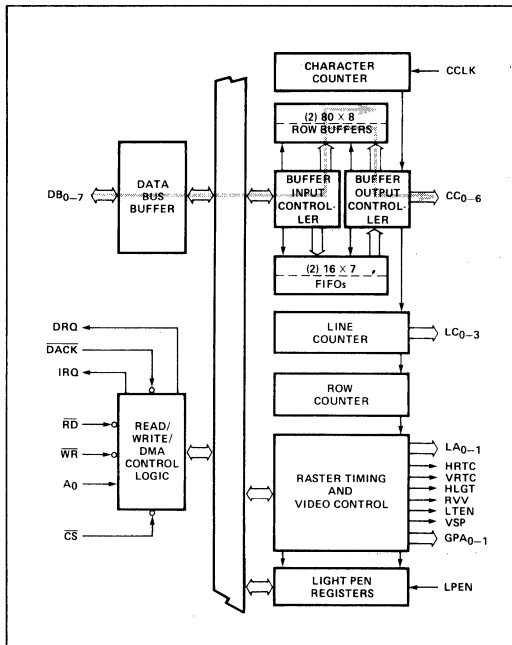
Before the start of a frame, the 8275 requests DMA and one row buffer is filled with characters.

After all the lines of the character row are scanned, the roles of the two row buffers are reversed and the same procedure is followed for the next row.



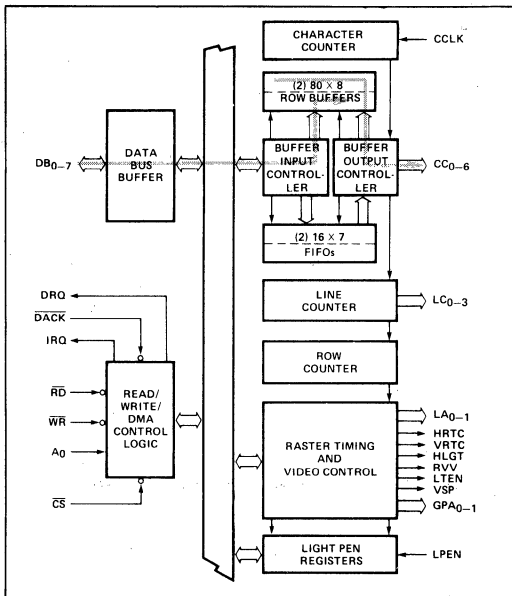
**Figure 7. First Row Buffer Filled**

When the first horizontal sweep is started, character codes are output to the character generator from the row buffer just filled. Simultaneously, DMA begins filling the other row buffer with the next row of characters.



**Figure 9. First Buffer Filled with Third Row, Second Row Displayed**

This is repeated until all of the character rows are displayed.



**Figure 8. Second Buffer Filled, First Row Displayed**

### Display Format

#### Screen Format

The 8275 can be programmed to generate from 1 to 80 characters per row, and from 1 to 64 rows per frame.

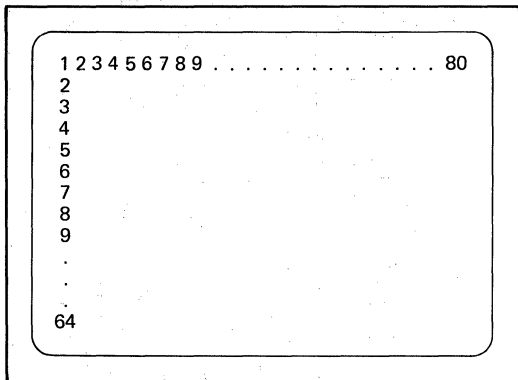


Figure 10. Screen Format

The 8275 can also be programmed to blank alternate rows. In this mode, the first row is displayed, the second blanked, the third displayed, etc. DMA is not requested for the blanked rows.

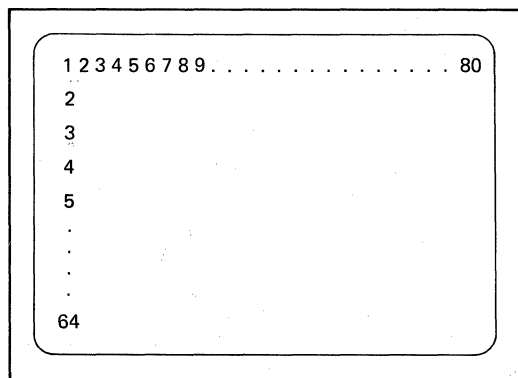


Figure 11. Blank Alternate Rows Mode

#### Row Format

The 8275 is designed to hold the line count stable while outputting the appropriate character codes during each horizontal sweep. The line count is incremented during horizontal retrace and the whole row of character codes are output again during the next sweep. This is continued until the whole character row is displayed.

The number of lines (horizontal sweeps) per character row is programmable from 1 to 16.

The output of the line counter can be programmed to be in one of two modes.

In mode 0, the output of the line counter is the same as the line number.

In mode 1, the line counter is offset by one from the line number.

**Note:** In mode 1, while the first line (line number 0) is being displayed, the last count is output by the line counter (see examples).

Line Number		Line Counter Mode 0	Line Counter Mode 1
0	□ □ □ □ □ □ □ □	0000	1111
1	□ □ □ □ ■ □ □ □	0001	0000
2	□ □ □ ■ □ □ □ □	0010	0001
3	□ □ ■ □ □ □ □ □	0011	0010
4	□ ■ □ □ □ □ □ □	0100	0011
5	□ ■ □ □ □ □ □ □	0101	0100
6	□ ■ ■ □ □ □ □ □	0110	0101
7	□ ■ □ □ □ □ □ □	0111	0110
8	□ ■ □ □ □ □ □ □	1000	0111
9	□ ■ □ □ □ □ □ □	1001	1000
10	□ □ □ □ □ □ □ □	1010	1001
11	□ □ □ □ □ □ □ □	1011	1010
12	□ □ □ □ □ □ □ □	1100	1011
13	□ □ □ □ □ □ □ □	1101	1100
14	□ □ □ □ □ □ □ □	1110	1101
15	□ □ □ □ □ □ □ □	1111	1110

Figure 12. Example of a 16-Line Format

Line Number		Line Counter Mode 0	Line Counter Mode 1
0	□ □ □ □ □ □ □ □	0000	1001
1	□ □ □ □ ■ □ □ □	0001	0000
2	□ □ ■ □ □ □ □ □	0010	0001
3	□ ■ □ □ □ □ □ □	0011	0010
4	□ ■ □ □ □ □ □ □	0100	0011
5	□ ■ ■ □ □ □ □ □	0101	0100
6	□ ■ □ □ □ □ □ □	0110	0101
7	□ ■ □ □ □ □ □ □	0111	0110
8	□ □ □ □ □ □ □ □	1000	0111
9	□ □ □ □ □ □ □ □	1001	1000

Figure 13. Example of a 10-Line Format

Mode 0 is useful for character generators that leave address zero blank and start at address 1. Mode 1 is useful for character generators which start at address zero.

Underline placement is also programmable (from line number 0 to 15). This is independent of the line counter mode.

If the line number of the underline is greater than 7 (line number MSB = 1), then the top and bottom lines will be blanked.

Line Number		Line Counter Mode 0	Line Counter Mode 1
0	□ □ □ □ □ □ □ □	0000	1011
1	□ □ □ □ ■ □ □ □	0001	0000
2	□ □ □ ■ □ □ □ □	0010	0001
3	□ □ ■ □ □ □ □ □	0011	0010
4	□ ■ □ □ □ □ □ □	0100	0011
5	□ ■ □ □ □ □ □ □	0101	0100
6	□ ■ ■ □ □ □ □ □	0110	0101
7	□ ■ □ □ □ □ □ □	0111	0110
8	■ □ □ □ □ □ □ □	1000	0111
9	■ ■ □ □ □ □ □ □	1001	1000
10	■ ■ ■ □ □ □ □ □	1010	1001
11	□ □ □ □ □ □ □ □	1011	1010

Top and Bottom Lines are Blanked

Figure 14. Underline in Line Number 10

If the line number of the underline is less than or equal to 7 (line number MSB = 0), then the top and bottom lines will not be blanked.

Line Number		Line Counter Mode 0	Line Counter Mode 1
0	□ □ □ ■ □ □ □ □	0000	0111
1	□ □ ■ □ □ □ □ □	0001	0000
2	□ ■ □ □ □ □ □ □	0010	0001
3	□ ■ □ □ □ □ □ □	0011	0010
4	□ ■ ■ □ □ □ □ □	0100	0011
5	□ ■ □ □ □ □ □ □	0101	0100
6	□ ■ □ □ □ □ □ □	0110	0101
7	■ ■ ■ □ □ □ □ □	0111	0110

Top and Bottom Lines are not Blanked

Figure 15. Underline in Line Number 7

If the line number of the underline is greater than the maximum number of lines, the underline will not appear.

Blanking is accomplished by the VSP (Video Suppression) signal. Underline is accomplished by the LTEN (Light Enable) signal.

Dot Format

Dot width and character width are dependent upon the external timing and control circuitry.

Dot level timing circuitry should be designed to accept the parallel output of the character generator and shift it out serially at the rate required by the CRT display.

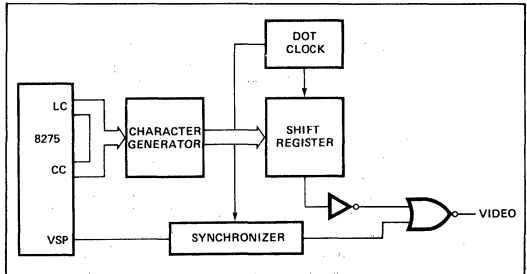


Figure 16. Typical Dot Level Block Diagram

Dot width is a function of dot clock frequency.

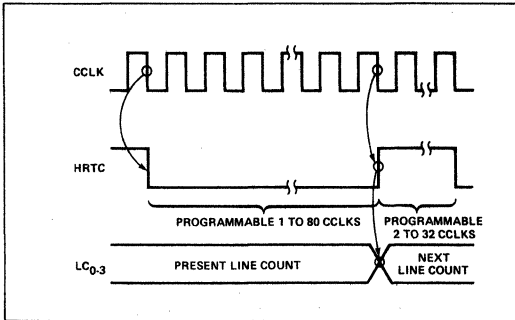
Character width is a function of the character generator width.

Horizontal character spacing is a function of the shift register length.

Note: Video control and timing signals must be synchronized with the video signal due to the character generator access delay.

**Raster Timing**

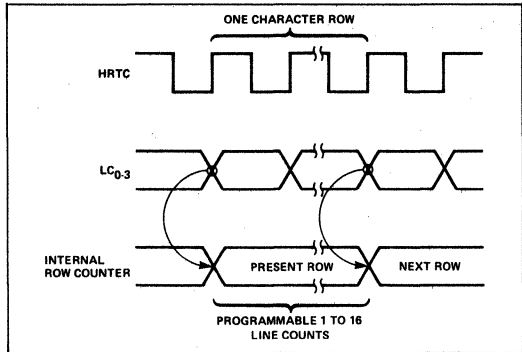
The character counter is driven by the character clock input (CCLK). It counts out the characters being displayed (programmable from 1 to 80). It then causes the line counter to increment, and it starts counting out the horizontal retrace interval (programmable from 2 to 32). This is constantly repeated.



**Figure 17. Line Timing**

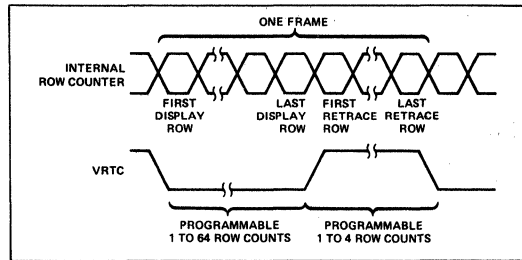
The line counter is driven by the character counter. It is used to generate the line address outputs (LC<sub>0-3</sub>) for the character generator. After it counts all of the lines in a character row (programmable from 1 to 16), it increments the row counter, and starts over again. (See Character Format Section for detailed description of Line Counter functions.)

The row counter is an internal counter driven by the line counter. It controls the functions of the row buffers and counts the number of character rows displayed.



**Figure 18. Row Timing**

After the row counter counts all of the rows in a frame (programmable from 1 to 64), it starts counting out the vertical retrace interval (programmable from 1 to 4).



**Figure 19. Frame Timing**

The Video Suppression Output (VSP) is active during horizontal and vertical retrace intervals.

Dot level timing circuitry must synchronize these outputs with the video signal to the CRT Display.

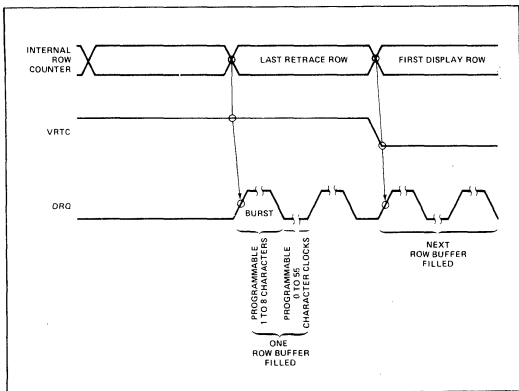
**DMA Timing**

The 8275 can be programmed to request burst DMA transfers of 1 to 8 characters. The interval between bursts is also programmable (from 0 to 55 character clock periods  $\pm 1$ ). This allows the user to tailor his DMA overhead to fit his system needs.

The first DMA request of the frame occurs one *row time* before the end of vertical retrace. DMA requests continue as programmed, until the row buffer is filled. If the row buffer is filled in the middle of a burst, the 8275 terminates the burst and resets the burst counter. No more DMA requests will occur until the *beginning* of the *next* row. At that time, DMA requests are activated as programmed until the other buffer is filled.

The first DMA request for a row will start at the first character clock of the preceding row. If the burst mode is used, the first DMA request may occur a number of character clocks later. This number is equal to the programmed burst space.

If, for any reason, there is a DMA underrun, a flag in the status word will be set.

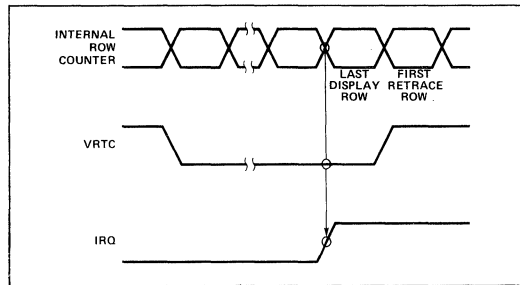


**Figure 20. DMA Timing**

The DMA controller is typically initialized for the next frame at the end of the current frame.

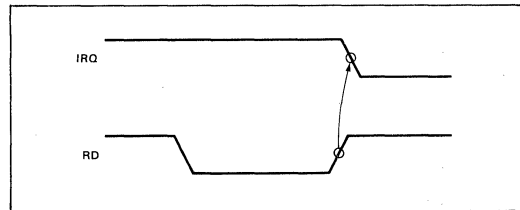
**Interrupt Timing**

The 8275 can be programmed to generate an interrupt request at the end of each frame. This can be used to reinitialize the DMA controller. If the 8275 interrupt enable flag is set, an interrupt request will occur at the *beginning* of the *last display row*.



**Figure 21. Beginning of Interrupt Request**

IRQ will go inactive after the status register is read.



**Figure 22. End of Interrupt Request**

A reset command will also cause IRQ to go inactive, but this is not recommended during normal service.

Another method of reinitializing the DMA controller is to have the DMA controller itself interrupt on terminal count. With this method, the 8275 interrupt enable flag should not be set.

**Note:** Upon power-up, the 8275 Interrupt Enable Flag may be set. As a result, the user's cold start routine should write a reset command to the 8275 before system interrupts are enabled.

## VISUAL ATTRIBUTES AND SPECIAL CODES

The characters processed by the 8275 are 8-bit quantities. The character code outputs provide the character generator with 7 bits of address. The Most Significant Bit is the extra bit and it is used to determine if it is a normal display character (MSB = 0), or if it is a Visual Attribute or Special Code (MSB = 1).

There are two types of Visual Attribute Codes. They are Character Attributes and Field Attributes.

### Character Attribute Codes

Character attribute codes are codes that can be used to generate graphics symbols without the use of a character generator. This is accomplished by selectively activating the Line Attribute outputs (LA<sub>0-1</sub>), the Video Suppression output (VSP), and the Light Enable output. The dot level timing circuitry can use these signals to generate the proper symbols.

Character attributes can be programmed to blink or be highlighted individually. Blinking is accomplished with the Video Suppression output (VSP). Blink frequency is equal to the screen refresh frequency divided by 32. Highlighting is accomplished by activating the Highlight output (HGLT).

### Character Attributes

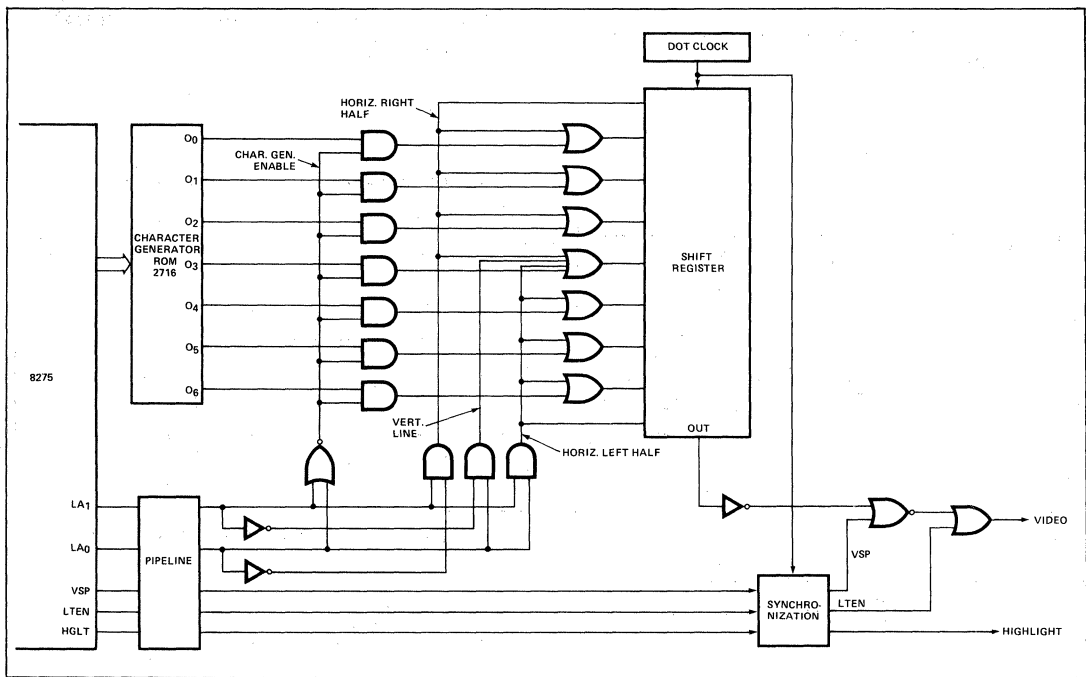
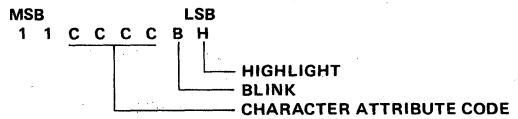


Figure 23. Typical Character Attribute Logic

**Table 2. Character Attributes**

Character attributes were designed to produce the following graphics:

CHARACTER ATTRIBUTE CODE "CCCC"	OUTPUTS				SYMBOL	DESCRIPTION	
	LA <sub>1</sub>	LA <sub>0</sub>	VSP	LTEN			
0000	Above Underline	0	0	1	0		Top Left Corner
	Underline	1	0	0	0		
	Below Underline	0	1	0	0		
0001	Above Underline	0	0	1	0		Top Right Corner
	Underline	1	1	0	0		
	Below Underline	0	1	0	0		
0010	Above Underline	0	1	0	0		Bottom Left Corner
	Underline	1	0	0	0		
	Below Underline	0	0	1	0		
0011	Above Underline	0	1	0	0		Bottom Right Corner
	Underline	1	1	0	0		
	Below Underline	0	0	1	0		
0100	Above Underline	0	0	1	0		Top Intersect
	Underline	0	0	0	1		
	Below Underline	0	1	0	0		
0101	Above Underline	0	1	0	0		Right Intersect
	Underline	1	1	0	0		
	Below Underline	0	1	0	0		
0110	Above Underline	0	1	0	0		Left Intersect
	Underline	1	0	0	0		
	Below Underline	0	1	0	0		
0111	Above Underline	0	1	0	0		Bottom Intersect
	Underline	0	0	0	1		
	Below Underline	0	0	1	0		
1000	Above Underline	0	0	1	0		Horizontal Line
	Underline	0	0	0	1		
	Below Underline	0	0	1	0		
1001	Above Underline	0	1	0	0		Vertical Line
	Underline	0	1	0	0		
	Below Underline	0	1	0	0		
1010	Above Underline	0	1	0	0		Crossed Lines
	Underline	0	0	0	1		
	Below Underline	0	1	0	0		
1011	Above Underline	0	0	0	0		Not Recommended *
	Underline	0	0	0	0		
	Below Underline	0	0	0	0		
1100	Above Underline	0	0	1	0		Special Codes
	Underline	0	0	1	0		
	Below Underline	0	0	1	0		
1101	Above Underline						Illegal
	Underline		Undefined				
	Below Underline						
1110	Above Underline						Illegal
	Underline		Undefined				
	Below Underline						
1111	Above Underline						Illegal
	Underline		Undefined				
	Below Underline						

\*Character Attribute Code 1011 is not recommended for normal operation. Since none of the attribute outputs are active, the character Generator will not be disabled, and an indeterminate character will be generated.

Character Attribute Codes 1101, 1110, and 1111 are illegal.

Blinking is active when B = 1.

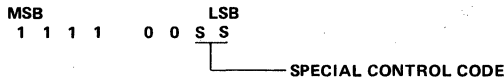
Highlight is active when H = 1.



**Special Codes**

Four special codes are available to help reduce memory, software, or DMA overhead.

**Special Control Character**



S	S	FUNCTION
0	0	End of Row
0	1	End of Row-Stop DMA
1	0	End of Screen
1	1	End of Screen-Stop DMA

The End of Row Code (00) activates VSP and holds it to the end of the line.

The End of Row-Stop DMA Code (01) causes the DMA Control Logic to stop DMA for the rest of the row when it is written into the Row Buffer. It affects the display in the same way as the End of Row Code (00).

The End of Screen Code (10) activates VSP and holds it to the end of the frame.

The End of Screen-Stop DMA Code (11) causes the DMA Control Logic to stop DMA for the rest of the frame when it is written into the Row Buffer. It affects the display in the same way as the End of Screen Code (10).

If the Stop DMA feature is not used, all characters after an End of Row character are ignored, except for the End of Screen character, which operates normally. All characters after an End of Screen character are ignored.

**Note:** If a Stop DMA character is not the last character in a burst or row, DMA is not stopped until after the next character is read. In this situation, a dummy character must be placed in memory after the Stop DMA character.

**Field Attributes**

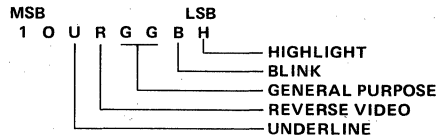
The field attributes are control codes which affect the visual characteristics for a field of characters, starting at the

character following the code up to, and including, the character which precedes the *next* field attribute code, or up to the end of the frame. The field attributes are reset during the vertical retrace interval.

There are six field attributes:

1. **Blink** — Characters following the code are caused to blink by activating the Video Suppression output (VSP). The blink frequency is equal to the screen refresh frequency divided by 32.
2. **Highlight** — Characters following the code are caused to be highlighted by activating the Highlight output (HGLT).
3. **Reverse Video** — Characters following the code are caused to appear with reverse video by activating the Reverse Video output (RVV).
4. **Underline** — Characters following the code are caused to be underlined by activating the Light Enable output (LTEN).
- 5,6. **General Purpose** — There are two additional 8275 outputs which act as general purpose, independently programmable field attributes. GPA<sub>0-1</sub> are active high outputs.

**Field Attribute Code**

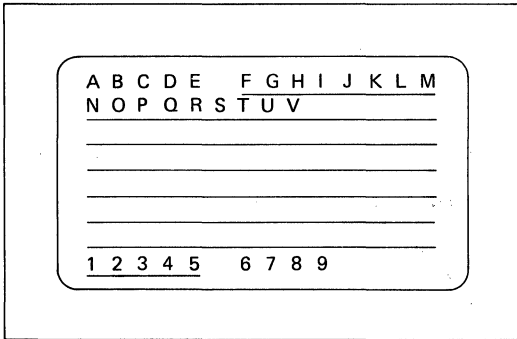


- H = 1 FOR HIGHLIGHTING
- B = 1 FOR BLINKING
- R = 1 FOR REVERSE VIDEO
- U = 1 FOR UNDERLINE
- GG = GPA<sub>1</sub>, GPA<sub>0</sub>

\*More than one attribute can be enabled at the same time. If the blinking and reverse video attributes are enabled simultaneously, only the reversed characters will blink.

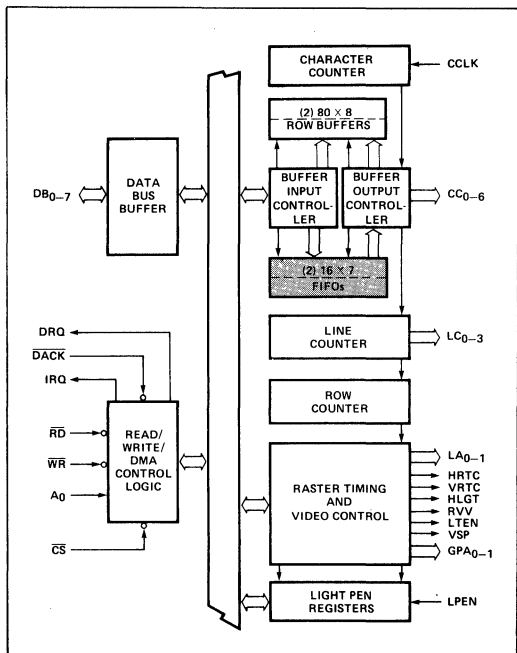
The 8275 can be programmed to provide visible or invisible field attribute characters.

If the 8275 is programmed in the visible field attribute mode, all field attributes will occupy a position on the screen. They will appear as blanks caused by activation of the Video Suppression output (VSP). The chosen visual attributes are activated after this blanked character.



**Figure 24. Example of the Visible Field Attribute Mode (Underline Attribute)**

If the 8275 is programmed in the invisible field attribute mode, the 8275 FIFO is activated.



**Figure 25. Block Diagram Showing FIFO Activation**

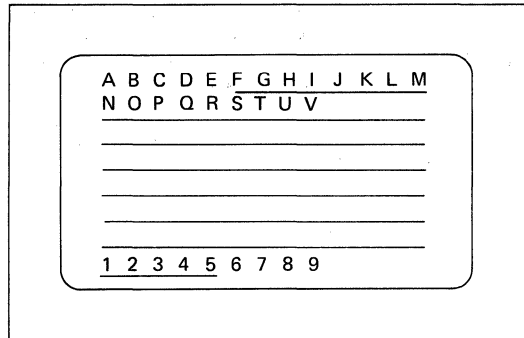
Each row buffer has a corresponding FIFO. These FIFOs are 16 characters by 7 bits in size.

When a field attribute is placed in the row buffer during DMA, the buffer input controller recognizes it and places the *next* character in the proper FIFO.

When a field attribute is placed in the Buffer Output Controller during display, it causes the controller to immediately put a character from the FIFO on the Character Code outputs (CC0-6). The chosen Visual Attributes are also activated.

Since the FIFO is 16 characters long, no more than 16 field attribute characters may be used per line in this mode. If more are used, a bit in the status word is set and the first characters in the FIFO are written over and lost.

**Note:** Since the FIFO is 7 bits wide, the MSB of any characters put in it are stripped off. Therefore, a Visual Attribute or Special Code must *not* immediately follow a field attribute code. If this situation does occur, the Visual Attribute or Special Code will be treated as a normal display character.



**Figure 26. Example of the Invisible Field Attribute Mode (Underline Attribute)**

**Field and Character Attribute Interaction**

Character Attribute Symbols are affected by the Reverse Video (RVV) and General Purpose (GPA0-1) field attributes. They are not affected by Underline, Blink or Highlight field attributes; however, these characteristics can be programmed *individually* for Character Attribute Symbols.

### Cursor Timing

The cursor location is determined by a cursor row register and a character position register which are loaded by command to the controller. The cursor can be programmed to appear on the display as:

1. a blinking underline
2. a blinking reverse video block
3. a non-blinking underline
4. a non-blinking reverse video block

The cursor blinking frequency is equal to the screen refresh frequency divided by 16.

If a non-blinking reverse video *cursor* appears in a non-blinking reverse video *field*, the cursor will appear as a normal video block.

If a non-blinking underline *cursor* appears in a non-blinking underline *field*, the cursor will not be visible.

### Light Pen Detection

A light pen consists of a micro switch and a tiny light sensor. When the light pen is pressed against the CRT screen, the micro switch enables the light sensor. When the raster sweep reaches the light sensor, it triggers the light pen output.

If the output of the light pen is presented to the 8275 LPEN input, the row and character position coordinates are stored in a pair of registers. These registers can be read on command. A bit in the status word is set, indicating that the light pen signal was detected. The LPEN input must be a 0 to 1 transition for proper operation.

**Note:** Due to internal and external delays, the character position coordinate will be off by at least three character positions. This has to be corrected in software.

### Device Programming

The 8275 has two programming registers, the Command Register (CREG) and the Parameter Register (PREG). It also has a Status Register (SREG). The Command Register can only be written into and the Status Registers can only be read from. They are addressed as follows:

A <sub>0</sub>	OPERATION	REGISTER
0	Read	PREG
0	Write	PREG
1	Read	SREG
1	Write	CREG

The 8275 expects to receive a command and a sequence of 0 to 4 parameters, depending on the command. If the proper number of parameter bytes are not received before another command is given, a status flag is set, indicating an improper command.

### Instruction Set

The 8275 instruction set consists of 8 commands.

COMMAND	NO. OF PARAMETER BYTES
Reset	4
Start Display	0
Stop Display	0
Read Light Pen	2
Load Cursor	2
Enable Interrupt	0
Disable Interrupt	0
Preset Counters	0

In addition, the status of the 8275 (SREG) can be read by the CPU at any time.

1. Reset Command:

	OPERATION	A <sub>0</sub>	DESCRIPTION	DATA BUS									
				MSB							LSB		
Command	Write	1	Reset Command	0	0	0	0	0	0	0	0	0	0
Parameters	Write	0	Screen Comp Byte 1	S	H	H	H	H	H	H	H	H	H
	Write	0	Screen Comp Byte 2	V	V	R	R	R	R	R	R	R	R
	Write	0	Screen Comp Byte 3	U	U	U	U	L	L	L	L	L	L
	Write	0	Screen Comp Byte 4	M	F	C	C	Z	Z	Z	Z	Z	Z

**Action** — After the reset command is written, DMA requests stop, 8275 interrupts are disabled, and the VSP output is used to blank the screen. HRTC and VRTC continue to run. HRTC and VRTC timing are random on power-up.

As parameters are written, the screen composition is defined.

Parameter — S Spaced Rows

S	FUNCTIONS
0	Normal Rows
1	Spaced Rows

Parameter — HHHHHHH Horizontal Characters/Row

H H H H H H H H	NO. OF CHARACTERS PER ROW
0 0 0 0 0 0 0 0	1
0 0 0 0 0 0 0 1	2
0 0 0 0 0 0 1 0	3
.	.
.	.
1 0 0 1 1 1 1 1	80
1 0 1 0 0 0 0 0	Undefined
.	.
.	.
1 1 1 1 1 1 1 1	Undefined

Parameter — VV Vertical Retrace Row Count

V V	NO. OF ROW COUNTS PER VRTC
0 0	1
0 1	2
1 0	3
1 1	4

Parameter — RRRRRR Vertical Rows/Frame

R R R R R R	NO. OF ROWS/FRAME
0 0 0 0 0 0	1
0 0 0 0 0 1	2
0 0 0 0 1 0	3
.	.
.	.
1 1 1 1 1 1	64

Parameter — UUUU Underline Placement

U U U U	LINE NUMBER OF UNDERLINE
0 0 0 0	1
0 0 0 1	2
0 0 1 0	3
.	.
.	.
1 1 1 1	16

Parameter — LLLL Number of Lines per Character Row

L L L L	NO. OF LINES/ROW
0 0 0 0	1
0 0 0 1	2
0 0 1 0	3
.	.
.	.
1 1 1 1	16

Parameter — M Line Counter Mode

M	LINE COUNTER MODE
0	Mode 0 (Non-Offset)
1	Mode 1 (Offset by 1 Count)

Parameter — F Field Attribute Mode

F	FIELD ATTRIBUTE MODE
0	Transparent
1	Non-Transparent

Parameter — CC Cursor Format

C C	CURSOR FORMAT
0 0	Blinking reverse video block
0 1	Blinking underline
1 0	Nonblinking reverse video block
1 1	Nonblinking underling

Parameter — ZZZZ Horizontal Retrace Count

Z Z Z Z	NO. OF CHARACTER COUNTS PER HRTC
0 0 0 0	2
0 0 0 1	4
0 0 1 0	6
.	.
.	.
1 1 1 1	32

**Note:** uuuu MSB determines blanking of top and bottom lines (1 = blanked, 0 = not blanked).

2. Start Display Command:

	OPERATION	A <sub>0</sub>	DESCRIPTION	DATA BUS	
				MSB	LSB
Command	Write	1	Start Display	0 0 1 S S S B B	
No parameters					

**S S S BURST SPACE CODE**

S S S	NO. OF CHARACTER CLOCKS BETWEEN DMA REQUESTS
0 0 0	0
0 0 1	7
0 1 0	15
0 1 1	23
1 0 0	31
1 0 1	39
1 1 0	47
1 1 1	55

**B B BURST COUNT CODE**

B B	NO. OF DMA CYCLES PER BURST
0 0	1
0 1	2
1 0	4
1 1	8

**Action** — 8275 interrupts are enabled, DMA requests begin, video is enabled, Interrupt Enable and Video Enable status flags are set.

3. Stop Display Command:

	OPERATION	A <sub>0</sub>	DESCRIPTION	DATA BUS	
				MSB	LSB
Command	Write	1	Stop Display	0 1 0 0 0 0 0 0	
No parameters					

**Action** — Disables video, interrupts remain enabled, HRTC and VRTC continue to run, Video Enable status flag is reset, and the "Start Display" command must be given to re-enable the display.

4. Read Light Pen Command

	OPERATION	A <sub>0</sub>	DESCRIPTION	DATA BUS	
				MSB	LSB
Command	Write	1	Read Light Pen	0 1 1 0 0 0 0 0	
Parameters	Read	0	Char. Number	(Char. Position in Row)	
	Read	0	Row Number	(Row Number)	

**Action** — The 8275 is conditioned to supply the contents of the light pen position registers in the next two read cycles of the parameter register. Status flags are not affected.

**Note:** Software correction of light pen position is required.

5. Load Cursor Position:

	OPERATION	A <sub>0</sub>	DESCRIPTION	DATA BUS	
				MSB	LSB
Command	Write	1	Load Cursor	1 0 0 0 0 0 0 0	
Parameters	Write	0	Char. Number	(Char. Position in Row)	
	Write	0	Row Number	(Row Number)	

**Action** — The 8275 is conditioned to place the next two parameter bytes into the cursor position registers. Status flags not affected.

6. Enable Interrupt Command:

	OPERATION	A <sub>0</sub>	DESCRIPTION	DATA BUS	
				MSB	LSB
Command	Write	1	Enable Interrupt	1 0 1 0 0 0 0 0	
No parameters					

**Action** — The interrupt enable status flag is set and interrupts are enabled.

7. Disable Interrupt Command:

	OPERATION	A <sub>0</sub>	DESCRIPTION	DATA BUS	
				MSB	LSB
Command	Write	1	Disable Interrupt	1 1 0 0 0 0 0 0	
No parameters					

**Action** — Interrupts are disabled and the interrupt enable status flag is reset.

8. Preset Counters Command:

	OPERATION	A <sub>0</sub>	DESCRIPTION	DATA BUS	
				MSB	LSB
Command	Write	1	Preset Counters	1 1 1 0 0 0 0 0	
No parameters					

**Action** — The internal timing counters are preset, corresponding to a screen display position at the top left corner. Two character clocks are required for this operation. The counters will remain in this state until any other command is given.

This command is useful for system debug and synchronization of clustered CRT displays on a single CPU.

**Status Flags**

Command	OPERATION	A <sub>0</sub>	DESCRIPTION	DATA BUS							
				MSB							LSB
Command	Read	1	Status Word	0	IE	IR	LP	IC	VE	OU	FO

- IE – (Interrupt Enable) Set or reset by command. It enables vertical retrace interrupt. It is automatically set by a “Start Display” command and reset with the “Reset” command.
- IR – (Interrupt Request) This flag is set at the beginning of display of the last row of the frame if the interrupt enable flag is set. It is reset after a status read operation.
- LP – This flag is set when the light pen input (LPEN) is activated and the light pen registers have been loaded. This flag is automatically reset after a status read.

- IC – (Improper Command) This flag is set when a command parameter string is too long or too short. The flag is automatically reset after a status read.
- VE – (Video Enable) This flag indicates that video operation of the CRT is enabled. This flag is set on a “Start Display” command, and reset on a “Stop Display” or “Reset” command.
- DU – (DMA Underrun) This flag is set whenever a data underrun occurs during DMA transfers. Upon detection of DU, the DMA operation is stopped and the screen is blanked until after the vertical retrace interval. This flag is reset after a status read.
- FO – (FIFO Overrun) This flag is set whenever the FIFO is overrun. It is reset on a status read.

## ABSOLUTE MAXIMUM RATINGS\*

Ambient Temperature Under Bias . . . . . 0°C to 70°C  
 Storage Temperature . . . . . -65°C to +150°C  
 Voltage On Any Pin  
     With Respect to Ground . . . . . -0.5V to +7V  
 Power Dissipation . . . . . 1 Watt

*\*NOTICE: Stresses above those listed under "Absolute Maximum Ratings" may cause permanent damage to the device. This is a stress rating only and functional operation of the device at these or any other conditions above those indicated in the operational sections of this specification is not implied.*

## D.C. CHARACTERISTICS (T<sub>A</sub> = 0°C to 70°C, V<sub>CC</sub> = 5V ±5%)

Symbol	Parameter	Min.	Max.	Units	Test Conditions
V <sub>IL</sub>	Input Low Voltage	-0.5	0.8	V	
V <sub>IH</sub>	Input High Voltage	2.0	V <sub>CC</sub> +0.5V	V	
V <sub>OL</sub>	Output Low Voltage		0.45	V	I <sub>OL</sub> = 2.2 mA
V <sub>OH</sub>	Output High Voltage	2.4		V	I <sub>OH</sub> = -400 μA
I <sub>IL</sub>	Input Load Current		±10	μA	V <sub>IN</sub> = V <sub>CC</sub> to 0V
I <sub>OFL</sub>	Output Float Leakage		±10	μA	V <sub>OUT</sub> = V <sub>CC</sub> to 0V
I <sub>CC</sub>	V <sub>CC</sub> Supply Current		160	mA	

## CAPACITANCE (T<sub>A</sub> = 25°C, V<sub>CC</sub> = GND = 0V)

Symbol	Parameter	Min.	Max.	Units	Test Conditions
C <sub>IN</sub>	Input Capacitance		10	pF	f <sub>c</sub> = 1 MHz
C <sub>I/O</sub>	I/O Capacitance		20	pF	Unmeasured pins returned to V <sub>SS</sub> .

## A.C. CHARACTERISTICS (T<sub>A</sub> = 0°C to 70°C, V<sub>CC</sub> = 5.0V ±5%, GND = 0V)

### Bus Parameters

#### READ CYCLE

Symbol	Parameter	Min.	Max.	Units	Test Conditions
t <sub>AR</sub>	Address Stable Before READ	0		ns	
t <sub>RA</sub>	Address Hold Time for READ	0		ns	
t <sub>RR</sub>	READ Pulse Width	250		ns	
t <sub>RD</sub>	Data Delay from READ		200	ns	C <sub>L</sub> = 150 pF
t <sub>DF</sub>	READ to Data Floating	20	100	ns	C <sub>L</sub> min. = 20 pF; C <sub>L</sub> max. = 150 pF

#### WRITE CYCLE

Symbol	Parameter	Min.	Max.	Units	Test Conditions
t <sub>AW</sub>	Address Stable Before WRITE	0		ns	
t <sub>WA</sub>	Address Hold Time for WRITE	0		ns	
t <sub>WW</sub>	WRITE Pulse Width	250		ns	
t <sub>DW</sub>	Data Setup Time for WRITE	150		ns	
t <sub>WD</sub>	Data Hold Time for WRITE	0		ns	

#### CLOCK TIMING

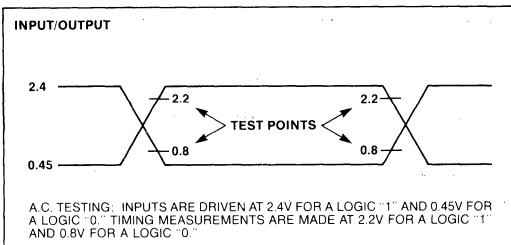
Symbol	Parameter	Min.	Max.	Units	Test Conditions
t <sub>CLK</sub>	Clock Period	480		ns	
t <sub>KH</sub>	Clock High	240		ns	
t <sub>KL</sub>	Clock Low	160		ns	
t <sub>KR</sub>	Clock Rise	5	30	ns	
t <sub>KF</sub>	Clock Fall	5	30	ns	

**A.C. CHARACTERISTICS (Continued)**

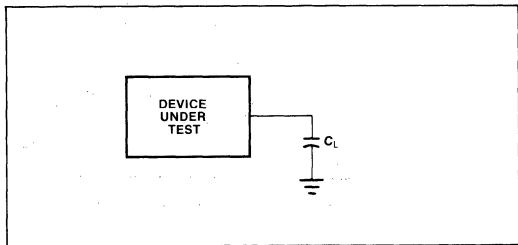
**OTHER TIMING**

Symbol	Parameter	Min.	Max.	Units	Test Conditions
t <sub>CC</sub>	Character Code Output Delay		150	ns	C <sub>L</sub> = 50 pF
t <sub>HR</sub>	Horizontal Retrace Output Delay		200	ns	C <sub>L</sub> = 50 pF
t <sub>LC</sub>	Line Count Output Delay		400	ns	C <sub>L</sub> = 50 pF
t <sub>AT</sub>	Control/Attribute Output Delay		275	ns	C <sub>L</sub> = 50 pF
t <sub>VR</sub>	Vertical Retrace Output Delay		275	ns	C <sub>L</sub> = 50 pF
t <sub>RI</sub>	IRQ↓ from RD↑		250	ns	C <sub>L</sub> = 50 pF
t <sub>WO</sub>	DRQ↑ from WR↑		250	ns	C <sub>L</sub> = 50 pF
t <sub>RO</sub>	DRQ↓ from WR↓		200	ns	C <sub>L</sub> = 50 pF
t <sub>LR</sub>	DACK↓ to WR↓	0		ns	
t <sub>RL</sub>	WR↑ to DACK↑	0		ns	
t <sub>PR</sub>	LPEN Rise		50	ns	
t <sub>PH</sub>	LPEN Hold	100		ns	

**A.C. TESTING INPUT, OUTPUT WAVEFORM**

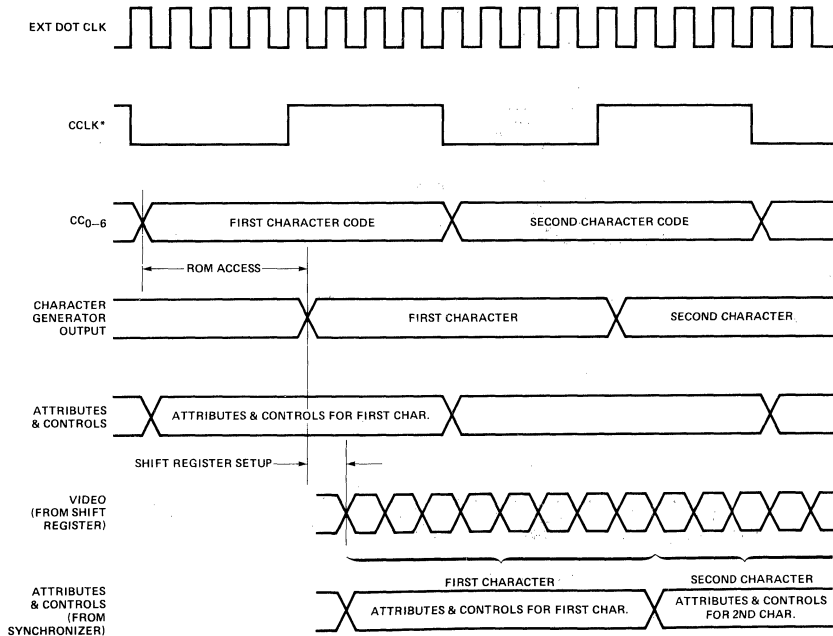


**A.C. TESTING LOAD CIRCUIT**



**WAVEFORMS**

**TYPICAL DOT LEVEL TIMING**

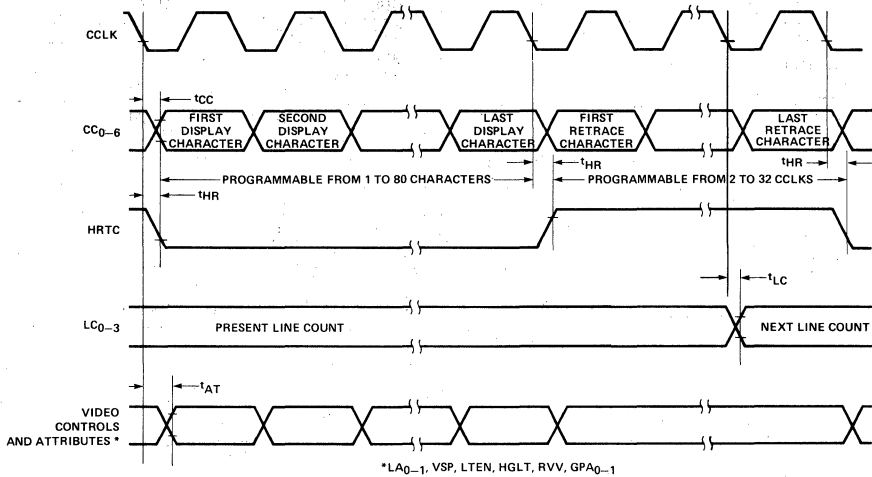


\*CCLK IS A MULTIPLE OF THE DOT CLOCK AND AN INPUT TO THE 8275.

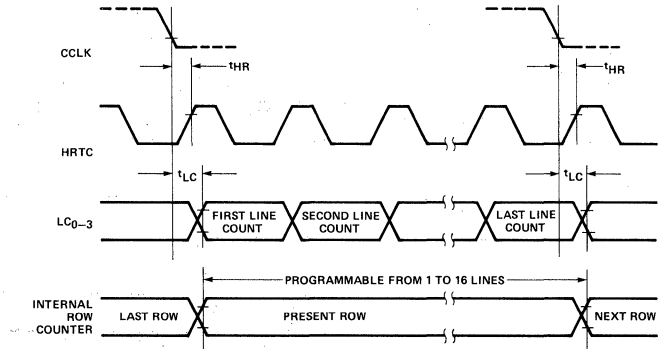


WAVEFORMS (Continued)

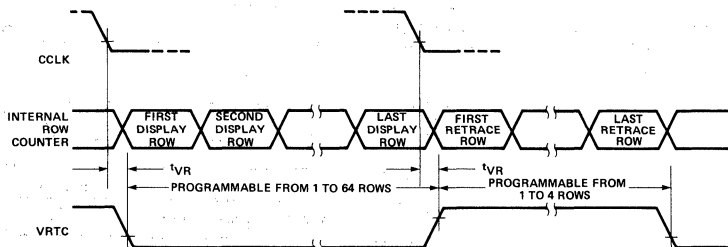
LINE TIMING



ROW TIMING

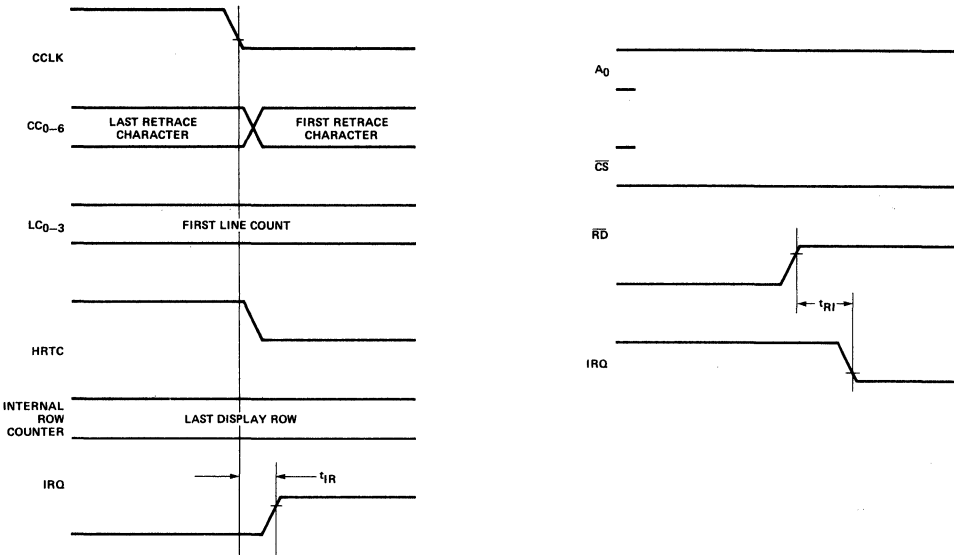


FRAME TIMING

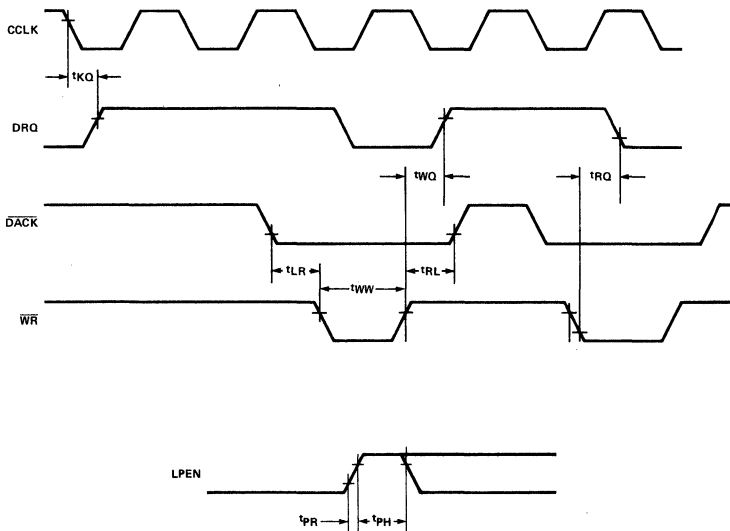


WAVEFORMS (Continued)

INTERRUPT TIMING

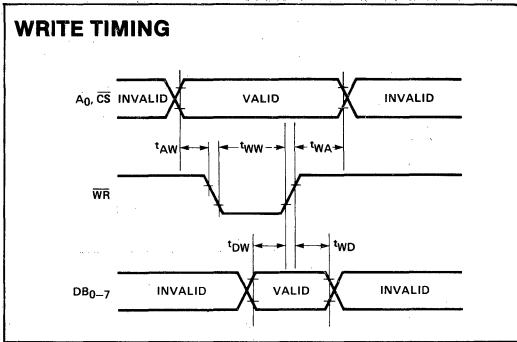


DMA TIMING

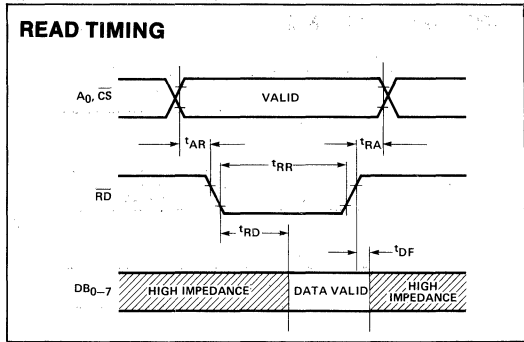


WAVEFORMS (Continued)

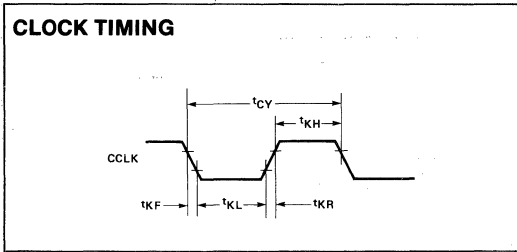
WRITE TIMING



READ TIMING



CLOCK TIMING



## 8276 SMALL SYSTEM CRT CONTROLLER

- Programmable Screen and Character Format
  - 6 Independent Visual Field Attributes
  - Cursor Control (4 Types)
- MCS-51®, MCS-85®, iAPX 86, and iAPX 88 Compatible
  - Dual Row Buffers
  - Single +5V Supply
  - 40-Pin Package

The Intel 8276 Small System CRT Controller is a single chip device intended to interface CRT raster scan displays with Intel microcomputers in minimum device-count systems. Its primary function is to refresh the display by buffering character information from main memory and keeping track of the display position of the screen. The flexibility designed into the 8276 will allow simple interface to almost any raster scan CRT display with a minimum system IC count.

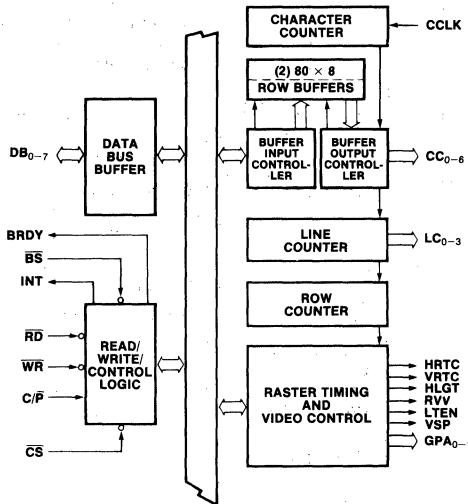


Figure 1. Block Diagram

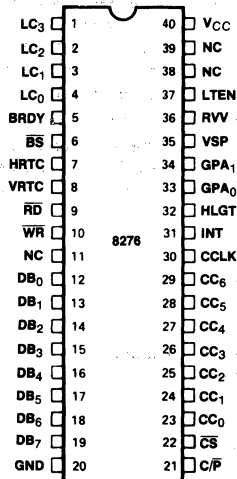


Figure 2. Pin Configuration

Table 1. Pin Descriptions

Symbol	Pin No.	Type	Name and Function
LC <sub>3</sub> LC <sub>2</sub> LC <sub>1</sub> LC <sub>0</sub>	1 2 3 4	O	<b>Line count.</b> Output from the line counter which is used to address the character generator for the line positions on the screen.
BRDY	5	O	<b>Buffer ready.</b> Output signal indicating that a Row Buffer is ready for loading of character data.
$\overline{BS}$	6	I	<b>Buffer select.</b> Input signal enabling WR for character data into the Row Buffers.
HRTC	7	O	<b>Horizontal retrace.</b> Output signal which is active during the programmed horizontal retrace interval. During this period the VSP output is high and the LTEN output is low.
VRTC	8	O	<b>Vertical retrace.</b> Output signal which is active during the programmed vertical retrace interval. During this period the VSP output is high and the LTEN output is low.
$\overline{RD}$	9	I	<b>Read input.</b> A control signal to read registers.
$\overline{WR}$	10	I	<b>Write input.</b> A control signal to write commands into the control registers or write data into the row buffers.
NC	11		<b>No connection.</b>
DB <sub>0</sub> DB <sub>1</sub> DB <sub>2</sub> DB <sub>3</sub> DB <sub>4</sub> DB <sub>5</sub> DB <sub>6</sub> DB <sub>7</sub>	12 13 14 15 16 17 18 19	I/O	<b>Bidirectional data bus.</b> Three-state lines. The outputs are enabled during a read of the C or P ports.
Ground	20		<b>Ground.</b>

Symbol	Pin No.	Type	Name and Function
V <sub>CC</sub>	40		<b>+5V power supply.</b>
NC	39		<b>No connection.</b>
NC	38		<b>No connection.</b>
LTEN	37	O	<b>Light enable.</b> Output signal used to enable the video signal to the CRT. This output is active at the programmed underline cursor position, and at positions specified by attribute codes.
RVV	36	O	<b>Reverse video.</b> Output signal used to activate the CRT circuitry to reverse the video signal. This output is active at the cursor position if a reverse video block cursor is programmed or at the positions specified by the field attribute codes.
VSP	35	O	<b>Video suppression.</b> Output signal used to blank the video signal to the CRT. This output is active: <ul style="list-style-type: none"> <li>— during the horizontal and vertical retrace intervals.</li> <li>— at the top and bottom lines of rows if underline is programmed to be number 8 or greater.</li> <li>— when an end of row or end of screen code is detected.</li> <li>— when a Row Buffer underrun occurs.</li> <li>— at regular intervals (1/16 frame frequency for cursor, 1/32 frame frequency for attributes)—to create blinking displays as specified by cursor or field attribute programming.</li> </ul>
GPA <sub>1</sub> GPA <sub>0</sub>	34 33	O	<b>General purpose attribute codes.</b> — Outputs which are enabled by the general purpose field attribute codes.
HLGT	32	O	<b>Highlight.</b> Output signal used to intensify the display at particular positions on the screen as specified by the field attribute codes.
INT	31	O	<b>Interrupt output.</b>
CCLK	30	I	<b>Character clock</b> (from dot/timing logic).
CC <sub>6</sub> CC <sub>5</sub> CC <sub>4</sub> CC <sub>3</sub> CC <sub>2</sub> CC <sub>1</sub> CC <sub>0</sub>	29 28 27 26 25 24 23	O	<b>Character codes.</b> Output from the row buffers used for character selection in the character generator.
$\overline{CS}$	22	I	<b>Chip select.</b> Enables $\overline{RD}$ of status or $\overline{WR}$ of command or parameters.
C/ P	21	I	<b>Port address.</b> A high input on this pin selects the "C" port or command registers and a low input selects the "P" port or parameter registers.

## FUNCTIONAL DESCRIPTION

### Data Bus Buffer

This 3-state, bidirectional, 8-bit buffer is used to interface the 8276 to the system Data Bus.

This functional block accepts inputs from the System Control Bus and generates control signals for overall device operation. It contains the Command, Parameter, and Status Registers that store the various control formats for the device functional definition.

C/ $\bar{P}$	OPERATION	REGISTER
0	Read	RESERVED
0	Write	PARAMETER
1	Read	STATUS
1	Write	COMMAND

### $\overline{RD}$ (READ)

A "low" on this input informs the 8276 that the CPU is reading status information from the 8276.

### $\overline{WR}$ (WRITE)

A "low" on this input informs the 8276 that the CPU is writing data or control words to the 8276.

### $\overline{CS}$ (CHIP SELECT)

A "low" on this input selects the 8276 for  $\overline{RD}$  or  $\overline{WR}$  of Commands, Status, and Parameters.

### BRDY (BUFFER READY)

A "high" on this output indicates that the 8276 is ready to receive character data.

### $\overline{BS}$ (BUFFER SELECT)

A "low" on this input enables  $\overline{WR}$  of character data to the 8276 row buffers.

### INT (INTERRUPT)

A "high" on this output informs the CPU that the 8276 needs interrupt service.

C/ $\bar{P}$	$\overline{RD}$	$\overline{WR}$	$\overline{CS}$	$\overline{BS}$	
0	0	1	0	1	Reserved
0	1	0	0	1	Write 8276 Parameter
1	0	1	0	1	Read 8276 Status
1	1	0	0	1	Write 8276 Command
X	1	0	1	0	Write 8276 Row Buffer
X	1	1	X	X	High Impedance
X	X	X	1	1	High Impedance

### Character Counter

The Character Counter is a programmable counter that is used to determine the number of characters to be displayed per row and the length of the horizontal retrace interval. It is driven by the CCLK (Character Clock) input, which should be derived from the external dot clock.

### Line Counter

The Line Counter is a programmable counter that is used to determine the number of horizontal lines (Raster Scans) per character row. Its outputs are used to address the external character generator.

### Row Counter

The Row Counter is a programmable counter that is used to determine the number of character rows to be displayed per frame and length of the vertical retrace interval.

### Raster Timing and Video Controls

The Raster Timing circuitry controls the timing of the HRTC (Horizontal Retrace) and VRTC (Vertical Retrace) outputs. The Video Control circuitry controls the generation of HGLT (Highlight), RVV (Reverse Video), LTEN (Light Enable), VSP (Video Suppress), and GPA<sub>0-1</sub> (General Purpose Attribute) outputs.

### Row Buffers

The Row Buffers are two 80-character buffers. They are filled from the microcomputer system memory with the character codes to be displayed. While one row buffer is displaying a row of characters, the other is being filled with the next row of characters.

### Buffer Input/Output Controllers

The Buffer Input/Output Controllers decode the characters being placed in the row buffers. If the character is a field attribute or special code, they control the appropriate action. (Example: A "Highlight" field attribute will cause the Buffer Output Controller to activate the HGLT output.)

**SYSTEM OPERATION**

The 8276 is programmable to a large number of different display formats. It provides raster timing, display row buffering, visual attribute decoding and cursor timing.

It is designed to interface with standard character generators for dot matrix decoding. Dot level timing must be provided by external circuitry.

**General Systems Operational Description**

Display characters are retrieved from memory and displayed on a row-by-row basis. The 8276 has two row buffers. While one row buffer is being used for display, the other is being filled with the next row of characters to be displayed. The number of display characters per row and the number of character rows per frame are software programmable, providing easy interface to most CRT displays. (See Programming Section.)

The 8276 uses BRDY to request character data to fill the row buffer that is not being used for display.

The 8276 displays character rows one scan line at a time. The number of scan lines per character row, the underline position, and blanking of top and bottom lines are programmable. (See Programming Section.)

The 8276 provides special Control Codes which can be used to minimize overhead. It also provides Visual Attribute Codes to cause special action on the screen without the use of the character generator. (See Visual Attributes Section.)

The 8276 also controls raster timing. This is done by generating Horizontal Retrace (HRTC) and Vertical Retrace (VRTC) signals. The timing of these signals is also programmable.

The 8276 can generate a cursor. Cursor location and format are programmable. (See Programming Section.)

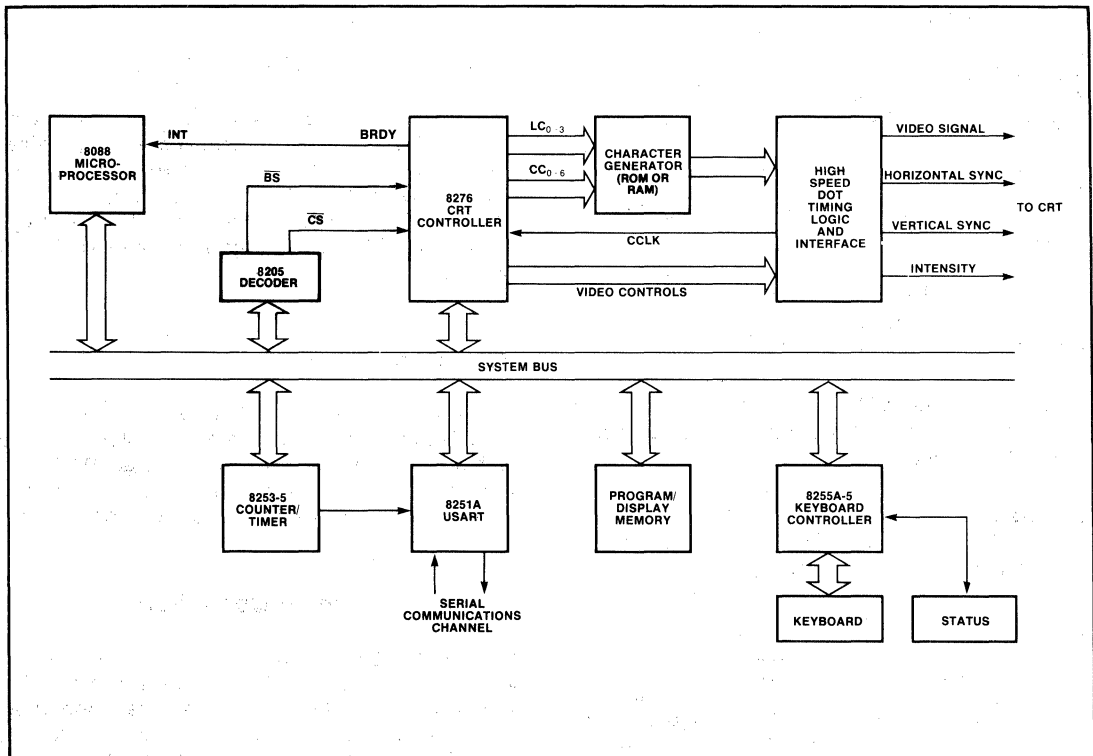


Figure 3. CRT System Block Diagram

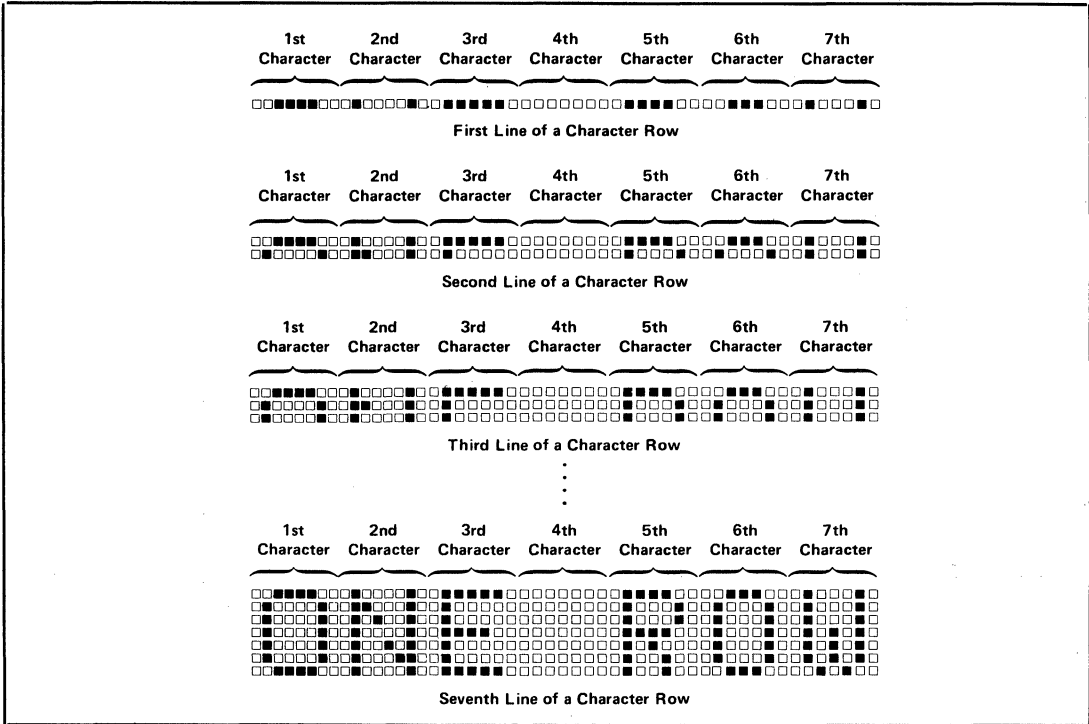


Figure 4. Display Of A Character Row

### Display Row Buffering

Before the start of a frame, the 8276 uses BRDY and BS to fill one row buffer with characters.

When the first horizontal sweep is started, character codes are output to the character generator from the row buffer just filled. Simultaneously, the other row buffer is filled with the next row of characters.

After all the lines of the character row are scanned, the buffers are swapped and the same procedure is followed for the next row.

This process is repeated until all of the character rows are displayed.

Row Buffering allows the CPU access to the display memory at all times except during Buffer Loading (about 25%). This compares favorably to alternative approaches which restrict CPU access to the display memory to occur only during horizontal and vertical retrace intervals (80% of the bus time is used to refresh the display.)

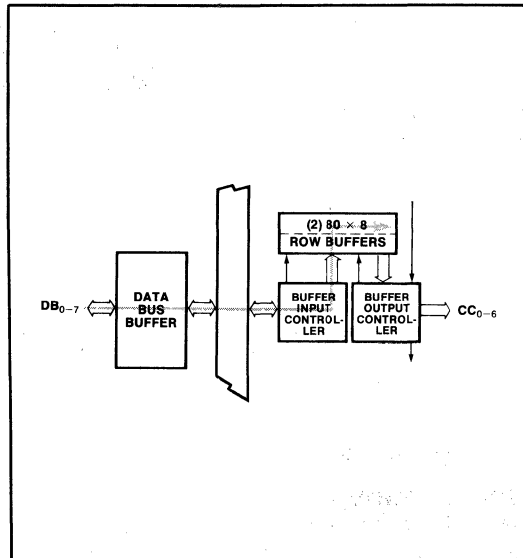


Figure 5. First Row Buffer Filled



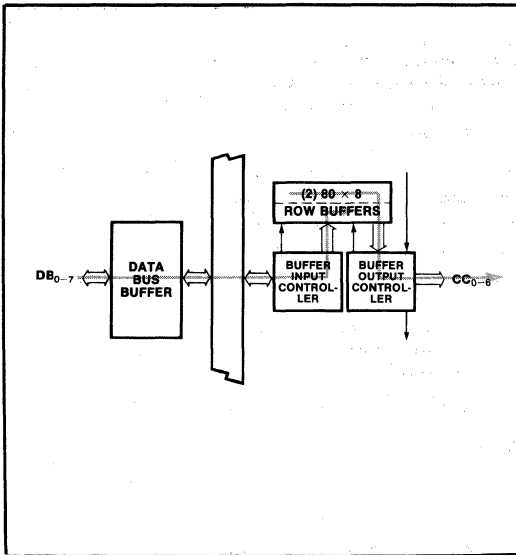


Figure 6. Second Row Buffer Filled, First Row Displayed

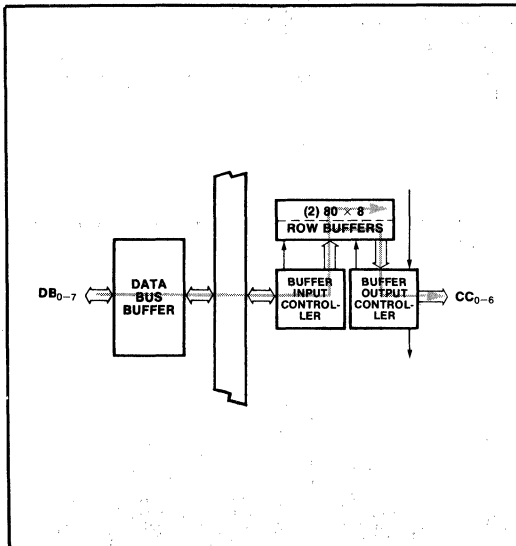


Figure 7. First Row Buffer Filled With Third Row, Second Row Displayed

**Display Format**

**SCREEN FORMAT**

The 8276 can be programmed to generate from 1 to 80 characters per row, and from 1 to 64 rows per frame.

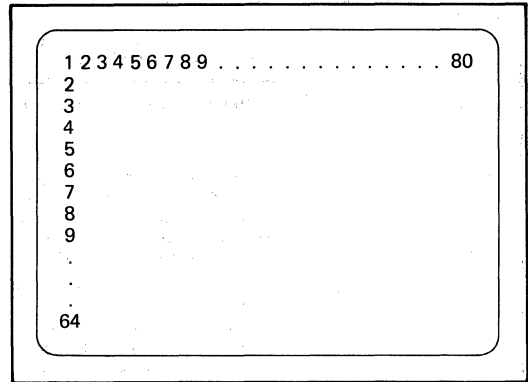


Figure 8. Screen Format

The 8276 can also be programmed to blank alternate rows. In this mode, the first row is displayed, the second blanked, the third displayed, etc. Display data is not requested for the blanked rows.

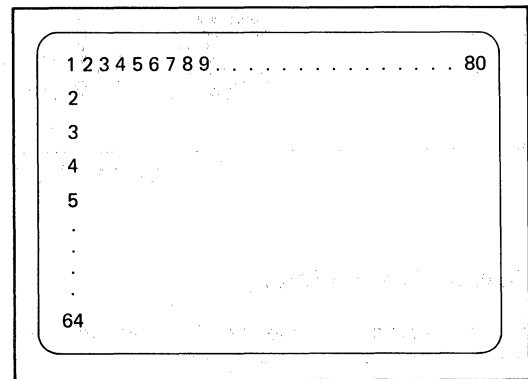


Figure 9. Blank Alternate Rows Mode

**ROW FORMAT**

The 8276 is designed to hold the line count stable while outputting the appropriate character codes during each horizontal sweep. The line count is incremented during horizontal retrace and the whole row of character codes are output again during the next sweep. This is continued until the entire character row is displayed.

The number of lines (horizontal sweeps) per character row is programmable from 1 to 16.

The output of the line counter can be programmed to be in one of two modes.

In mode 0, the output of the line counter is the same as the line number.

In mode 1, the line counter is offset by one from the line number.

**Note:** In mode 1, while the first line (line number 0) is being displayed, the last count is output by the line counter (see examples).

Line Number	Line Counter Mode 0	Line Counter Mode 1
0	0000	1111
1	0001	0000
2	0010	0001
3	0011	0010
4	0100	0011
5	0101	0100
6	0110	0101
7	0111	0110
8	1000	0111
9	1001	1000
10	1010	1001
11	1011	1010
12	1100	1011
13	1101	1100
14	1110	1101
15	1111	1110

Figure 10. Example of a 16-Line Format

Line Number	Line Counter Mode 0	Line Counter Mode 1
0	0000	1001
1	0001	0000
2	0010	0001
3	0011	0010
4	0100	0011
5	0101	0100
6	0110	0101
7	0111	0110
8	1000	0111
9	1001	1000

Figure 11. Example of a 10-Line Format

Mode 0 is useful for character generators that leave address zero blank and start at address 1. Mode 1 is useful for character generators which start at address zero.

Underline placement is also programmable (from line number 0 to 15). This is independent of the line counter mode.

If the line number of the underline is greater than 7 (line number MSB = 1), then the top and bottom lines will be blanked.

Line Number	Line Counter Mode 0	Line Counter Mode 1
0	0000	1011
1	0001	0000
2	0010	0001
3	0011	0010
4	0100	0011
5	0101	0100
6	0110	0101
7	0111	0110
8	1000	0111
9	1001	1000
10	1010	1001
11	1011	1010

Top and Bottom Lines are Blanked

Figure 12. Underline in Line Number 10

If the line number of the underline is less than or equal to 7 (line number MSB = 0), then the top and bottom lines will not be blanked.

Line Number	Line Counter Mode 0	Line Counter Mode 1
0	0000	0111
1	0001	0000
2	0010	0001
3	0011	0010
4	0100	0011
5	0101	0100
6	0110	0101
7	0111	0110

Top and Bottom Lines are not Blanked

Figure 13. Underline in Line Number 7

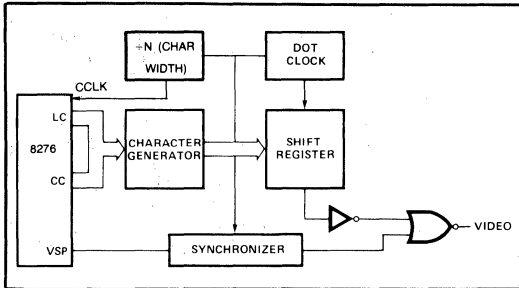
If the line number of the underline is greater than the maximum number of lines, the underline will not appear.

Blanking is accomplished by the VSP (Video Suppression) signal. Underline is accomplished by the LTEN (Light Enable) signal.

**DOT FORMAT**

Dot width and character width are dependent upon the external timing and control circuitry.

Dot level timing circuitry should be designed to accept the parallel output of the character generator and shift it out serially at the rate required by the CRT display.



**Figure 14. Typical Dot Level Block Diagram**

Dot width is a function of dot clock frequency.

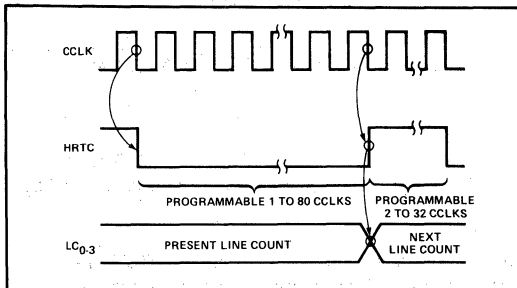
Character width is a function of the character generator width.

Horizontal character spacing is a function of the shift register length.

**Note:** Video control and timing signals must be synchronized with the video signal due to the character generator access delay.

**Raster Timing**

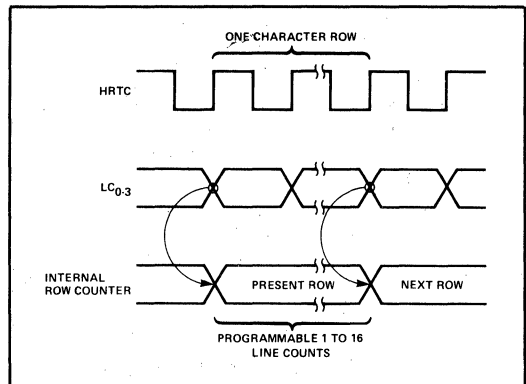
The character counter is driven by the character clock input (CCLK). It counts out the characters being displayed (programmable from 1 to 80). It then causes the line counter to increment, and it starts counting out the horizontal retrace interval (programmable from 2 to 32). This process is constantly repeated.



**Figure 15. Line Timing**

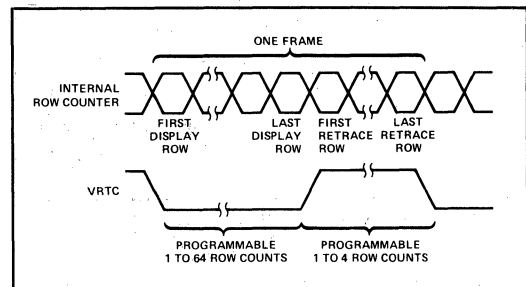
The line counter is driven by the character counter. It is used to generate the line address outputs (LC<sub>0-3</sub>) for the character generator. After it counts all of the lines in a character row (programmable from 1 to 16), it increments the row counter, and starts over again. (See Character Format Section for detailed description of Line Counter functions.)

The row counter is an internal counter driven by the line counter. It controls the functions of the row buffers and counts the number of character rows displayed.



**Figure 16. Row Timing**

After the row counter counts all of the rows in a frame (programmable from 1 to 64), it starts counting out the vertical retrace interval (programmable from 1 to 4).



**Figure 17. Frame Timing**

The Video Suppression Output (VSP) is active during horizontal and vertical retrace intervals.

Dot level timing circuitry must synchronize these outputs with the video signal to the CRT Display.

## Interrupt Timing

The 8276 can be programmed to generate an interrupt request at the end of each frame. If the 8276 interrupt enable flag is set, an interrupt request will occur at the *beginning* of the *last display row*.

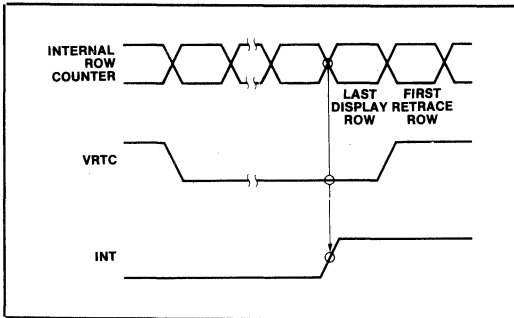


Figure 18. Beginning of Interrupt

INT will go inactive after the status register is read.

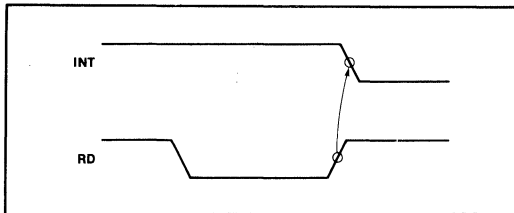


Figure 19. End of Interrupt

A reset command will also cause INT to go inactive, but this is not recommended during normal service.

**Note:** Upon power-up, the 8276 Interrupt Enable Flag may be set. As a result, the user's cold start routine should write a reset command to the 8276 before system interrupts are enabled.

## VISUAL ATTRIBUTES AND SPECIAL CODES

The characters processed by the 8276 are 8-bit quantities. The character code outputs provide the character generator with 7 bits of address. The Most Significant Bit is the extra bit and it is used to determine if it is a normal display character (MSB = 0), or if it is a Field Attribute or Special Code (MSB = 1).

## Special Codes

Four special codes are available to help reduce bus usage.

### SPECIAL CONTROL CHARACTER

MSB  
1 1 1 1 0 0 S S  
LSB  
SPECIAL CONTROL CODE

S	S	FUNCTION
0	0	End of Row
0	1	End of Row-Stop Buffer Loading
1	0	End of Screen
1	1	End of Screen-Stop Buffer Loading

The End of Row Code (00) activates VSP and holds it to the end of the line.

The End of Row-Stop Buffer Loading (BRDY) Code (01) causes the Buffer Loading Control Logic to stop buffer loading for the rest of the row upon being written into the Row Buffer. It affects the display in the same way as the End of Row Code (00).

The End of Screen Code (10) activates VSP and holds it to the end of the frame.

The End of Screen-Stop Buffer Loading (BRDY) Code (11) causes the Row Buffer Control Logic to stop buffer loading for the rest of the frame upon being written. It affects the display in the same way as the End of Screen Code (10).

If the Stop Buffer Loading feature is not used, all characters after an End of Row character are ignored, except for the End of Screen character, which operates normally. All characters after an End of Screen character are ignored.

**Note:** If a Stop Buffer Loading is not the last character in a row, Buffer Loading is not stopped until after the next character is read. In this situation, a dummy character must be placed in memory after the Stop Buffer Loading character.

## Field Attributes

The field attributes are control codes which affect the visual characteristics for a field of characters, starting at the character following the code up to, and including, the character which precedes the *next* field attribute code, or up to the end of the frame. The field attributes are reset during the vertical retrace interval.

The 8276 can be programmed to provide visible field attribute characters; all field attribute codes will occupy a position on the screen. These codes will appear as blanks caused by activation of the Video Suppression output (VSP). The chosen visual attributes are activated after this blanked character.

There are six field attributes:

1. *Blink*—Characters following the code are caused to blink by activating the Video Suppression output (VSP). The blink frequency is equal to the screen refresh frequency divided by 32.
2. *Highlight*—Characters following the code are caused to be highlighted by activating the Highlight output (HGLT).
3. *Reverse Video*—Characters following the code are caused to appear with reverse video by activating the Reverse Video output (RVV).
4. *Underline*—Characters following the code are caused to be underlined by activating the Light Enable output (LTEN).
- 5,6. *General Purpose*—There are two additional 8276 outputs which act as general purpose, independently programmable field attributes.  $GPA_{0-1}$  are active high outputs.

- H = 1 FOR HIGHLIGHTING
- B = 1 FOR BLINKING
- R = 1 FOR REVERSE VIDEO
- U = 1 FOR UNDERLINE
- GG =  $GPA_1, GPA_0$

**Note:** More than one attribute can be enabled at the same time. If the blinking and reverse video attributes are enabled simultaneously, only the reversed characters will blink.

### Cursor Timing

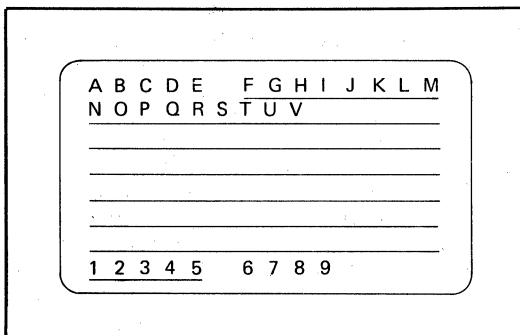
The cursor location is determined by a cursor row register and a character position register which are loaded by command to the controller. The cursor can be programmed to appear on the display as:

1. a blinking underline
2. a blinking reverse video block
3. a non-blinking underline
4. a non-blinking reverse video block

The cursor blinking frequency is equal to the screen refresh frequency divided by 16.

If a non-blinking reverse video *cursor* appears in a non-blinking reverse video *field*, the cursor will appear as a normal video block.

If a non-blinking underline *cursor* appears in a non-blinking underline *field*, the cursor will not be visible.



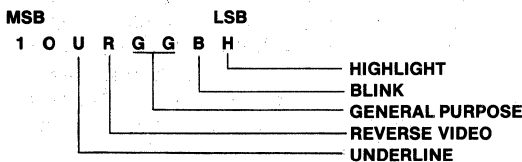
**Figure 20. Example of a Visible Field Attribute (Underline Attribute)**

### Device Programming

The 8276 has two programming registers, the Command Register and the Parameter Register. It also has a Status Register. The Command Register can only be written into and the Status Register can only be read from. They are addressed as follows:

C/P	OPERATION	REGISTER
0	Read	Reserved
0	Write	Parameter
1	Read	Status
1	Write	Command

#### FIELD ATTRIBUTE CODE



The 8276 expects to receive a command and a sequence of 0 to 4 parameters, depending on the command. If the proper number of parameter bytes are not received before another command is given, a status flag is set, indicating an improper command.

**Instruction Set**

The 8276 instruction set consists of 7 commands.

COMMAND	NO. OF PARAMETER BYTES
Reset	4
Start Display	0
Stop Display	0
Load Cursor	2
Enable Interrupt	0
Disable Interrupt	0
Preset Counters	0

In addition, the status of the 8276 can be read by the CPU at any time.

**1. RESET COMMAND**

Command	OPERATION	C/P	DESCRIPTION	DATA BUS	
				MSB	LSB
Parameters	Write	1	Reset Command	0	0 0 0 0 0 0 0 0
	Write	0	Screen Comp Byte 1	S	H H H H H H H H
	Write	0	Screen Comp Byte 2	V	V R R R R R R R
	Write	0	Screen Comp Byte 3	U	U U U U L L L L
	Write	0	Screen Comp Byte 4	M	1 C C Z Z Z Z Z

**Action**—After the reset command is written, BRDY goes inactive, 8276 interrupts are disabled, and the VSP output is used to blank the screen. HRTC and VRTC continue to run. HRTC and VRTC timing are random on power-up

As parameters are written, the screen composition is defined.

**Parameter—S Spaced Rows**

S	FUNCTIONS
0	Normal Rows
1	Spaced Rows

**Parameter—HHHHHHH Horizontal Characters/Row**

H H H H H H H H	NO. OF CHARACTERS PER ROW
0 0 0 0 0 0 0 0	1
0 0 0 0 0 0 0 1	2
0 0 0 0 0 0 1 0	3
.	.
.	.
1 0 0 1 1 1 1 1	80
1 0 1 0 0 0 0 0	Undefined
.	.
.	.
1 1 1 1 1 1 1 1	Undefined

**Parameter—VV Vertical Retrace Row Count**

V V	NO. OF ROW COUNTS PER VRTC
0 0	1
0 1	2
1 0	3
1 1	4

**Parameter—RRRRRR Vertical Rows/Frame**

R R R R R R	NO. OF ROWS/FRAME
0 0 0 0 0 0	1
0 0 0 0 0 1	2
0 0 0 0 1 0	3
.	.
.	.
1 1 1 1 1 1	64

**Parameter—UUUU Underline Placement**

U U U U	LINE NUMBER OF UNDERLINE
0 0 0 0	1
0 0 0 1	2
0 0 1 0	3
.	.
.	.
1 1 1 1	16

**Parameter—LLLL Number of Lines per Character Row**

L L L L	NO. OF LINES/ROW
0 0 0 0	1
0 0 0 1	2
0 0 1 0	3
.	.
.	.
1 1 1 1	16

**Parameter—M Line Counter Mode**

M	LINE COUNTER MODE
0	Mode 0 (Non-Offset)
1	Mode 1 (Offset by 1 Count)

**Parameter—CC Cursor Format**

C C	CURSOR FORMAT
0 0	Blinking reverse video block
0 1	Blinking underline
1 0	Non-blinking reverse video block
1 1	Non-blinking underline

**Parameter—ZZZZ Horizontal Retrace Count**

Z Z Z Z	NO. OF CHARACTER COUNTS PER HRTC
0 0 0 0	2
0 0 0 1	4
0 0 1 0	6
1 1 1 1	32

**Note:** uuuu MSB determines blanking of top and bottom lines (1 = blanked, 0 = not blanked).

**2. START DISPLAY COMMAND**

	OPERATION	C/P	DESCRIPTION	MSB	DATA BUS	LSB
Command	Write	1	Start Display	0	0 1 0 0 0 0 0	0
No parameters						

**Action**—8276 interrupts are enabled, BRDY goes active, video is enabled, Interrupt Enable and Video Enable status flags are set.

**3. STOP DISPLAY COMMAND**

	OPERATION	C/P	DESCRIPTION	MSB	DATA BUS	LSB
Command	Write	1	Stop Display	0	1 0 0 0 0 0 0	0
No parameters						

**Action**—Disables video, interrupts remain enabled, HRTC and VRTC continue to run, Video Enable status flag is reset, and the "Start Display" command must be given to reenable the display.

**4. LOAD CURSOR POSITION**

	OPERATION	C/P	DESCRIPTION	MSB	DATA BUS	LSB
Command	Write	1	Load Cursor	1	0 0 0 0 0 0 0	0
Parameters	Write	0	Char. Number	(Char. Position in Row)		
	Write	0	Row Number	(Row Number)		

**Action**—The 8276 is conditioned to place the next two parameter bytes into the cursor position registers. Status flag not affected.

**5. ENABLE INTERRUPT COMMAND**

	OPERATION	C/P	DESCRIPTION	MSB	DATA BUS	LSB
Command	Write	1	Enable Interrupt	1	0 1 0 0 0 0 0	0
No parameters						

**Action**—The interrupt enable flag is set and interrupts are enabled.

**6. DISABLE INTERRUPT COMMAND**

	OPERATION	C/P	DESCRIPTION	MSB	DATA BUS	LSB
Command	Write	1	Disable Interrupt	1	1 0 0 0 0 0 0	0
No parameters						

**Action**—Interrupts are disabled and the interrupt enable status flag is reset.

**7. PRESET COUNTERS COMMAND**

	OPERATION	C/P	DESCRIPTION	MSB	DATA BUS	LSB
Command	Write	1	Preset Counters	1	1 1 0 0 0 0 0	0
No parameters						

**Action**—The internal timing counters are preset, corresponding to a screen display position at the top left corner. Two character clocks are required for this operation. The counters will remain in this state until any other command is given.

This command is useful for system debug and synchronization of clustered CRT displays on a single CPU.

**Status Flags**

	OPERATION	C/P	DESCRIPTION	MSB	DATA BUS	LSB
Command	Read	1	Status Word	0	IE IR X IC VE BU X	

- IE — (Interrupt Enable) Set or reset by command. It enables vertical retrace interrupt. It is automatically set by a "Start Display" command and reset with the "Reset" command.
- IR — (Interrupt Request) This flag is set at the beginning of display of the last row of the frame if the interrupt enable flag is set. It is reset after a status read operation.
- IC — (Improper Command) This flag is set when a command parameter string is too long or too short. The flag is automatically reset after a status read.
- VE — (Video Enable) This flag indicates that video operation of the CRT is enabled. This flag is set on a "Start Display" command, and reset on a "Stop Display" or "Reset" command.
- BU — (Buffer Underrun) This flag is set whenever a Row Buffer is not filled with character data in time for a buffer swap required by the display. Upon activation of this bit, buffer loading ceases, and the screen is blanked until after the vertical retrace interval.

**ABSOLUTE MAXIMUM RATINGS\***

Ambient Temperature Under Bias . . . . 0°C to 70°C  
 Storage Temperature . . . . . -65°C to +150°C  
 Voltage On Any Pin  
     With Respect to Ground . . . . . -0.5V to +7V  
 Power Dissipation . . . . . 1 Watt

*\*NOTICE:* Stresses above those listed under "Absolute Maximum Ratings" may cause permanent damage to the device. This is a stress rating only and functional operation of the device at these or any other conditions above those indicated in the operational sections of this specification is not implied.

**D.C. CHARACTERISTICS** ( $T_A = 0^\circ\text{C}$  to  $70^\circ\text{C}$ ;  $V_{CC} = 5V \pm 5\%$ )

SYMBOL	PARAMETER	MIN.	MAX.	UNITS	TEST CONDITIONS
$V_{IL}$	Input Low Voltage	-0.5	0.8	V	
$V_{IH}$	Input High Voltage	2.0	$V_{CC} + 0.5V$	V	
$V_{OL}$	Output Low Voltage		0.45	V	$I_{OL} = 2.2 \text{ mA}$
$V_{OH}$	Output High Voltage	2.4		V	$I_{OH} = -400 \mu\text{A}$
$I_{IL}$	Input Load Current		$\pm 10$	$\mu\text{A}$	$V_{IN} = V_{CC}$ to 0V
$I_{OFL}$	Output Float Leakage		$\pm 10$	$\mu\text{A}$	$V_{OUT} = V_{CC}$ to 0.45V
$I_{CC}$	$V_{CC}$ Supply Current		160	mA	

**CAPACITANCE** ( $T_A = 25^\circ\text{C}$ ;  $V_{CC} = \text{GND} = 0V$ )

SYMBOL	PARAMETER	MIN.	MAX.	UNITS	TEST CONDITIONS
$C_{IN}$	Input Capacitance		10	pF	$f_C = 1 \text{ MHz}$
$C_{I/O}$	I/O Capacitance		20	pF	Unmeasured pins returned to $V_{SS}$ .



**A.C. CHARACTERISTICS** ( $T_A = 0^\circ\text{C}$  to  $70^\circ\text{C}$ ;  $V_{CC} = 5.0\text{V} \pm 5\%$ ;  $GND = 0\text{V}$ )**Bus Parameters (Note 1)****READ CYCLE**

SYMBOL	PARAMETER	MIN.	MAX.	UNITS	TEST CONDITIONS
$t_{AR}$	Address Stable Before READ	0		ns	
$t_{RA}$	Address Hold Time for READ	0		ns	
$t_{RR}$	READ Pulse Width	250		ns	
$t_{RD}$	Data Delay from READ		200	ns	$C_L = 150\text{pF}$
$t_{DF}$	READ to Data Floating	20	100	ns	

**WRITE CYCLE**

SYMBOL	PARAMETER	MIN.	MAX.	UNITS	TEST CONDITIONS
$t_{AW}$	Address Stable Before WRITE	0		ns	
$t_{WA}$	Address Hold Time for WRITE	0		ns	
$t_{WW}$	WRITE Pulse Width	250		ns	
$t_{DW}$	Data Setup Time for WRITE	150		ns	
$t_{WD}$	Data Hold Time for WRITE	0		ns	

**CLOCK TIMING**

SYMBOL	PARAMETER	MIN.	MAX.	UNITS	TEST CONDITIONS
$t_{CLK}$	Clock Period	480		ns	
$t_{KH}$	Clock High	240		ns	
$t_{KL}$	Clock Low	160		ns	
$t_{KR}$	Clock Rise	5	30	ns	
$t_{KF}$	Clock Fall	5	30	ns	

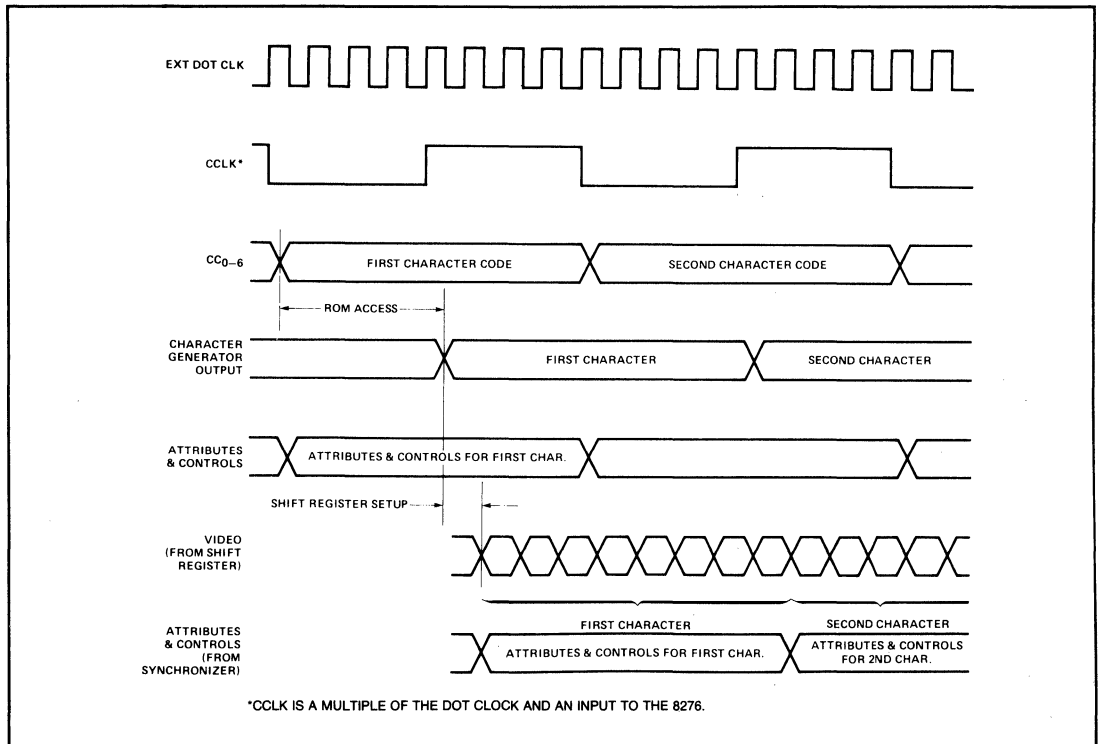
Note 1: AC timings measured at  $V_{OH} = 2.0$ ,  $V_{OL} = 0.8$ ,  $V_{IH} = 2.4$ ,  $V_{IL} = 0.45$ .

**OTHER TIMING**

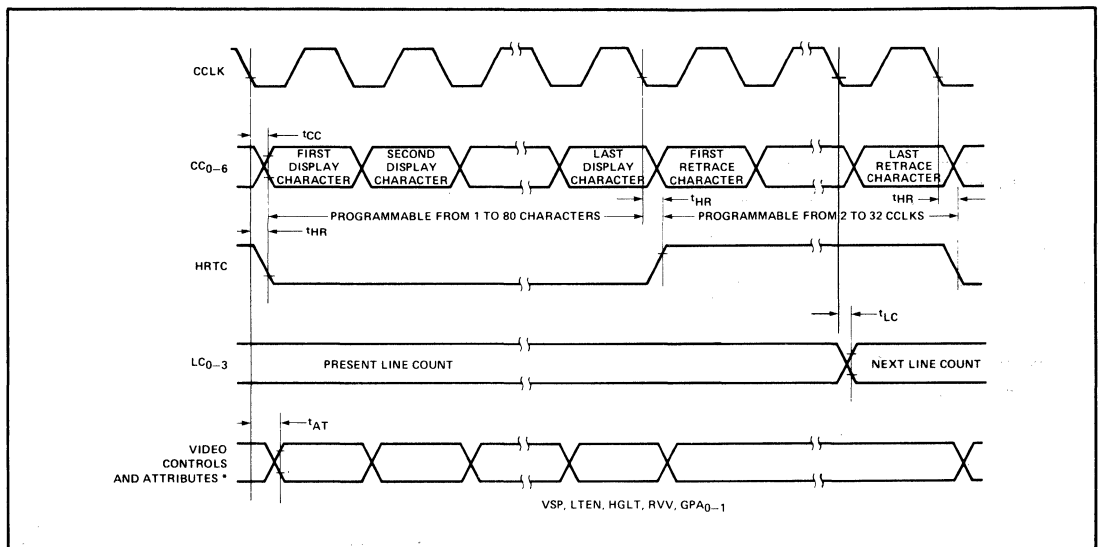
SYMBOL	PARAMETER	MIN.	MAX.	UNITS	TEST CONDITIONS
$t_{CC}$	Character Code Output Delay		150	ns	$C_L = 50\text{ pF}$
$t_{HR}$	Horizontal Retrace Output Delay		200	ns	$C_L = 50\text{ pF}$
$t_{LC}$	Line Count Output Delay		400	ns	$C_L = 50\text{ pF}$
$t_{AT}$	Control/Attribute Output Delay		275	ns	$C_L = 50\text{ pF}$
$t_{VR}$	Vertical Retrace Output Delay		275	ns	$C_L = 50\text{ pF}$
$t_{RI}$	$\text{INT}\downarrow$ from $\text{RD}\uparrow$		250	ns	$C_L = 50\text{ pF}$
$t_{WQ}$	$\text{BRDY}\uparrow$ from $\text{WR}\uparrow$		250	ns	$C_L = 50\text{ pF}$
$t_{RQ}$	$\text{BRDY}\downarrow$ from $\text{WR}\downarrow$		200	ns	$C_L = 50\text{ pF}$
$t_{LR}$	$\text{BS}\downarrow$ to $\text{WR}\downarrow$	0		ns	
$t_{RL}$	$\text{WR}\uparrow$ to $\text{BS}\uparrow$	0		ns	

WAVEFORMS

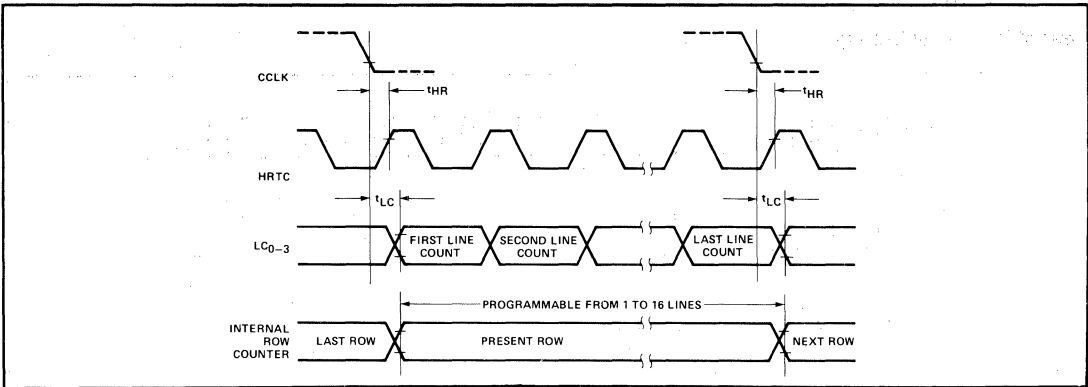
Typical Dot Level Timing



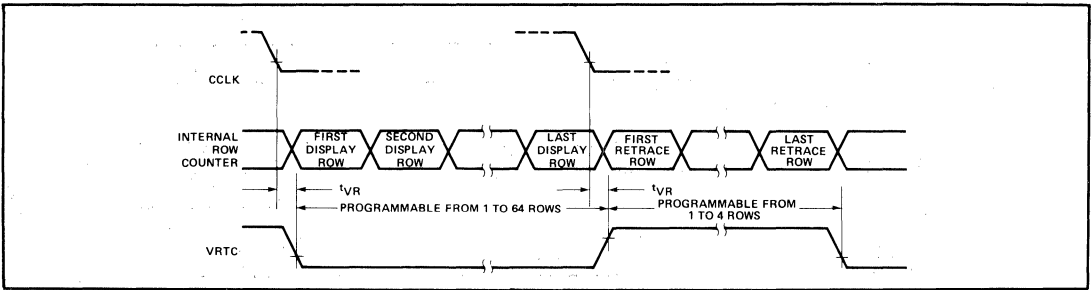
Line Timing



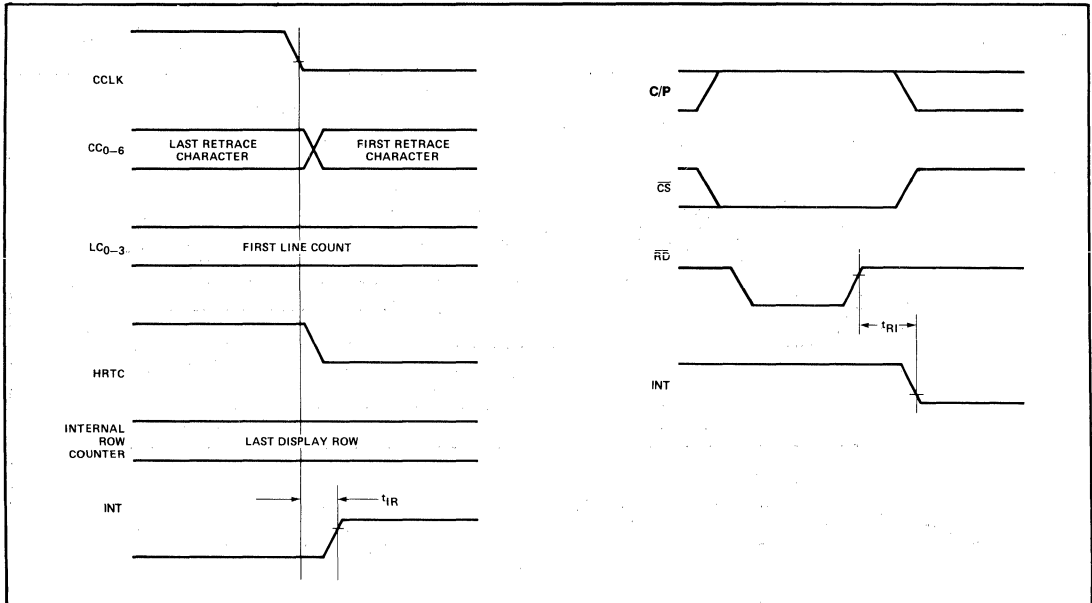
Row Timing



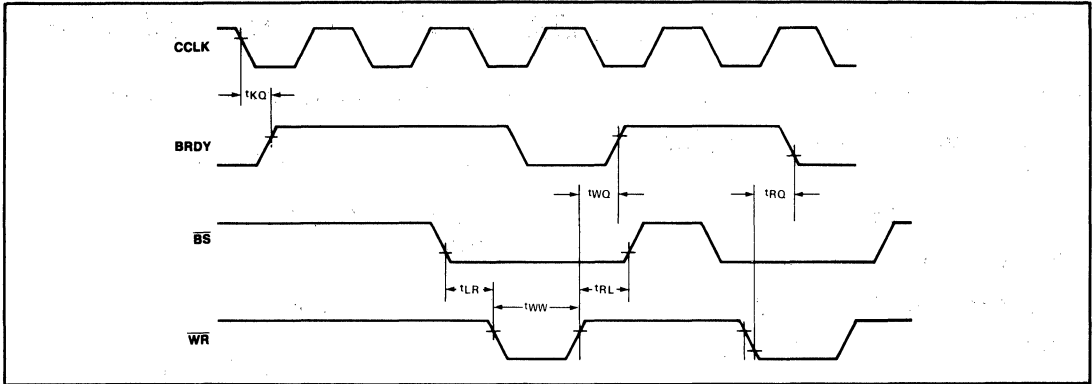
Frame Timing



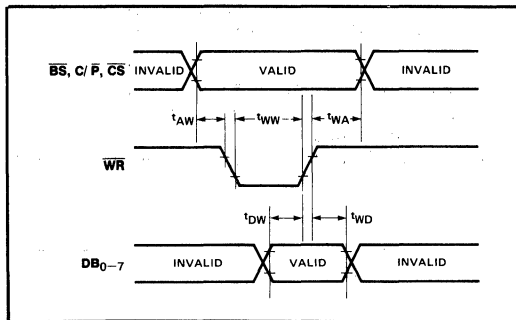
Interrupt Timing



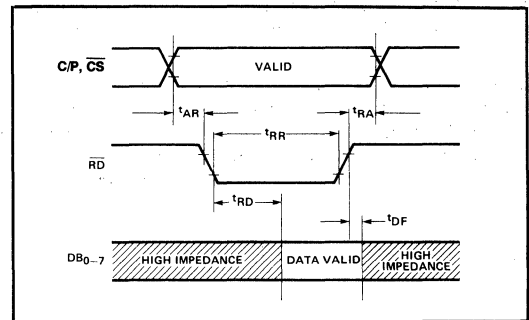
**Timing for Buffer Loading**



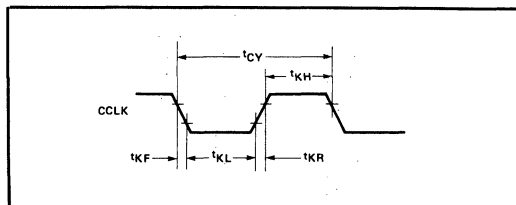
**Write Timing**



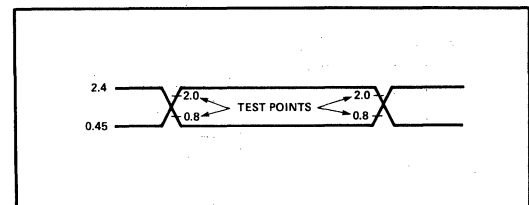
**Read Timing**



**Clock Timing**



**Input and Output Waveforms for A.C. Tests**



FOR A.C. TESTING, INPUTS ARE DRIVEN AT 2.4V FOR A LOGIC "1" AND 0.45V FOR A LOGIC "0." TIMING MEASUREMENTS FOR INPUT AND OUTPUT SIGNALS ARE MADE AT 2.0V FOR A LOGIC "1" AND 0.8V FOR A LOGIC "0."



# 8279/8279-5 PROGRAMMABLE KEYBOARD/DISPLAY INTERFACE

- MCS-85™ Compatible 8279-5
- Simultaneous Keyboard Display Operations
- Scanned Keyboard Mode
- Scanned Sensor Mode
- Strobed Input Entry Mode
- 8-Character Keyboard FIFO
- 2-Key Lockout or N-Key Rollover with Contact Debounce
- Dual 8- or 16-Numerical Display
- Single 16-Character Display
- Right or Left Entry 16-Byte Display RAM
- Mode Programmable from CPU
- Programmable Scan Timing
- Interrupt Output on Key Entry

The Intel® 8279 is a general purpose programmable keyboard and display I/O interface device designed for use with Intel® microprocessors. The keyboard portion can provide a scanned interface to a 64-contact key matrix. The keyboard portion will also interface to an array of sensors or a strobed interface keyboard, such as the hall effect and ferrite variety. Key depressions can be 2-key lockout or N-key rollover. Keyboard entries are debounced and strobed in an 8-character FIFO. If more than 8 characters are entered, overrun status is set. Key entries set the interrupt output line to the CPU.

The display portion provides a scanned display interface for LED, incandescent, and other popular display technologies. Both numeric and alphanumeric segment displays may be used as well as simple indicators. The 8279 has 16X8 display RAM which can be organized into dual 16X4. The RAM can be loaded or interrogated by the CPU. Both right entry, calculator and left entry typewriter display formats are possible. Both read and write of the display RAM can be done with auto-increment of the display RAM address.

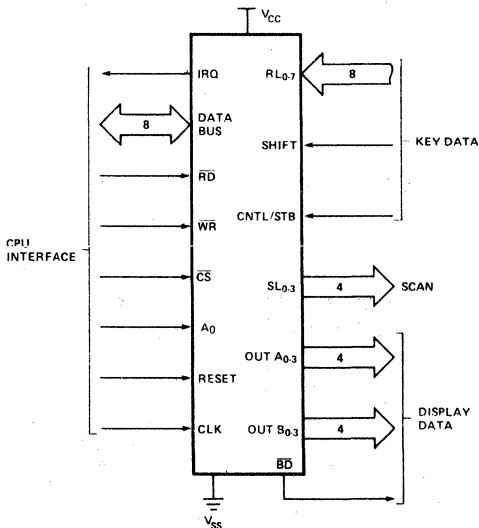


Figure 1. Logic Symbol

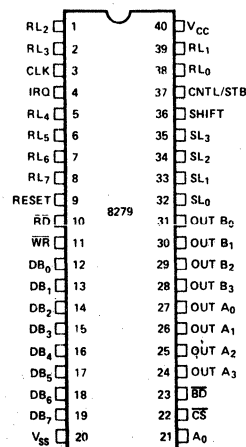


Figure 2. Pin Configuration

## HARDWARE DESCRIPTION

The 8279 is packaged in a 40 pin DIP. The following is a functional description of each pin.

**Table 1. Pin Descriptions**

Symbol	Pin No.	Name and Function
DB <sub>0</sub> -DB <sub>7</sub>	8	<b>Bi-directional data bus:</b> All data and commands between the CPU and the 8279 are transmitted on these lines.
CLK	1	<b>Clock:</b> Clock from system used to generate internal timing.
RESET	1	<b>Reset:</b> A high signal on this pin resets the 8279. After being reset the 8279 is placed in the following mode: 1) 16 8-bit character display—left entry. 2) Encoded scan keyboard—2 key lockout. Along with this the program clock prescaler is set to 31.
CS	1	<b>Chip Select:</b> A low on this pin enables the interface functions to receive or transmit.
A <sub>0</sub>	1	<b>Buffer Address:</b> A high on this line indicates the signals in or out are interpreted as a command or status. A low indicates that they are data.
$\overline{RD}$ , $\overline{WR}$	2	<b>Input/Output Read and Write:</b> These signals enable the data buffers to either send data to the external bus or receive it from the external bus.
IRQ	1	<b>Interrupt Request:</b> In a keyboard mode, the interrupt line is high when there is data in the FIFO/Sensor RAM. The interrupt line goes low with each FIFO/Sensor RAM read and returns high if there is still information in the RAM. In a sensor mode, the interrupt line goes high whenever a change in a sensor is detected.
V <sub>SS</sub> , V <sub>CC</sub>	2	<b>Ground and power supply pins.</b>
SL <sub>0</sub> -SL <sub>3</sub>	4	<b>Scan Lines:</b> Scan lines which are used to scan the key switch or sensor matrix and the display digits. These lines can be either encoded (1 of 16) or decoded (1 of 4).
RL <sub>0</sub> -RL <sub>7</sub>	8	<b>Return Line:</b> Return line inputs which are connected to the scan lines through the keys or sensor switches. They have active internal pullups to keep them high until a switch closure pulls one low. They also serve as an 8-bit input in the Strobed Input mode.

Symbol	Pin No.	Name and Function
SHIFT	1	<b>Shift:</b> The shift input status is stored along with the key position on key closure in the Scanned Keyboard modes. It has an active internal pullup to keep it high until a switch closure pulls it low.
CNTL/STB	1	<b>Control/Strobed Input Mode:</b> For keyboard modes this line is used as a control input and stored like status on a key closure. The line is also the strobe line that enters the data into the FIFO in the Strobed Input mode.  (Rising Edge). It has an active internal pullup to keep it high until a switch closure pulls it low.
OUT A <sub>0</sub> -OUT A <sub>3</sub> OUT B <sub>0</sub> -OUT B <sub>3</sub>	4 4	<b>Outputs:</b> These two ports are the outputs for the 16 x 4 display refresh registers. The data from these outputs is synchronized to the scan lines (SL <sub>0</sub> -SL <sub>3</sub> ) for multiplexed digit displays. The two 4 bit ports may be blanked independently. These two ports may also be considered as one 8-bit port.
$\overline{BD}$	1	<b>Blank Display:</b> This output is used to blank the display during digit switching or by a display blanking command.

## FUNCTIONAL DESCRIPTION

Since data input and display are an integral part of many microprocessor designs, the system designer needs an interface that can control these functions without placing a large load on the CPU. The 8279 provides this function for 8-bit microprocessors.

The 8279 has two sections: keyboard and display. The keyboard section can interface to regular typewriter style keyboards or random toggle or thumb switches. The display section drives alphanumeric displays or a bank of indicator lights. Thus the CPU is relieved from scanning the keyboard or refreshing the display.

The 8279 is designed to directly connect to the microprocessor bus. The CPU can program all operating modes for the 8279. These modes include:

### Input Modes

- Scanned Keyboard — with encoded (8 x 8 key keyboard) or decoded (4 x 8 key keyboard) scan lines. A key depression generates a 6-bit encoding of key position. Position and shift and control status are stored in the FIFO. Keys are automatically debounced with 2-key lockout or N-key rollover.
- Scanned Sensor Matrix — with encoded (8 x 8 matrix switches) or decoded (4 x 8 matrix switches) scan lines. Key status (open or closed) stored in RAM addressable by CPU.
- Strobed Input — Data on return lines during control line strobe is transferred to FIFO.

### Output Modes

- 8 or 16 character multiplexed displays that can be organized as dual 4-bit or single 8-bit ( $B_0 = D_0, A_3 = D_7$ ).
- Right entry or left entry display formats.

Other features of the 8279 include:

- Mode programming from the CPU.
- Clock Prescaler
- Interrupt output to signal CPU when there is keyboard or sensor data available.
- An 8 byte FIFO to store keyboard information.
- 16 byte internal Display RAM for display refresh. This RAM can also be read by the CPU.

## PRINCIPLES OF OPERATION

The following is a description of the major elements of the 8279 Programmable Keyboard/Display interface device. Refer to the block diagram in Figure 3.

### I/O Control and Data Buffers

The I/O control section uses the  $\overline{CS}$ ,  $A_0$ ,  $\overline{RD}$  and  $\overline{WR}$  lines to control data flow to and from the various internal registers and buffers. All data flow to and from the 8279 is enabled by  $\overline{CS}$ . The character of the information, given or desired by the CPU, is identified by  $A_0$ . A logic one means the information is a command or status. A logic zero means the information is data.  $\overline{RD}$  and  $\overline{WR}$  determine the direction of data flow through the Data Buffers. The Data Buffers are bi-directional buffers that connect the internal bus to the external bus. When the chip is not selected ( $\overline{CS} = 1$ ), the devices are in a high impedance state. The drivers input during  $\overline{WR} \bullet \overline{CS}$  and output during  $\overline{RD} \bullet \overline{CS}$ .

### Control and Timing Registers and Timing Control

These registers store the keyboard and display modes and other operating conditions programmed by the CPU. The modes are programmed by presenting the proper command on the data lines with  $A_0 = 1$  and then sending a  $\overline{WR}$ . The command is latched on the rising edge of  $\overline{WR}$ .

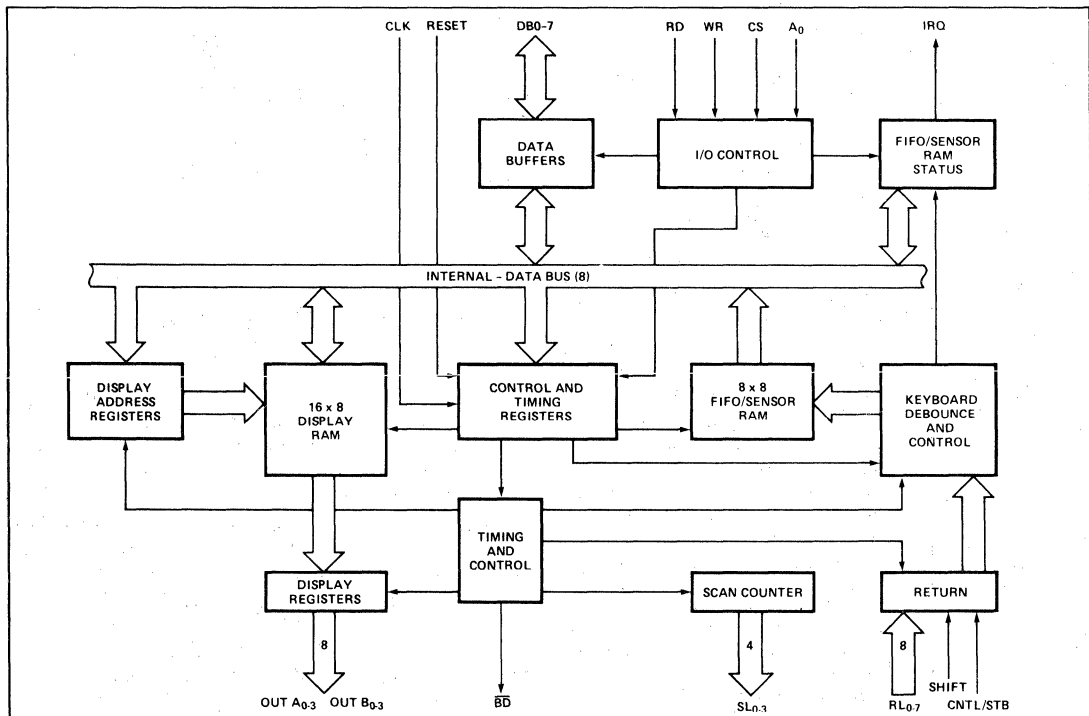


Figure 3. Internal Block Diagram

The command is then decoded and the appropriate function is set. The timing control contains the basic timing counter chain. The first counter is a  $\div N$  prescaler that can be programmed to yield an internal frequency of 100 kHz which gives a 5.1 ms keyboard scan time and a 10.3 ms debounce time. The other counters divide down the basic internal frequency to provide the proper key scan, row scan, keyboard matrix scan, and display scan times.

**Scan Counter**

The scan counter has two modes. In the encoded mode, the counter provides a binary count that must be externally decoded to provide the scan lines for the keyboard and display. In the decoded mode, the scan counter decodes the least significant 2 bits and provides a decoded 1 of 4 scan. Note that when the keyboard is in decoded scan, so is the display. This means that only the first 4 characters in the Display RAM are displayed.

In the encoded mode, the scan lines are active high outputs. In the decoded mode, the scan lines are active low outputs.

**Return Buffers and Keyboard Debounce and Control**

The 8 return lines are buffered and latched by the Return Buffers. In the keyboard mode, these lines are scanned, looking for key closures in that row. If the debounce circuit detects a closed switch, it waits about 10 msec to check if the switch remains closed. If it does, the address of the switch in the matrix plus the status of SHIFT and CONTROL are transferred to the FIFO. In the scanned Sensor Matrix modes, the contents of the return lines is directly transferred to the corresponding row of the Sensor RAM (FIFO) each key scan time. In Strobed Input mode, the contents of the return lines are transferred to the FIFO on the rising edge of the CNTL/STB line pulse.

**FIFO/Sensor RAM and Status**

This block is a dual function 8 x 8 RAM. In Keyboard or Strobed Input modes, it is a FIFO. Each new entry is written into successive RAM positions and each is then read in order of entry. FIFO status keeps track of the number of characters in the FIFO and whether it is full or empty. Too many reads or writes will be recognized as an error. The status can be read by an  $\overline{RD}$  with  $\overline{CS}$  low and  $A_0$  high. The status logic also provides an IRQ signal when the FIFO is not empty. In Scanned Sensor Matrix mode, the memory is a Sensor RAM. Each row of the Sensor RAM is loaded with the status of the corresponding row of sensor in the sensor matrix. In this mode, IRQ is high if a change in a sensor is detected.

**Display Address Registers and Display RAM**

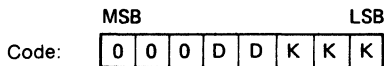
The Display Address Registers hold the address of the word currently being written or read by the CPU and the two 4-bit nibbles being displayed. The read/write addresses are programmed by CPU command. They also can be set to auto increment after each read or write. The Display RAM can be directly read by the CPU after the correct mode and address is set. The addresses for the A and B nibbles are automatically updated by the 8279 to match data entry by the CPU. The A and B nibbles can be entered independently or as one word, according to the mode that is set by the CPU. Data entry to the display can be set to either left or right entry. See Interface Considerations for details.

**SOFTWARE OPERATION**

**8279 commands**

The following commands program the 8279 operating modes. The commands are sent on the Data Bus with  $\overline{CS}$  low and  $A_0$  high and are loaded to the 8279 on the rising edge of  $\overline{WR}$ .

**Keyboard/Display Mode Set**



Where DD is the Display Mode and KKK is the Keyboard Mode.

**DD**

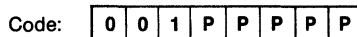
- 0 0 8 8-bit character display — Left entry
- 0 1 16 8-bit character display — Left entry\*
- 1 0 8 8-bit character display — Right entry
- 1 1 16 8-bit character display — Right entry

For description of right and left entry, see Interface Considerations. Note that when decoded scan is set in keyboard mode, the display is reduced to 4 characters independent of display mode set.

**KKK**

- 0 0 0 Encoded Scan Keyboard — 2 Key Lockout\*
- 0 0 1 Decoded Scan Keyboard — 2-Key Lockout
- 0 1 0 Encoded Scan Keyboard — N-Key Rollover
- 0 1 1 Decoded Scan Keyboard — N-Key Rollover
- 1 0 0 Encoded Scan Sensor Matrix
- 1 0 1 Decoded Scan Sensor Matrix
- 1 1 0 Strobed Input, Encoded Display Scan
- 1 1 1 Strobed Input, Decoded Display Scan

**Program Clock**



All timing and multiplexing signals for the 8279 are generated by an internal prescaler. This prescaler divides the external clock (pin 3) by a programmable integer. Bits P P P P P determine the value of this integer which ranges from 2 to 31. Choosing a divisor that yields 100 kHz will give the specified scan and debounce times. For instance, if Pin 3 of the 8279 is being clocked by a 2 MHz signal, P P P P P should be set to 10100 to divide the clock by 20 to yield the proper 100 kHz operating frequency.

**Read FIFO/Sensor RAM**



The CPU sets up the 8279 for a read of the FIFO/Sensor RAM by first writing this command. In the Scan Key-

\*Default after reset.



board Mode, the Auto-Increment flag (AI) and the RAM address bits (AAA) are irrelevant. The 8279 will automatically drive the data bus for each subsequent read ( $A_0 = 0$ ) in the same sequence in which the data first entered the FIFO. All subsequent reads will be from the FIFO until another command is issued.

In the Sensor Matrix Mode, the RAM address bits AAA select one of the 8 rows of the Sensor RAM. If the AI flag is set ( $AI = 1$ ), each successive read will be from the subsequent row of the sensor RAM.

**Read Display RAM**

Code: 

0	1	1	AI	A	A	A	A
---	---	---	----	---	---	---	---

The CPU sets up the 8279 for a read of the Display RAM by first writing this command. The address bits AAAA select one of the 16 rows of the Display RAM. If the AI flag is set ( $AI = 1$ ), this row address will be incremented after each following read or write to the Display RAM. Since the same counter is used for both reading and writing, this command sets the next read or write address and the sense of the Auto-Increment mode for both operations.

**Write Display RAM**

Code: 

1	0	0	AI	A	A	A	A
---	---	---	----	---	---	---	---

The CPU sets up the 8279 for a write to the Display RAM by first writing this command. After writing the command with  $A_0 = 1$ , all subsequent writes with  $A_0 = 0$  will be to the Display RAM. The addressing and Auto-Increment functions are identical to those for the Read Display RAM. However, this command does not affect the source of subsequent Data Reads; the CPU will read from whichever RAM (Display or FIFO/Sensor) which was last specified. If, indeed, the Display RAM was last specified, the Write Display RAM will, nevertheless, change the next Read location.

**Display Write Inhibit/Blanking**

Code: 

1	0	1	X	IW	IW	BL	BL
---	---	---	---	----	----	----	----

The IW Bits can be used to mask nibble A and nibble B in applications requiring separate 4-bit display ports. By setting the IW flag ( $IW = 1$ ) for one of the ports, the port becomes marked so that entries to the Display RAM from the CPU do not affect that port. Thus, if each nibble is input to a BCD decoder, the CPU may write a digit to the Display RAM without affecting the other digit being displayed. It is important to note that bit  $B_0$  corresponds to bit  $D_0$  on the CPU bus, and that bit  $A_3$  corresponds to bit  $D_7$ .

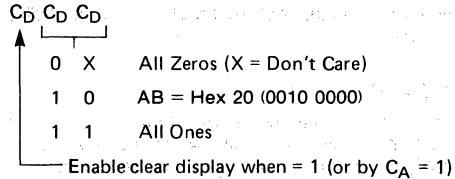
If the user wishes to blank the display, the BL flags are available for each nibble. The last Clear command issued determines the code to be used as a "blank." This code defaults to all zeros after a reset. Note that both BL flags must be set to blank a display formatted with a single 8-bit port.

**Clear**

Code: 

1	1	0	$C_D$	$C_D$	$C_D$	$C_F$	$C_A$
---	---	---	-------	-------	-------	-------	-------

The  $C_D$  bits are available in this command to clear all rows of the Display RAM to a selectable blanking code as follows:



During the time the Display RAM is being cleared ( $\sim 160 \mu s$ ), it may not be written to. The most significant bit of the FIFO status word is set during this time. When the Display RAM becomes available again, it automatically resets.

If the  $C_F$  bit is asserted ( $C_F = 1$ ), the FIFO status is cleared and the interrupt output line is reset. Also, the Sensor RAM pointer is set to row 0.

$C_A$ , the Clear All bit, has the combined effect of  $C_D$  and  $C_F$ ; it uses the  $C_D$  clearing code on the Display RAM and also clears FIFO status. Furthermore, it resynchronizes the internal timing chain.

**End Interrupt/Error Mode Set**

Code: 

1	1	1	E	X	X	X	X
---	---	---	---	---	---	---	---

 X = Don't care.

For the sensor matrix modes this command lowers the IRQ line and enables further writing into RAM. (The IRQ line would have been raised upon the detection of a change in a sensor value. This would have also inhibited further writing into the RAM until reset).

For the N-key rollover mode — if the E bit is programmed to "1" the chip will operate in the special Error mode. (For further details, see Interface Considerations Section.)

**Status Word**

The status word contains the FIFO status, error, and display unavailable signals. This word is read by the CPU when  $A_0$  is high and  $\overline{CS}$  and  $\overline{RD}$  are low. See Interface Considerations for more detail on status word.

**Data Read**

Data is read when  $A_0$ ,  $\overline{CS}$  and  $\overline{RD}$  are all low. The source of the data is specified by the Read FIFO or Read Display commands. The trailing edge of  $\overline{RD}$  will cause the address of the RAM being read to be incremented if the Auto-Increment flag is set. FIFO reads always increment (if no error occurs) independent of AI.

**Data Write**

Data that is written with  $A_0$ ,  $\overline{CS}$  and  $\overline{WR}$  low is always written to the Display RAM. The address is specified by the latest Read Display or Write Display command. Auto-Incrementing on the rising edge of  $\overline{WR}$  occurs if AI set by the latest display command.

## INTERFACE CONSIDERATIONS

### Scanned Keyboard Mode, 2-Key Lockout

There are three possible combinations of conditions that can occur during debounce scanning. When a key is depressed, the debounce logic is set. Other depressed keys are looked for during the next two scans. If none are encountered, it is a single key depression and the key position is entered into the FIFO along with the status of CNTL and SHIFT lines. If the FIFO was empty, IRQ will be set to signal the CPU that there is an entry in the FIFO. If the FIFO was full, the key will not be entered and the error flag will be set. If another closed switch is encountered, no entry to the FIFO can occur. If all other keys are released before this one, then it will be entered to the FIFO. If this key is released before any other, it will be entirely ignored. A key is entered to the FIFO only once per depression, no matter how many keys were pressed along with it or in what order they were released. If two keys are depressed within the debounce cycle, it is a simultaneous depression. Neither key will be recognized until one key remains depressed alone. The last key will be treated as a single key depression.

### Scanned Keyboard Mode, N-Key Rollover

With N-key Rollover each key depression is treated independently from all others. When a key is depressed, the debounce circuit waits 2 keyboard scans and then checks to see if the key is still down. If it is, the key is entered into the FIFO. Any number of keys can be depressed and another can be recognized and entered into the FIFO. If a simultaneous depression occurs, the keys are recognized and entered according to the order the keyboard scan found them.

### Scanned Keyboard — Special Error Modes

For N-key rollover mode the user can program a special error mode. This is done by the "End Interrupt/Error Mode Set" command. The debounce cycle and key-validity check are as in normal N-key mode. If during a single debounce cycle, two keys are found depressed, this is considered a simultaneous multiple depression, and sets an error flag. This flag will prevent any further writing into the FIFO and will set interrupt (if not yet set). The error flag could be read in this mode by reading the FIFO STATUS word. (See "FIFO STATUS" for further details.) The error flag is reset by sending the normal CLEAR command with CF = 1.

### Sensor Matrix Mode

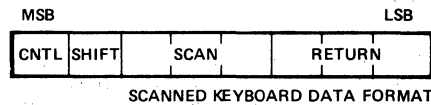
In Sensor Matrix mode, the debounce logic is inhibited. The status of the sensor switch is inputted directly to the Sensor RAM. In this way the Sensor RAM keeps an image of the state of the switches in the sensor matrix. Although debouncing is not provided, this mode has the advantage that the CPU knows how long the sensor was closed and when it was released. A keyboard mode can only indicate a validated closure. To make the software easier, the designer should functionally group the sensors by row since this is the format in which the CPU will read them. The IRQ line goes high if any sensor value change is detected at the end of a sensor matrix scan. The IRQ line is cleared by the first data read operation if the Auto-

Increment flag is set to zero, or by the End Interrupt command if the Auto-Increment flag is set to one.

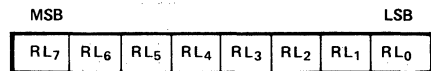
**Note:** Multiple changes in the matrix Addressed by (SL<sub>0-3</sub> = 0) may cause multiple interrupts. (SL<sub>0</sub> = 0 in the Decoded Mode). Reset may cause the 8279 to see multiple changes.

### Data Format

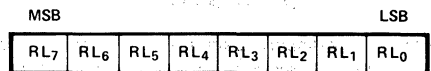
In the Scanned Keyboard mode, the character entered into the FIFO corresponds to the position of the switch in the keyboard plus the status of the CNTL and SHIFT lines (non-inverted). CNTL is the MSB of the character and SHIFT is the next most significant bit. The next three bits are from the scan counter and indicate the row the key was found in. The last three bits are from the column counter and indicate to which return line the key was connected.



In Sensor Matrix mode, the data on the return lines is entered directly in the row of the Sensor RAM that corresponds to the row in the matrix being scanned. Therefore, each switch position maps directly to a Sensor RAM position. The SHIFT and CNTL inputs are ignored in this mode. Note that switches are not necessarily the only thing that can be connected to the return lines in this mode. Any logic that can be triggered by the scan lines can enter data to the return line inputs. Eight multiplexed input ports could be tied to the return lines and scanned by the 8279.



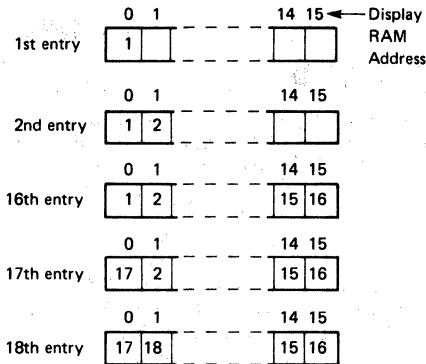
In Strobed Input mode, the data is also entered to the FIFO from the return lines. The data is entered by the rising edge of a CNTL/STB line pulse. Data can come from another encoded keyboard or simple switch matrix. The return lines can also be used as a general purpose strobed input.



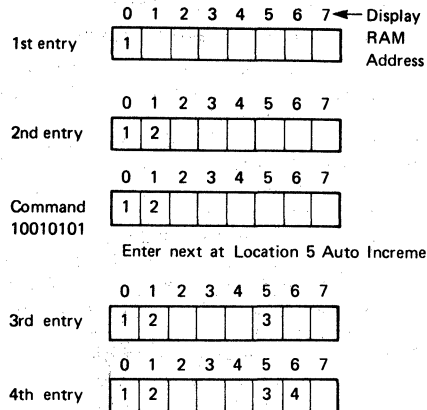
### Display

#### Left Entry

Left Entry mode is the simplest display format in that each display position directly corresponds to a byte (or nibble) in the Display RAM. Address 0 in the RAM is the left-most display character and address 15 (or address 7 in 8 character display) is the right most display character. Entering characters from position zero causes the display to fill from the left. The 17th (9th) character is entered back in the left most position and filling again proceeds from there.



LEFT ENTRY MODE (AUTO INCREMENT)

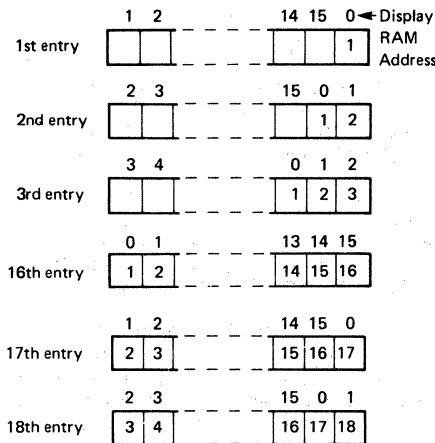


LEFT ENTRY MODE (AUTO INCREMENT)

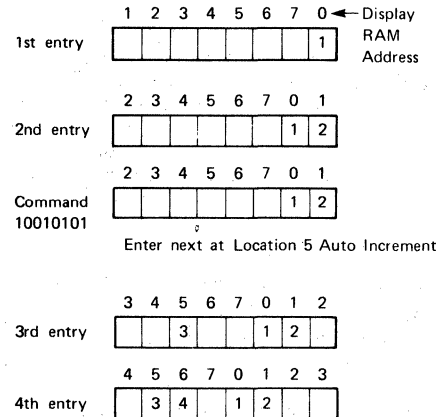
**Right Entry**

Right entry is the method used by most electronic calculators. The first entry is placed in the right most display character. The next entry is also placed in the right most character after the display is shifted left one character. The left most character is shifted off the end and is lost.

In the Right Entry mode, Auto Incrementing and non Incrementing have the same effect as in the Left Entry except if the address sequence is interrupted:



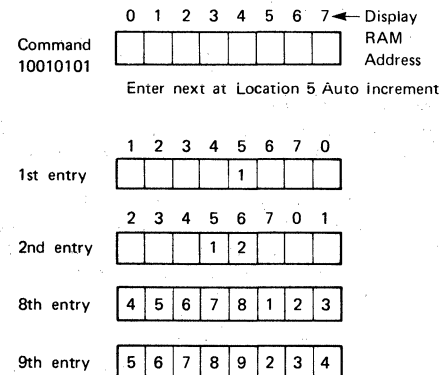
RIGHT ENTRY MODE (AUTO INCREMENT)



RIGHT ENTRY MODE (AUTO INCREMENT)

Note that now the display position and register address do not correspond. Consequently, entering a character to an arbitrary position in the Auto Increment mode may have unexpected results. Entry starting at Display RAM address 0 with sequential entry is recommended.

Starting at an arbitrary location operates as shown below:



RIGHT ENTRY MODE (AUTO INCREMENT)

**Auto Increment**

In the Left Entry mode, Auto Incrementing causes the address where the CPU will next write to be incremented by one and the character appears in the next location. With non-Auto Incrementing the entry is both to the same RAM address and display position. Entry to an arbitrary address in the Auto Increment mode has no undesirable side effects and the result is predictable:

Entry appears to be from the initial entry point.

**8/16 Character Display Formats**

If the display mode is set to an 8 character display, the on duty-cycle is double what it would be for a 16 character display (e.g., 5.1 ms scan time for 8 characters vs. 10.3 ms for 16 characters with 100 kHz internal frequency).

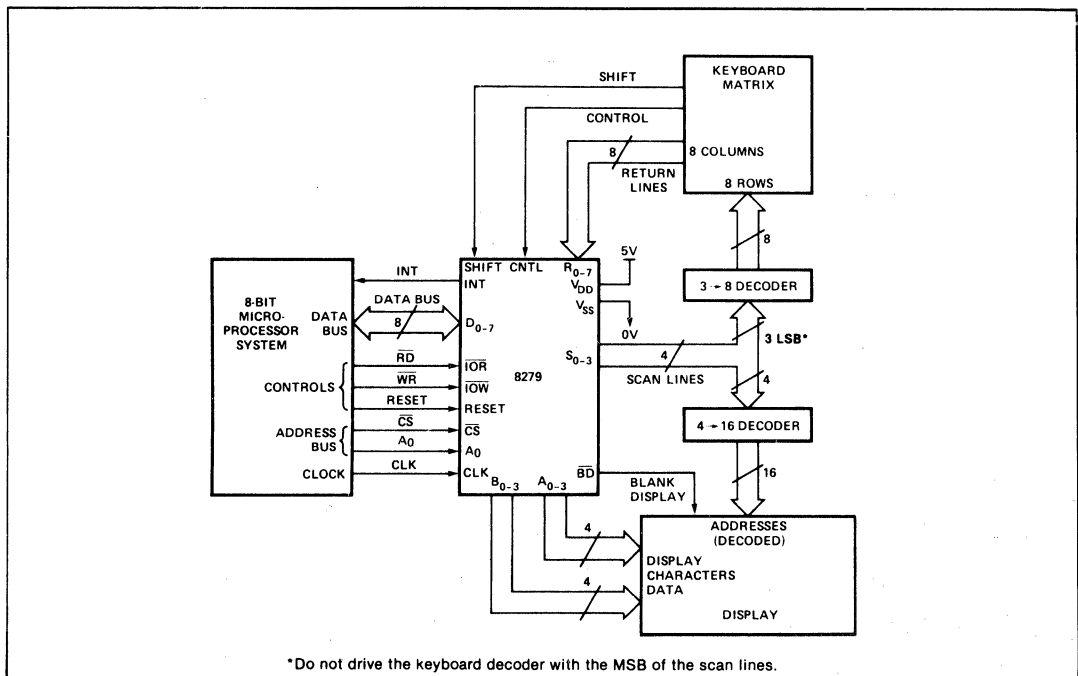
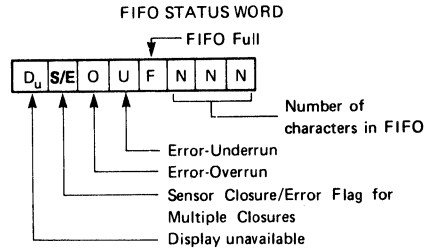
**G. FIFO Status**

FIFO status is used in the Keyboard and Strobed Input modes to indicate the number of characters in the FIFO and to indicate whether an error has occurred. There are two types of errors possible: overrun and underrun. Overrun occurs when the entry of another character into a full FIFO is attempted. Underrun occurs when the CPU tries to read an empty FIFO.

The FIFO status word also has a bit to indicate that the Display RAM was unavailable because a Clear Display or Clear All command had not completed its clearing operation.

In a Sensor Matrix mode, a bit is set in the FIFO status word to indicate that at least one sensor closure indication is contained in the Sensor RAM.

In Special Error Mode the S/E bit is showing the error flag and serves as an indication to whether a simultaneous multiple closure error has occurred.



**Figure 4. System Block Diagram**

**ABSOLUTE MAXIMUM RATINGS\***

Ambient Temperature	0°C to 70°C
Storage Temperature	-65°C to 125°C
Voltage on any Pin with Respect to Ground	-0.5V to +7V
Power Dissipation	1 Watt

*\*NOTICE: Stresses above those listed under "Absolute Maximum Ratings" may cause permanent damage to the device. This is a stress rating only and functional operation of the device at these or any other conditions above those indicated in the operational sections of this specification is not implied. Exposure to absolute maximum rating conditions for extended periods may affect device reliability.*

**D.C. CHARACTERISTICS** [ $T_A = 0^\circ\text{C}$  to  $70^\circ\text{C}$ ,  $V_{SS} = V_{CC} = +5V \pm 5\%$ ,  $V_{CC} = +5V \pm 10\%$  (8279-5)]

Symbol	Parameter	Min.	Max.	Unit	Test Conditions
$V_{IL1}$	Input Low Voltage for Return Lines	-0.5	1.4	V	
$V_{IL2}$	Input Low Voltage for All Others	-0.5	0.8	V	
$V_{IH1}$	Input High Voltage for Return Lines	2.2		V	
$V_{IH2}$	Input High Voltage for All Others	2.0		V	
$V_{OL}$	Output Low Voltage		0.45	V	Note 1
$V_{OH1}$	Output High Voltage on Interrupt Line	3.5		V	Note 2
$V_{OH2}$	Other Outputs	2.4			
$I_{IL1}$	Input Current on Shift, Control and Return Lines		+10 -100	$\mu\text{A}$ $\mu\text{A}$	$V_{IN} = V_{CC}$ $V_{IN} = 0V$
$I_{IL2}$	Input Leakage Current on All Others		$\pm 10$	$\mu\text{A}$	$V_{IN} = V_{CC}$ to $0V$
$I_{OFL}$	Output Float Leakage		$\pm 10$	$\mu\text{A}$	$V_{OUT} = V_{CC}$ to $0V$
$I_{CC}$	Power Supply Current		120	$\text{mA}$	

**CAPACITANCE**

Symbol	Parameter	Typ.	Max.	Unit	Test Conditions
$C_{IN}$	Input Capacitance	5	10	$\text{pF}$	$V_{IN} = V_{CC}$
$C_{OUT}$	Output Capacitance	10	20	$\text{pF}$	$V_{OUT} = V_{CC}$

**A.C. CHARACTERISTICS** [ $T_A = 0^\circ\text{C}$  to  $70^\circ\text{C}$ ,  $V_{SS} = 0V$ , (Note 3)]

**Bus Parameters**
**READ CYCLE**

Symbol	Parameter	8279		8279-5		Unit
		Min.	Max.	Min.	Max.	
$t_{AR}$	Address Stable Before $\overline{\text{READ}}$	50		0		ns
$t_{RA}$	Address Hold Time for $\overline{\text{READ}}$	5		0		ns
$t_{RR}$	$\overline{\text{READ}}$ Pulse Width	420		250		ns
$t_{RD}^{[4]}$	Data Delay from $\overline{\text{READ}}$		300		150	ns
$t_{AD}^{[4]}$	Address to Data Valid		450		250	ns
$t_{DF}$	$\overline{\text{READ}}$ to Data Floating	10	100	10	100	ns
$t_{RCY}$	Read Cycle Time	1		1		$\mu\text{s}$

**A.C. CHARACTERISTICS (Continued)**
**WRITE CYCLE**

Symbol	Parameter	8279		8279-5		Unit
		Min.	Max.	Min.	Max.	
$t_{AW}$	Address Stable Before $\overline{WRITE}$	50		0		ns
$t_{WA}$	Address Hold Time for $\overline{WRITE}$	20		0		ns
$t_{WW}$	$\overline{WRITE}$ Pulse Width	400		250		ns
$t_{DW}$	Data Set Up Time for $\overline{WRITE}$	300		150		ns
$t_{WD}$	Data Hold Time for $\overline{WRITE}$	40		0		ns
$t_{WCY}$	Write Cycle Time	1		1		$\mu$ S

**OTHER TIMINGS**

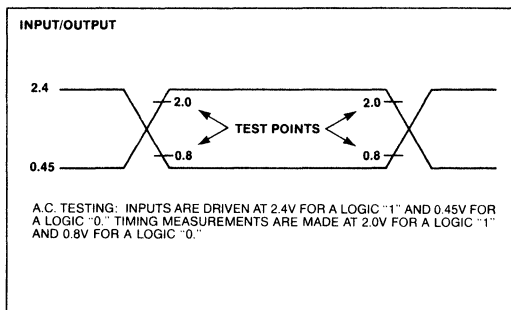
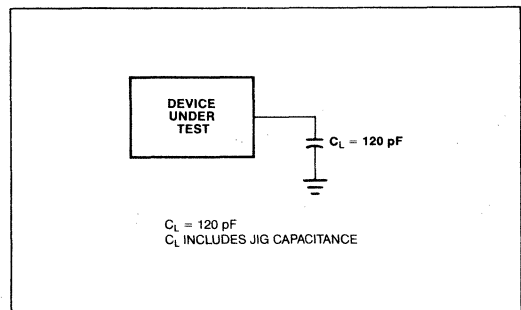
Symbol	Parameter	8279		8279-5		Unit
		Min.	Max.	Min.	Max.	
$t_{\phi W}$	Clock Pulse Width	230		120		nsec
$t_{CY}$	Clock Period	500		320		nsec

Keyboard Scan Time ..... 5.1 msec  
 Keyboard Debounce Time ..... 10.3 msec  
 Key Scan Time ..... 80  $\mu$ sec  
 Display Scan Time ..... 10.3 msec

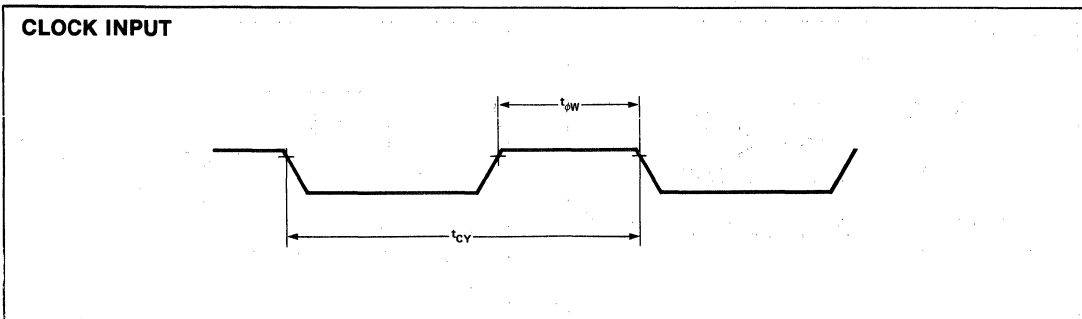
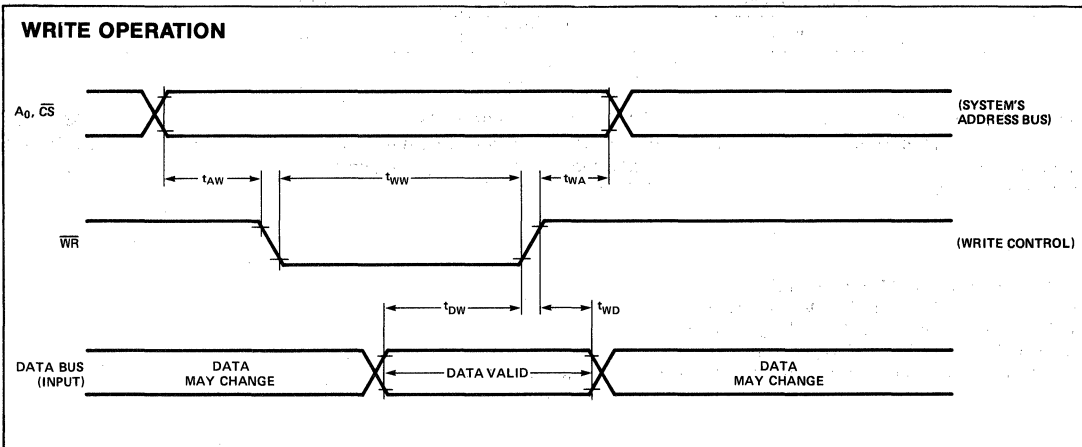
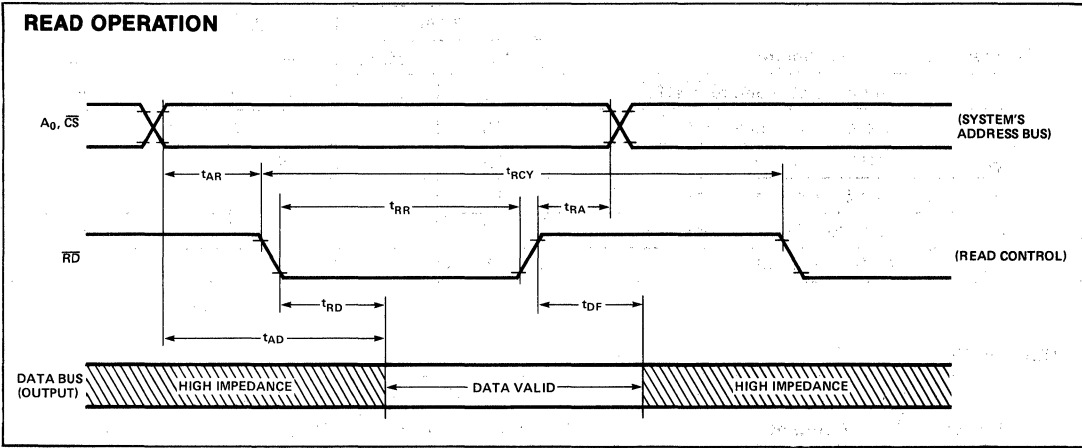
Digit-on Time ..... 480  $\mu$ sec  
 Blanking Time ..... 160  $\mu$ sec  
 Internal Clock Cycle<sup>[5]</sup> ..... 10  $\mu$ sec

**NOTES:**

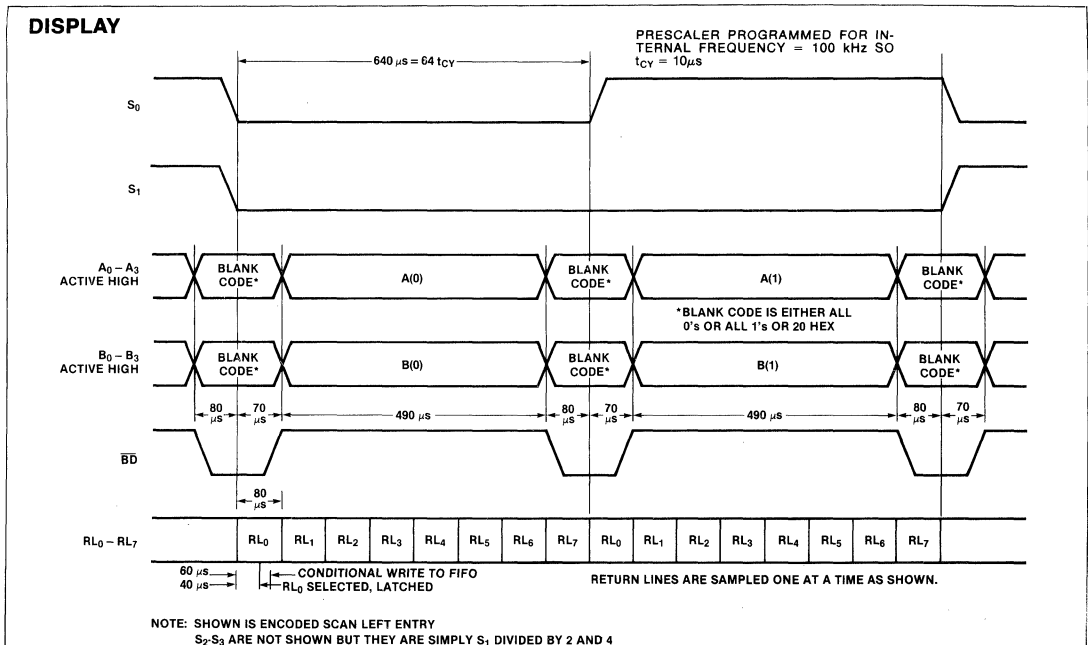
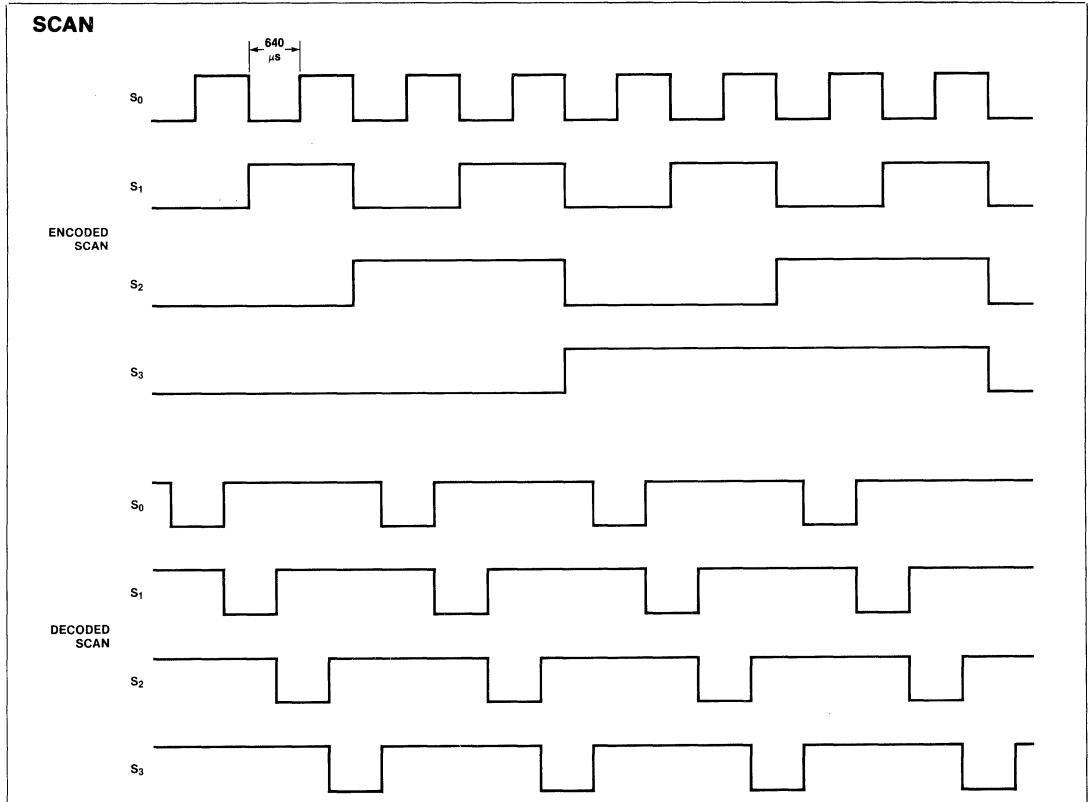
1. 8279,  $I_{OL} = 1.6\text{mA}$ ; 8279-5,  $I_{OL} = 2.2\text{mA}$ .
2. 8279,  $I_{OH} = -100\mu\text{A}$ ; 8279-5,  $I_{OH} = -400\mu\text{A}$ .
3. 8279,  $V_{CC} = +5V \pm 5\%$ ; 8279-5,  $V_{CC} = +5V \pm 10\%$ .
4. 8279,  $C_L = 100\text{pF}$ ; 8279-5,  $C_L = 150\text{pF}$ .
5. The Prescaler should be programmed to provide a 10  $\mu$ s internal clock cycle.

**A.C. TESTING INPUT, OUTPUT WAVEFORM**

**A.C. TESTING LOAD CIRCUIT**


WAVEFORMS



WAVEFORMS (Continued)



NOTE: SHOWN IS ENCODED SCAN LEFT ENTRY  
 $S_2 - S_3$  ARE NOT SHOWN BUT THEY ARE SIMPLY  $S_1$ , DIVIDED BY 2 AND 4





# ***Slave Processors***

10/10/00

10/10/00



10/10/00

10/10/00

---

# Introduction to the UPI-41A™

## Contents

<b>INTRODUCTION</b>	2-2
<b>UPI-41 VS. UPI-41A</b>	2-2
<b>UPI/MASTER PROTOCOL</b>	2-3
<b>EXAMPLE APPLICATIONS</b>	2-9
8-Digit Multiplexed LED Display	2-10
Sensor Matrix Controller	2-15
Combination I/O Device	2-19
<b>DEBUG TECHNIQUES</b>	2-27
<b>CONCLUSION</b>	2-30
<b>APPENDIX A</b>	2-31

## INTRODUCTION TO THE UPI-41A™

### Introduction

Since the introduction in 1974 of the second generation of microprocessors, such as the 8080, a wide range of peripheral interface devices have appeared. At first, these devices solved application problems of a general nature; i.e., parallel interface (8255), serial interface (8251), timing (8253), interrupt control (8259). However, as the speed and density of LSI technology increased, more and more intelligence was incorporated into the peripheral devices. This allowed more specific application problems to be solved, such as floppy disk control (8271), CRT control (8275), and data link control (8273). The advantage to the system designer of this increased peripheral device intelligence is that many of the peripheral control tasks are now handled externally to the main processor in the peripheral hardware rather than internally in the main processor software. This reduced main processor overhead results in increased system throughput and reduced software complexity.

In spite of the number of peripheral devices available, the pervasiveness of the microprocessor has been such that there is still a large number of peripheral control applications not yet satisfied by dedicated LSI. Complicating this problem is the fact that new applications are emerging faster than the manufacturers can react in developing new, dedicated peripheral controllers. To address this problem, a new microcomputer-based Universal Peripheral Interface (UPI-41A) device was developed.

In essence, the UPI-41A acts as a slave processor to the main system CPU. The UPI contains its own processor, memory, and I/O, and is completely user programmable; that is, the entire peripheral control algorithm can be programmed locally in the UPI, instead of taxing the master processor's main memory. This distributed processing concept allows the UPI to handle the real-time tasks such as encoding keyboards, controlling printers, or multiplexing displays, while the main processor is handling non-real-time dependent tasks such as buffer management or arithmetic. The UPI relies on the master only for initialization, elementary commands, and data transfers. This technique results in an overall increase in system efficiency since both processors—the master CPU and the slave UPI—are working in parallel.

This application note presents three UPI-41A applications which are roughly divided into two groups: applications whose complexity and UPI code space

requirements allow them to either stand alone or be incorporated as just one task in a “multi-tasking” UPI, and applications which are complete UPI applications in themselves. Applications in the first group are a simple LED display and sensor matrix controllers. A combination serial/parallel/ I/O device is an application in the second group. Each application illustrates different UPI configurations and features. However, before the application details are presented, a section on the UPI/master protocol requirements is included. These protocol requirements are key to UPI software development. For convenience, the UPI block diagram is reproduced in Figure 1 and the instruction set summary in Table 1.

### UPI-41 vs. UPI-41A

The UPI-41A is an enhanced version of the UPI-41. It incorporates several architectural features not found on the “non-A” device:

- Separate Data In and Data Out data bus buffer registers
- User-definable STATUS register bits
- Programmable master interrupts for the OBF and IBF flags
- Programmable DMA interface to external DMA controller.

The separate Data In (DBBIN) and Data Out (DBBOUT) registers greatly simplify the master/UPI protocol compared to the UPI-41. The master need only check IBF before writing to DBBIN and OBF before reading DBBOUT. No data bus buffer lock-out is required.

The most significant nibble of the STATUS register, undefined in the UPI-41, is user-definable in UPI-41A. It may be loaded directly from the most significant nibble of the Accumulator (MOV STS,A). These extra four STATUS bits are useful for transferring additional status information to the master. This application note uses this feature extensively.

A new instruction, EN FLAGS, allows OBF and IBF to be reflected on PORT 2 BIT 4 and PORT 2 BIT 5 respectively. This feature enables interrupt-driven data transfers when these pins are interrupt sources to the master.

By executing an EN DMA instruction PORT 2 BIT 6 becomes a DRQ (DMA Request) output and PORT 2 BIT 7 becomes DACK (DMA Acknowledge). Setting DRQ requests a DMA cycle to an external DMA controller. When the cycle is granted, the DMA controller returns DACK plus either RD (Read) or WR (Write). DACK automatically forces

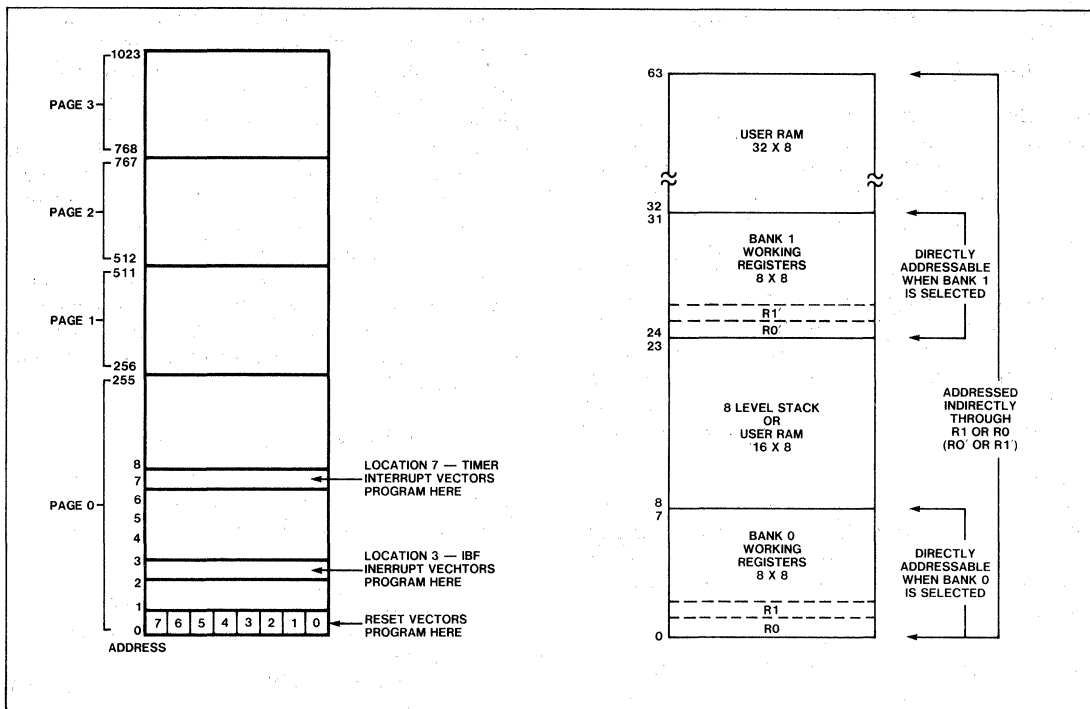


Figure 1A. Program Memory Map

Figure 1B. Data Memory Map

$\overline{CS}$  and  $A_0$  low internally and clears DRQ. This selects the appropriate data buffer register (DBBOUT for DACK and  $\overline{RD}$ , DBBIN for DACK and  $\overline{WR}$ ) for the DMA transfer.

Like the “non-A”, the UPI-41A is available in both ROM (8041A) and EPROM (8741A) Program Memory versions. This application note deals exclusively with the UPI-41A since the applications use the “A” enhanced features.

**UPI/MASTER PROTOCOL**

As in most closely coupled multiprocessor systems, the various processors communicate via a shared resource. This shared resource is typically specific locations in RAM or in registers through which status and data are passed. In the case of a master processor and a UPI-41A, the shared resource is 3 separate, master-addressable, registers internal to the UPI. These registers are the status register (STATUS), the Data Bus Buffer Input register (DBBIN), and the Data Bus Output register (DBBOUT). [Data Bus Buffer direction is relative to the UPI]. To illustrate this register interface, consider the 8085A/UPI system in Figure 2.

Looking into the UPI from the 8085A, the 8085A sees only the three registers mentioned above. If the 8085A wishes to issue a command to the UPI, it does so by writing the command to the DBBIN register according to the decoding of Table 2. Data for the UPI is also passed via the DBBIN register. (The UPI differentiates commands and data by examining the  $A_0$  pin. Just how this is done is covered shortly.) Data from the UPI for the 8085A is passed in the DBBOUT register. The 8085A may interrogate the UPI's status by reading the UPI's STATUS register. Four bits of the STATUS register act as flags and are used to handshake data and commands into and out of the UPI. The STATUS register format is shown in Figure 3.

BIT 0 is OBF (Output Buffer Full). This flag indicates to the master when the UPI has placed data in the DBBOUT register. OBF is set when the UPI writes to DBBOUT and is reset when the master reads DBBOUT. The master finds meaningful data in the DBBOUT register only when OBF is set.

The Input Buffer Full (IBF) flag is BIT 1. The UPI uses this flag as an indicator that the master has written to the DBBIN register. The master uses IBF

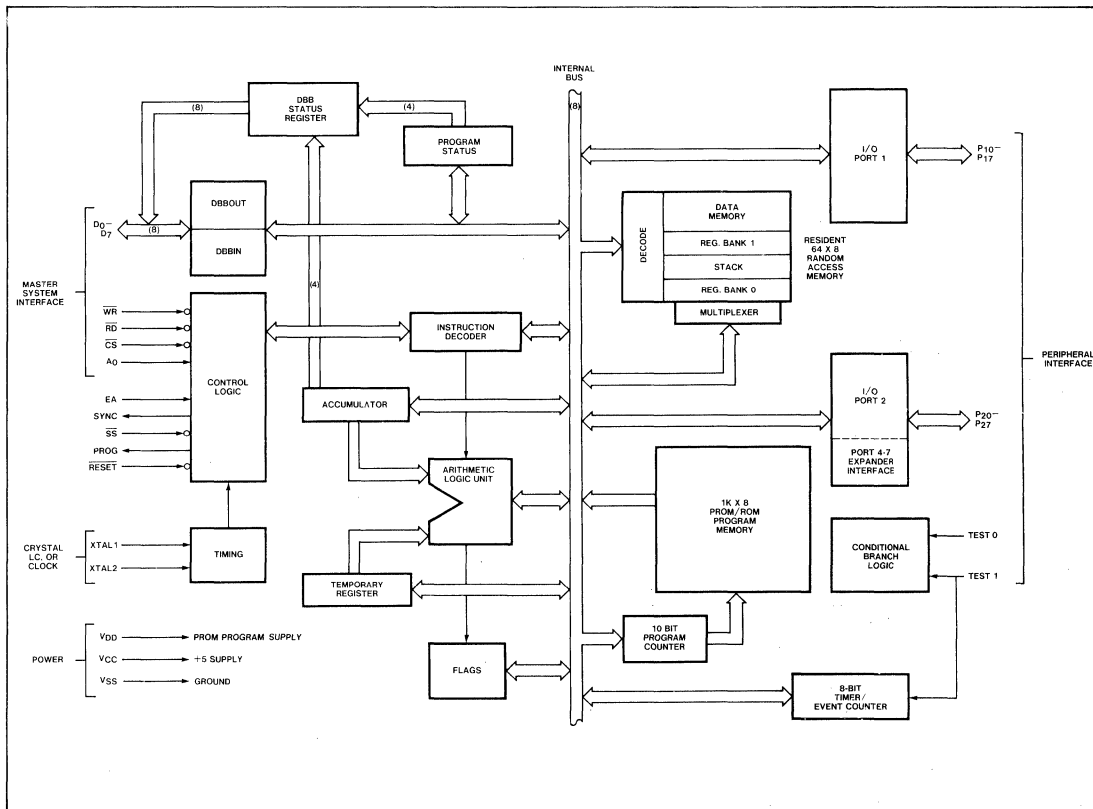


Figure 1C. UPI-41A Block Diagram

to indicate when the UPI has accepted a particular command or data byte. The master should examine IBF before outputting anything to the UPI. IBF is set when the master writes to DBBIN and is reset when the UPI reads DBBIN. The master must wait until IBF=0 before writing new data or commands to DBBIN. Conversely, the UPI must ensure IBF=1 before reading DBBIN.

The third STATUS register bit is F<sub>0</sub> (FLAG 0). This is a general purpose flag that the UPI can set, reset, and test. It is typically used to indicate a UPI error or busy condition to the master.

FLAG 1 (F<sub>1</sub>) is the final dedicated STATUS bit. Like F<sub>0</sub> the UPI can set, reset, and test this flag. However, in addition, F<sub>1</sub> reflects the state of the A<sub>0</sub> pin whenever the master writes to the DBBIN register. The UPI uses this flag to delineate between master command and data writes to DBBIN.

The remaining four STATUS register bits are user definable. Typical uses of these bits are as status in-

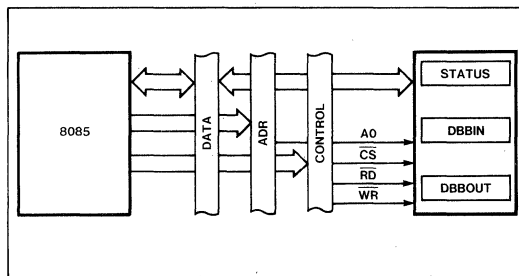


Figure 2. Register Interface

dicators for individual tasks in a multitasking UPI or as UPI generated interrupt status. These bits find a wide variety of uses in the upcoming applications.

Looking into the 8085A from the UPI, the UPI sees the two DBB registers plus the IBF, OBF, and F<sub>1</sub> flags. The UPI can write from its accumulator to DBBOUT or read DBBIN into the accumulator. The UPI cannot read OBF, IBF, or F<sub>1</sub> directly, but these flags may be tested using conditional jump

# APPLICATIONS

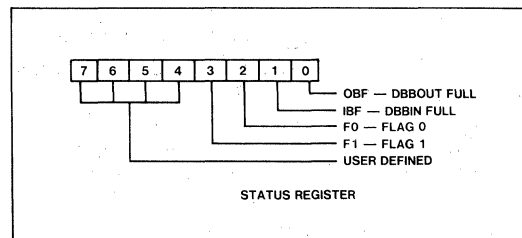
**Table 1. Instruction Set Summary**

Mnemonic	Description	Bytes	Cycles
<b>Accumulator</b>			
ADD A,R <sub>r</sub>	Add register to A	1	1
ADD A,@R <sub>r</sub>	Add data memory to A	1	1
ADD A,#data	Add immediate to A	2	2
ADDC A,R <sub>r</sub>	Add register to A with carry	1	1
ADDC A @R <sub>r</sub>	Add data memory to A with carry	1	1
ADDC A,#data	Add immed. to A with carry	2	2
ANL a,R <sub>r</sub>	AND register to A	1	1
ANL A,@R <sub>r</sub>	AND data memory to A	1	1
ANL A,#data	AND immediate to A	2	2
ORL A,R <sub>r</sub>	OR register to A	1	1
ORL A@R <sub>r</sub>	OR data memory to A	1	1
ORL A,#data	OR immediate to A	2	2
XRL A,R <sub>r</sub>	Exclusive OR register to A	1	1
XRL A,@R <sub>r</sub>	Exclusive OR data memory to A	1	1
XRL A,#data	Exclusive OR immediate to A	2	2
INC A	Increment A	1	1
DEC A	Decrement A	1	1
CLR A	Clear A	1	1
CPL A	Complement A	1	1
DA A	Decimal Adjust A	1	1
SWAP A	Swap digits of A	1	1
RL A	Rotate A left	1	1
RLC A	Rotate A left through carry	1	1
RR A	Rotate A right	1	1
RRC A	Rotate A right through carry	1	1
<b>Input/Output</b>			
IN A,P <sub>p</sub>	Input port to A	1	2
OUTL P <sub>p</sub> ,A	Output A to port	1	2
ANL P <sub>p</sub> ,#data	AND immediate to port	2	2
ORL P <sub>p</sub> ,#data	OR immediate to port	2	2
IN A,DBB	Input DBB to A, clear IBF	1	1
OUT DBB,A	Output A to DBB, set OBF	1	1
MOV STS,A	A <sub>4-7</sub> to Bits 4-7 of Status	1	1
MOVD A,P <sub>p</sub>	Input Expander port to A	1	2
MOVD P <sub>p</sub> ,A	Output A to Expander port	1	2
ANLD P <sub>p</sub> ,A	AND A to Expander port	1	2
ORLD P <sub>p</sub> ,A	OR A to Expander port	1	2
<b>Data Moves</b>			
MOV A,R <sub>r</sub>	Move register to A	1	1
MOV A,@R <sub>r</sub>	Move data memory to A	1	1
MOV A,#data	Move immediate to A	2	2
MOV R <sub>r</sub> ,A	Move A to register	1	1
MOV @R <sub>r</sub> ,A	Move A to data memory	1	1
MOV R <sub>r</sub> ,#data	Move immediate to register	2	2
MOV @R <sub>r</sub> ,#data	Move immediate to data memory	2	2
MOV A,PSW	Move PSW to A	1	1
MOV PSW,A	Move A to PSW	1	1
XCH A,R <sub>r</sub>	Exchange A and register	1	1
XCH A,@R <sub>r</sub>	Exchange A and data memory	1	1
XCHD A@R <sub>r</sub>	Exchange digit of A and register	1	1
MOVP A,@A	Move to A from current page	1	2
MOVP3, A,@A	Move to A from page 3	1	2

Mnemonic	Description	Bytes	Cycles
<b>Timer/Counter</b>			
MOV A,T	Read Timer/Counter	1	1
MOV T,A	Load Timer/Counter	1	1
STRT T	Start Timer	1	1
STRT CNT	Start Counter	1	1
STOP TCNT	Stop Timer/Counter	1	1
EN TCNTI	Enable Timer/Counter Interrupt	1	1
DIS TCNTI	Disable Timer/Counter Interrupt	1	1
<b>Control</b>			
EN DMA	Enable DMA Handshake Lines	1	1
EN I	Enable IBF Interrupt	1	1
DIS I	Disable IBF Interrupt	1	1
EN FLAGS	Enable Master Interrupts	1	1
SEL RB0	Select register bank 0	1	1
SEL RB1	Select register bank 1	1	1
NOP	No Operation	1	1
<b>Registers</b>			
INC R <sub>r</sub>	Increment register	1	1
INC @R <sub>r</sub>	Increment data memory	1	1
DEC R <sub>r</sub>	Decrement register	1	1
<b>Subroutine</b>			
CALL addr	Jump to subroutine	2	2
RET	Return	1	2
RETR	Return and restore status	1	2
<b>Flags</b>			
CLR C	Clear Carry	1	1
CPL C	Complement Carry	1	1
CLR F0	Clear Flag 0	1	1
CPL F0	Complement Flag 0	1	1
CLR F1	Clear F1 Flag	1	1
CPL F1	Complement F1 Flag	1	1
<b>Branch</b>			
JMP ADDR	Jump unconditional	2	2
JMPP @A	Jump indirect	1	2
DJNZ R,addr	Decrement register and skip	2	2
JC addr	Jump on Carry=1	2	2
JNC addr	Jump on Carry=0	2	2
JZ addr	Jump on A Zero	2	2
JNZ addr	Jump on A not Zero	2	2
JT0 addr	Jump on T0=1	2	2
JNT0 addr	Jump on T0=0	2	2
JT1 addr	Jump on T1=1	2	2
JNT1 addr	Jump on T1=0	2	2
JF0 addr	Jump on F0 Flag=1	2	2
JF1 addr	Jump on F1 Flag=1	2	2
JTF addr	Jump on Timer Flag=1, Clear Flag	2	2
JNIBF addr	Jump on IBF Flag=0	2	2
JOBF addr	Jump on OBF Flag=1	2	2
JBb addr	Jump on Accumulator Bit	2	2

**Table 2. Register Decoding**

CS	AO	RD	WR	REGISTER
0	0	0	1	READ DBBOUT
0	1	0	1	READ STATUS
0	0	1	0	WRITE DBBIN (DATA)
0	1	1	0	WRITE DBBIN (COMMAND)
1	X	X	X	NO ACTION



**Figure 3. Status Register Format**

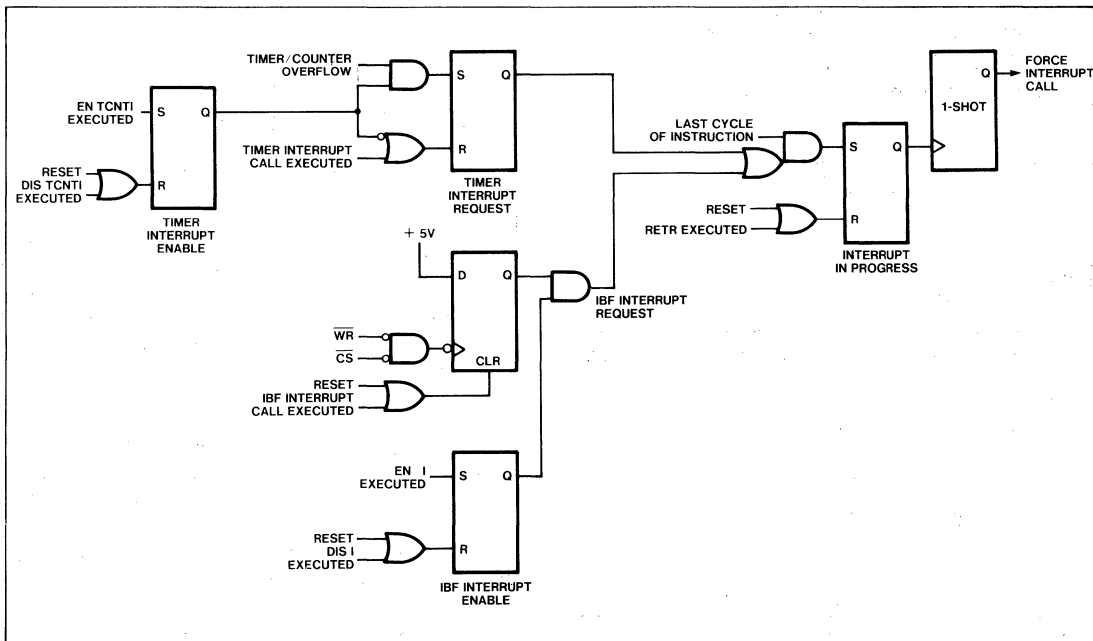
instructions. The UPI should make sure that OBF is reset before writing new data into DBBOUT to ensure that the master has read previous DBBOUT data. IBF should also be tested before reading DBBIN since DBBIN data is valid only when IBF is set. As was mentioned earlier, the UPI uses F<sub>1</sub> to differentiate between command and data contents in DBBIN when IBF is set. The UPI may also write the upper 4-bits of its accumulator to the upper 4-bits of the STATUS register. These bits are thus user definable.

The UPI can test the flags at any time during its internal program execution. It essentially "polls" the STATUS register for changes. If faster response is needed to master commands and data, the UPI's internal interrupt structure can be used. If IBF interrupts are enabled, a master write to DBBIN (either command or data) sets IBF which generates an internal CALL to location 03H in program memory. At this point, working register contents can be saved using bank switching, the accumulator saved in a spare working register, and the DBBIN register read and serviced. The interrupt logic for the IBF interrupt is shown in Figure 4. A few observations concerning this logic are appropriate. Note that if the master writes to DBBIN while the UPI is still servicing the last IBF interrupt (a RETR instruction has not been executed), the IBF Interrupt Pending line

is made high which causes a new CALL to 03H as soon as the first RETR is executed. No EN I (Enable Interrupt) instruction is needed to rearm the interrupt logic as is needed in a 8080 or 8085A system; the RETR performs this function. Also note that executing a DIS I to disable further IBF interrupts does not clear a pending interrupt. Only a CALL to location 03H or RESET clears a pending IBF interrupt.

Keeping in mind that the actual master/UPI protocol is dependent on the application, probably the best way to illustrate correct protocol is by example. Let's consider using the UPI as a simple parallel I/O device. (This is a trivial application but it embodies all of the important protocol considerations.) Since the UPI may be either interrupt or non-interrupt driven internally, both cases are considered.

Let's take the easiest configuration first; using the UPI PORT 1 as an 8-bit output port. From the UPI's point-of-view, this is an input-only application since all that is required is that the UPI input data from the master. Once the master writes data to the UPI, the UPI reads the DBBIN register and transfers the data to PORT 1. No testing for commands versus data is needed since the UPI "knows" it only performs one task—no commands are needed.



**Figure 4. UPI-41A Interrupt Structure**



Non-interrupt driven UPI software is shown in Figure 5A while Figure 5B shows interrupt based software. For Figure 5A, the UPI simply waits until it sees IBF go high indicating the master has written a data byte to DBBIN. The UPI then reads DBBIN, transfers it to PORT 1, and returns to waiting for the next data. For the interrupt-driven UPI, Figure 5B, once the EN I instruction is executed, the UPI simply waits for the IBF interrupt before handling the data. The UPI could handle other tasks during this waiting time. When the master writes the data to DBBIN, an IBF interrupt is generated which performs a CALL to location 03H. At this point the UPI reads DBBIN (no testing of IBF is needed since an IBF interrupt implies that IBF is set), transfers the data to PORT 1, and executes an RETR which returns program flow to the main program.

Software for the master 8085A is included in Figure 5C. The only requirement for the master to output data to the UPI is that it check the UPI to be sure the previous data had been taken before writing new data. To accomplish this the master simply reads the STATUS register looking for IBF=0 before writing the next data.

```

UPI INPUT ONLY EXAMPLE—PORT 1 USED AS OUTPUT PORT
UPI POLLS IBF FOR DATA

RESET:  JNIBF  RESET      ; WAIT ON IBF FOR INPUT
        IN    A,DBB      ; INPUT THERE, SO READ IT
        OUTL  P1,A       ; TRANSFER DATA TO PORT 1
        JMP   RESET      ; GO WAIT FOR NEXT DATA
    
```

Figure 5A. Single Output Port Example—Polling

```

UPI INPUT ONLY EXAMPLE—PORT 1 USED AS OUTPUT PORT
DATA INPUT IS INTERRUPT-DRIVEN ON IBF

RESET:  EN    I          ; ENABLE IBF INTERRUPTS
        JMP   RESET+1    ; LOOP WAITING FOR INPUT
IBFINT: IN    A,DBB      ; READ DATA FROM DBBIN
        OUTL  P1,A       ; TRANSFER DATA TO PORT 1
        RETR             ; RETURN WITH RESTORE
    
```

Figure 5B. Single Output Port Example—Interrupt

```

8085 SOFTWARE FOR UPI INPUT-ONLY EXAMPLE
DATA FOR OUTPUT IS PASSED IN REG. C

UPIOUT: IN    STATUS     ; READ UPI STATUS
        ANI   IBF        ; LOOK AT IBF
        JNZ  UPIOUT      ; WAIT FOR IBF=0
        MOV  A,C          ; GET DATA FROM C
        OUT  DBBIN       ; OUTPUT DATA TO DBBIN
        RET               ; DONE, RETURN
    
```

Figure 5C. 8085A Code for Single Output Port Example

Figure 6A illustrates the case where UPI PORT 2 is used as an 8-bit input port. This configuration is termed UPI output-only as the master does not write (input) to the UPI but simply reads either the STATUS or the DBBOUT registers. In this example only the OBF flag is used. OBF signals the master that the UPI has placed new port data in DBBOUT. The UPI loops testing OBF. When OBF is clear, the master has read the previous data and UPI then reads its input port (PORT 2) and places this data in DBBOUT. It then waits on OBF until the master reads DBBOUT before reading the input port again. When the master wishes to read the input port data, Figure 6B, it simply checks for OBF being set in the STATUS register before reading DBBOUT. While this technique illustrates proper protocol, it should be noted that it is not meant to be a good method of using the UPI as an input port since the master would never get the newest status of the port.

```

UPI OUTPUT ONLY EXAMPLE—PORT 2 USED AS INPUT PORT
PORT DATA IS AVAILABLE IN DBBOUT

RESET:  JOBF  RESET      ; LOOP IF OBF=1 (DATA NOT READ)
        IN    A,P2      ; DBBOUT CLEAR, READ PORT
        OUT  DBB,A       ; TRANSFER PORT DATA TO DBBOUT
        JMP   RESET      ; WAIT FOR MASTER TO READ DATA
    
```

Figure 6A. Single Input Port Example

```

8085 SOFTWARE FOR UPI OUTPUT—ONLY EXAMPLE
INPUT DATA RETURNED IN REG. A

UPIIN:  IN    STATUS     ; READ UPI STATUS
        ANI   OBF        ; LOOK AT OBF
        JZ   UPIIN       ; WAIT UNTIL OBF=1
        IN    DBBOUT     ; READ DBBOUT
        RET               ; RETURN WITH DATA IN A
    
```

Figure 6B. 8085A Single Input Port Code

The above examples can easily be combined. Figure 7 shows UPI software to use PORT 1 as an output port simultaneously with PORT 2 as an input port. The program starts with the UPI checking IBF to see if the master has written data destined for the output port into DBBIN. If IBF is set, the UPI reads DBBIN and transfers the data to the output port (PORT 1). If IBF is not set or once the data is transferred to the output port if it was, OBF is tested. If OBF is reset (indicating the master has read DBBOUT), the input port (PORT 2) is read and transferred to DBBOUT. If OBF is set, the master has yet to read DBBOUT so the program just loops back to test IBF.

The master software is identical to the separate input/output examples; the master must test IBF

```

UPI INPUT/OUTPUT EXAMPLE—PORT 1 OUTPUT, PORT 2 INPUT
RESET:  JNIBF  OUT1      ; IF IBF=0, DO OUTPUT
        IN     A, DBB    ; IF IBF=1, READ DBBIN
        OUTL   P1, A     ; TRANSFER DATA TO PORT 1
OUT1:   JOBFL  RESET     ; IF OBF=1, GO TEST IBF
        IN     A, P2     ; IF OBF=0, READ PORT 2
        OUT    DBB, A    ; TRANSFER PORT DATA TO DBBOUT
        JMP    RESET     ; GO CHECK FOR INPUT
    
```

**Figure 7. Combination Output/Input Port Example**

and OBF before writing output port data into DBBIN or before reading input port from DBBOUT respectively.

In all of the three examples above, the UPI treats information from the master solely as data. There has been no need to check if DBBIN information is a command rather than data since the applications do not require commands. But what if both PORTs 1 and 2 were used as output ports? The UPI needs to know into which port to put the data. Let's use a command to select which port.

Recall that both commands and data pass through DBBIN. The state of the A<sub>0</sub> pin at the time of the write to DBBIN is used to distinguish commands from data. By convention, DBBIN writes with A<sub>0</sub>=0 are for data, and those with A<sub>0</sub>=1 are commands. When DBBIN is written into, F<sub>1</sub> (FLAG 1) is set to the state of A<sub>0</sub>. The UPI tests F<sub>1</sub> to determine if the information in the DBBIN register is data or command.

For the case of two output ports, let's assume that the master selects the desired port with a command prior to writing the data. (We could just use F<sub>1</sub> as a port select but that would not illustrate the subtle differences between commands and data). Let's define the port select commands such that BIT 1=1 if the next data is for PORT 1 (Write PORT 1=0000 0010) and BIT 2=1 if the next data is for PORT 2 (Write PORT 2=0000 0100). (The number of the set bit selects the port.) Any other bits are ignored. This assignment is completely arbitrary; we could use any command structure, but this one has the advantage of being simple.

Note that the UPI must "remember" from DBBIN write to write which port has been selected. Let's use F<sub>0</sub> (FLAG 0) for this purpose. If a Write PORT 1 command is received, F<sub>0</sub> is reset. If the command is Write PORT 2, F<sub>0</sub> is set. When the UPI finds data in DBBIN, F<sub>0</sub> is interrogated and the data is loaded into the previously selected port. The UPI software is shown in Figure 8A.

```

UPI DUAL OUTPUT PORT EXAMPLE—BOTH PORT 1 AND 2 OUTPUTS
COMMAND SELECTS DESIRED PORT
WRITE PORT 1—0000 0010 (02H)
WRITE PORT 2—0000 0100 (04H)

FLAG 0 USED TO REMEMBER WHICH PORT WAS SELECTED
BY LAST COMMAND.

RESET:  JNIBF  RESET     ; WAIT FOR MASTER INPUT
        IN     A, DBB    ; READ INPUT
        JF1    CMD      ; IF F1=1, COMMAND INPUT
        JF0    PORT2    ; INPUT IS DATA, TEST F0
        OUTL   P1, A    ; F0=0, SO OUTPUT TO PORT 1
        JMP    RESET     ; WAIT FOR NEXT INPUT
PORT2:  OUTL   P2, A    ; F0=1, SO OUTPUT TO PORT 2
        JMP    RESET     ; WAIT FOR NEXT INPUT
CMD:    JB1    PT1      ; TEST COMMAND BITS (BIT 1)
        JB2    PT2      ; TEST BIT 2
        JMP    RESET     ; NEITHER BIT SET, WAIT FOR INPUT
PT1:    CLR    F0       ; PORT 1 SELECTED, CLEAR F0
        JMP    RESET     ; WAIT FOR INPUT
PT2:    CLR    F0       ; PORT 2 SELECTED, SET F0
        CPL    F0
        JMP    RESET     ; WAIT FOR INPUT
    
```

**Figure 8A. Dual Output Port Example**

Initially, the UPI simply waits until IBF is set indicating the master has written into DBBIN. Once IBF is set, DBBIN is read and F<sub>1</sub> is tested for a command. If F<sub>1</sub>=1, the DBBIN byte is a command. Assuming a command, BIT 1 is tested to see if the command selected PORT 1. If so, F<sub>0</sub> is cleared and the program returns to wait for the data. If BIT 1=0, BIT 2 is tested. If BIT 2 is set, PORT 2 is selected so F<sub>0</sub> is set. The program then loops back waiting for the next master input. This input is the desired port data. If BIT 2 was not set, F<sub>0</sub> is not changed and no action is taken.

When IBF=1 is again detected, the input is again tested for command or data. Since it is necessarily data, DBBIN is read and F<sub>0</sub> is tested to determine which port was previously selected. The data is then output to that port, following which the program waits for the next input. Note that since F<sub>0</sub> still selects the previous port, the next input could be more data for that port. The port selection command could be thought of as a port select flip-flop control; once a selection is made, data may be repeatedly written to that port until the other port is selected. Master software, Figure 8B, simply must check IBF before writing either a command or data to DBBIN. Otherwise, the master software is straightforward.

For the sake of completeness, UPI software for implementing two input ports is given in Figure 9. This case is simpler than the dual output case since the UPI can assume that all writes to DBBIN are port selection commands so no command/data testing is required. Once the Port Read command is input, the selected port is read and the port data is placed in DBBOUT. Note that in this case F<sub>0</sub> is used as a UPI

error indicator. If the master happened to issue an invalid command (a command without either BIT 1 or 2 set), F<sub>0</sub> is set to notify the master that the UPI did not know how to interpret the command. F<sub>0</sub> is also set if the master commanded a port read before it had read DBBOUT from the previous command. The UPI simply tests OBF just prior to loading DBBOUT and if OBF=1, F<sub>0</sub> is set to indicate the error.

All of the above examples are, in themselves, rather trivial applications of the UPI although they could easily be incorporated as one of several tasks in a UPI handling multiple small tasks. We have covered them primarily to introduce the UPI concept and to illustrate some master/UPI protocol. Before moving on to more realistic UPI applications, let's discuss two UPI features that do not directly relate to the master/UPI protocol but greatly enhance the UPI's transfer capability.

In addition to the OBF and IBF bits in the STATUS register, these flags can also be made available directly on two port pins. These port pins can then be used as interrupt sources to the master. By executing an EN FLAGS instruction, PORT 2 pin 4 reflects the condition of OBF and PORT 2 pin 5 reflects the inverted condition of IBF ( $\overline{\text{IBF}}$ ). These dedicated outputs can then be enabled or disabled via their respective port bit values; i.e., P<sub>24</sub> reflects OBF as long as an instruction is executed which sets P<sub>24</sub> (i.e. ORL P<sub>2</sub>,#10H). The same action applies to the  $\overline{\text{IBF}}$  output except P<sub>25</sub> is used. Thus P<sub>24</sub> may serve as a DATA AVAILABLE interrupt output. Likewise for P<sub>25</sub> as a READY-TO-ACCEPT-DATA interrupt. This greatly simplifies interrupt-driven master-slave data transfers.

```

: 8085 SOFTWARE FOR DUAL OUTPUT PORT EXAMPLE
: THIS ROUTINE WRITES DATA IN REG. C TO PORT 1
: (SAME ROUTINE FOR PORT 2—JUST CHANGE COMMAND)

PORT1: IN    STATUS    ; READ UPI STATUS
        ANI    IBF      ; LOOK AT IBF
        JNZ   PORT1    ; WAIT UNTIL IBF=0
        MVI   A, 0000010B ; LOAD WRITE PORT1 CMD
        OUT   UPICMD    ; OUTPUT TO UPI COMMAND PORT
P1:    IN    STATUS    ; READ UPI STATUS AGAIN
        ANI    IBF      ; LOOK AT IBF
        JNZ   P1       ; WAIT UNTIL COMMAND ACCEPTED
        MOV   A, C      ; GET DATA FROM C
        OUT   DBBIN     ; OUTPUT TO DBBIN
        RET              ; DONE, RETURN
    
```

**Figure 8B. 8085A Dual Output Port Example Code**

The UPI also supports a DMA transfer interface. If an EN DMA instruction is executed, PORT 2 pin 6 becomes a DMA Request (DRQ) output and P<sub>27</sub> becomes a high impedance DMA Acknowledge

```

: UPI DUAL INPUT PORT EXAMPLE—BOTH PORT 1 AND 2 INPUTS
: COMMAND SELECTS WHICH PORT IS TO BE READ
: FLAG 0 USED AS ERROR FLAG

RESET: JNIBF RESET    ; WAIT FOR INPUT
        CLR   F0      ; CLEAR ERROR FLAG
        IN   A, DBB    ; READ INPUT (COMMAND)
        JB1  PT1      ; TEST BIT 1 (PORT 1)
        JB2  PT2      ; TEST BIT 2 (PORT 2)
ERROR:  CPL   F0      ; ERROR—COMPLEMENT F0
        JMP  RESET    ; WAIT FOR INPUT
PT1:   IN   A, P1      ; READ PORT 1
        JOBF ERROR    ; TEST OBF BEFORE LOADING DBBOUT
        OUT  DBB, A    ; LOAD PORT 1 DATA INTO DBBOUT
        JMP  RESET    ; WAIT FOR INPUT
PT2:   IN   A, P2      ; READ PORT 2
        JOBF ERROR    ; TEST OBF BEFORE LOADING DBBOUT
        OUT  DBB, A    ; LOAD PORT 2 DATA INTO DBBOUT
        JMP  RESET    ; WAIT FOR INPUT
    
```

**Figure 9. Dual Input Port Example**

( $\overline{\text{DACK}}$ ) input. Any instruction which would normally set P<sub>26</sub> now sets DRQ. DRQ is cleared when  $\overline{\text{DACK}}$  is low and either  $\overline{\text{RD}}$  or  $\overline{\text{WR}}$  is low. When  $\overline{\text{DACK}}$  is low, CS and A<sub>0</sub> are forced low internally which allows data bus transfers between DBBOUT or DBBIN to occur, depending upon whether  $\overline{\text{WR}}$  or  $\overline{\text{RD}}$  is true. Of course, the function requires the use of an external DMA controller.

Now that we have discussed the aspects of the UPI protocol and data transfer interfaces, let's move on to the actual applications.

## EXAMPLE APPLICATIONS

Each of the following three sections presents the hardware and software details of a UPI application. Each application utilizes one of the protocols mentioned in the last section. The first example is a simple 8-digit LED display controller. This application requires only that the UPI perform input operations from the DBBIN; DBBOUT is not used. The reverse is true for the second application: a sensor matrix controller. The final application involves both DBBOUT and DBBIN operations: a combination serial/parallel I/O device.

The core master processor system with which these applications were developed is the iSBC 80/30 single board computer. This board provides an especially convenient UPI environment since it contains a dedicated socket specifically interfaced for the UPI-41A. The 80/30 uses the 8085A as the master processor. The I/O and peripheral complement on the 80/30 include 12 vectored priority interrupts (8 on an 8259 Programmable Interrupt Controller and 4 on the 8085A itself), an 8253 Programmable Interval Timer supplying three 16-bit programmable timers (one is dedicated as a programmable baud rate generator), a high speed serial channel provided by a 8251 Programmable USART, and 24 parallel I/O

lines implemented with an 8255A Programmable Parallel Interface. The memory complement contains 16K bytes of RAM using 2117 16K bit Dynamic RAMs and the 8202 Dynamic RAM Controller, and up to 8K bytes of ROM/EPROM with sockets compatible with 2716, 2758, or 2332 devices. The 80/30's RAM uses a dual port architecture. That is, the memory can be considered a global system resource, accessible from the on-board 8085A as well as from remote CPUs and other devices via the MULTIBUS. The 80/30 contains MULTIBUS control logic which allows up to 16 80/30s or other bus masters to share the same system bus. (More detailed information on the iSBC 80/30 and other iSBC products may be found in the latest Intel *Systems Data Catalog*.)

A block diagram of the iSBC 80/30 is shown in Figure 10. Details of the UPI interface are shown in Figure 11. This interface decodes the UPI registers in the following format:

Register	Operations
Read STATUS	IN E5H
Write DBBIN (command)	OUT E5H
Read DBBOUT (data)	IN E4H
Write DBBIN (data)	OUT E4H

## 8-Digit Multiplexed LED Display

The traditional method of interfacing an LED display with a microprocessor is to use a data latch along with a BDC-to-7-segment decoder for each digit of the display. Thus two ICs, seven current limiting resistors, and about 45 connections are required for each digit. These requirements are, of course, multiplied by the total number of digits desired. The obvious disadvantages of this method are high parts count and high power dissipation since each digit is "ON" continuously. Instead, a scheme of time multiplexing the display can be used to decrease both parts count and power dissipation.

Display multiplexing basically involves connecting the same segment (a, b, c, d, e, f, or g) of each digit in parallel and driving the common digit element (anode or cathode) of each digit separately. This is shown schematically in Figure 12. The various digits of the display are not all on at once; rather, only one digit at a time is energized. As each digit is energized, the appropriate segments for that digit are turned on. Each digit is enabled in this way, in sequence, at a rate fast enough to ensure that each digit appears to be "ON" continuously. This implies that the display must be "refreshed" at periodic intervals to keep the digits flicker-free. If the CPU had to handle this task, it would have to suspend normal

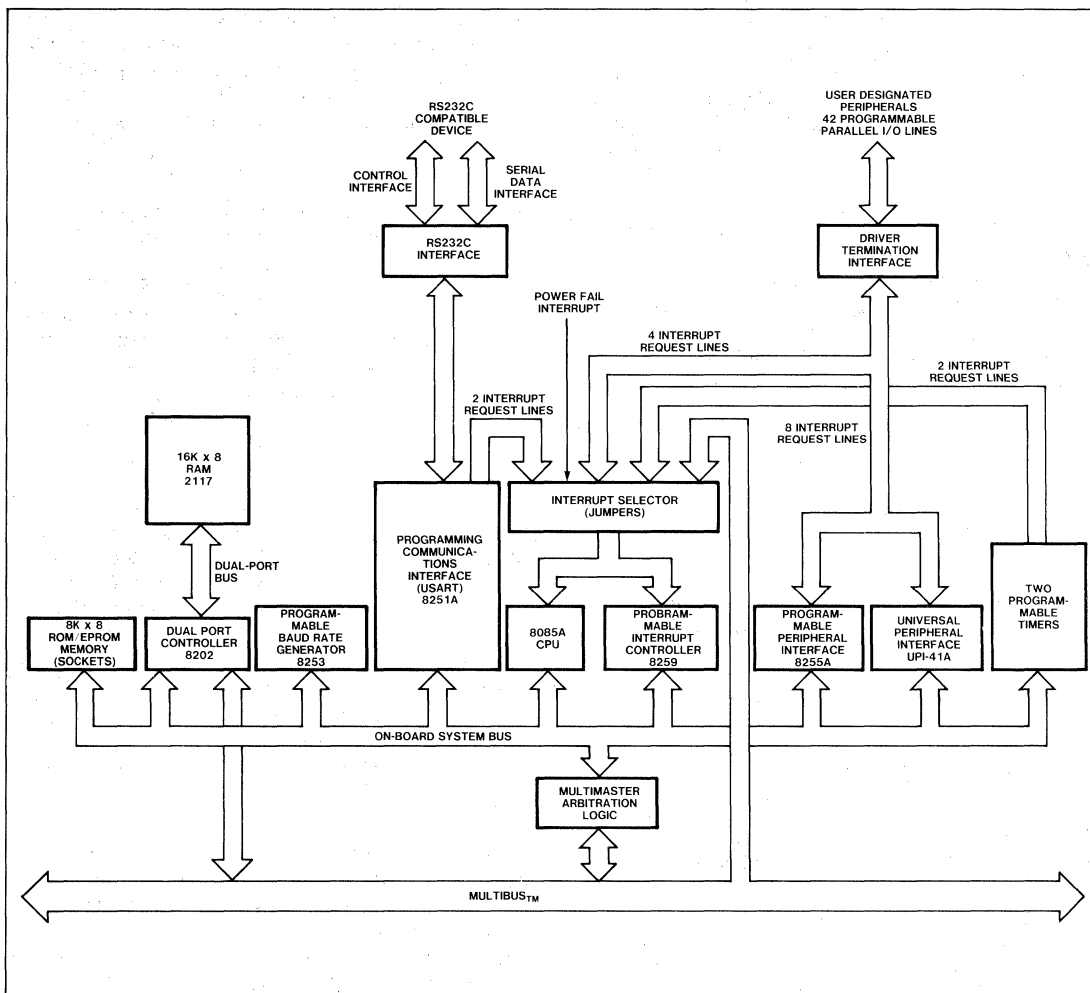
processing, go update the display, and then return to its normal flow. This extra burden is ideally handled by a UPI. The master CPU could simply give characters to the UPI and let the UPI do the actual segment decoding, display multiplexing, and refreshing.

As an example of this technique, Figure 13 shows the UPI controlling an 8-digit LED display. All digit segments are connected in parallel and are driven through segment drivers by the UPI PORT 1. The lower 3 bits of PORT 2 are inputs to a 3-to-8 decoder which selects an individual digit through a digit driver. A fourth PORT 2 line is used as a decoder enable input. The remaining PORT 2 lines plus the TEST 0 and TEST 1 inputs are available for other tasks.

Internally, the UPI uses the counter/timer in the interval timer mode to define the interval between display refreshes. Once the timer is loaded with the desired interval and started, the UPI is free to handle other tasks. It is only when a timer overflow interrupt occurs that the UPI handles the short display multiplexing routine. The display multiplexing can be considered a background task which is entirely interrupt-driven. The amount of time spent multiplexing is such that there is ample time to handle a non-timer task in the UPI foreground. (We'll discuss this timing shortly.)

When a timer interrupt occurs, the UPI turns off all digits via the decoder enable. The next digit's segment contents are retrieved from the internal data memory and output via PORT 1 to the segment drivers. Finally, the next digit's location is placed on PORT 2 (P20-P22) and the decoder enabled. This displays the digit's segment information until the next interrupt. The timer is then restarted for the next interval. This process continues repeatedly for each digit in sequence.

As a prelude to discussing the UPI software, let's examine the internal data memory structure used in this application, Figure 14. This application requires only 14 of the 64 total data memory locations. The top eight locations are dedicated to the Display Map; one location for each digit. These locations contain the segment and decimal point information for each character. Just how characters are loaded into this section of memory is covered shortly. Register R7 of Register Bank 1 is used as the temporary Accumulator store during the interrupt service routines. Register R3 stores the digit number of the next digit to be displayed. R2 is a temporary storage register for characters during input routine. R0 is



**Figure 10. iSBC 80/30 Block Diagram**

the offset pointer pointing to the Display Map location of the next digit. That makes 12 locations so far. The remaining two locations are the two stack locations required to store the return address plus status during the timer and input interrupt service routines. The remaining unused locations, all of Register Bank 0, 14 bytes of stack, 4 in Register Bank 1, and 24 general purpose RAM locations, are all available for use by any foreground task.

The UPI software consists of only three short routines. One, INIT, is used strictly during initialization. DISPLA is the multiplexing routine called at a timer interrupt. INPUT is the character input handler called at an IBF interrupt. The flow

charts for these routines are shown in Figures 14A through 14C.

INIT initializes the UPI by simply turning off all segment and digit drivers, filling the Display Map with blank characters, loading and starting the timer, and enabling both timer and IBF interrupts. Although the flow chart shows the program looping at this point, it is here that the code for any foreground task is inserted. The only restrictions on this foreground task are that it not use I/O lines dedicated to the display and that it not require dedicated use of the timer. It could share the timer if precautions are taken to ensure that the display will still be refreshed at the required interval.

# APPLICATIONS

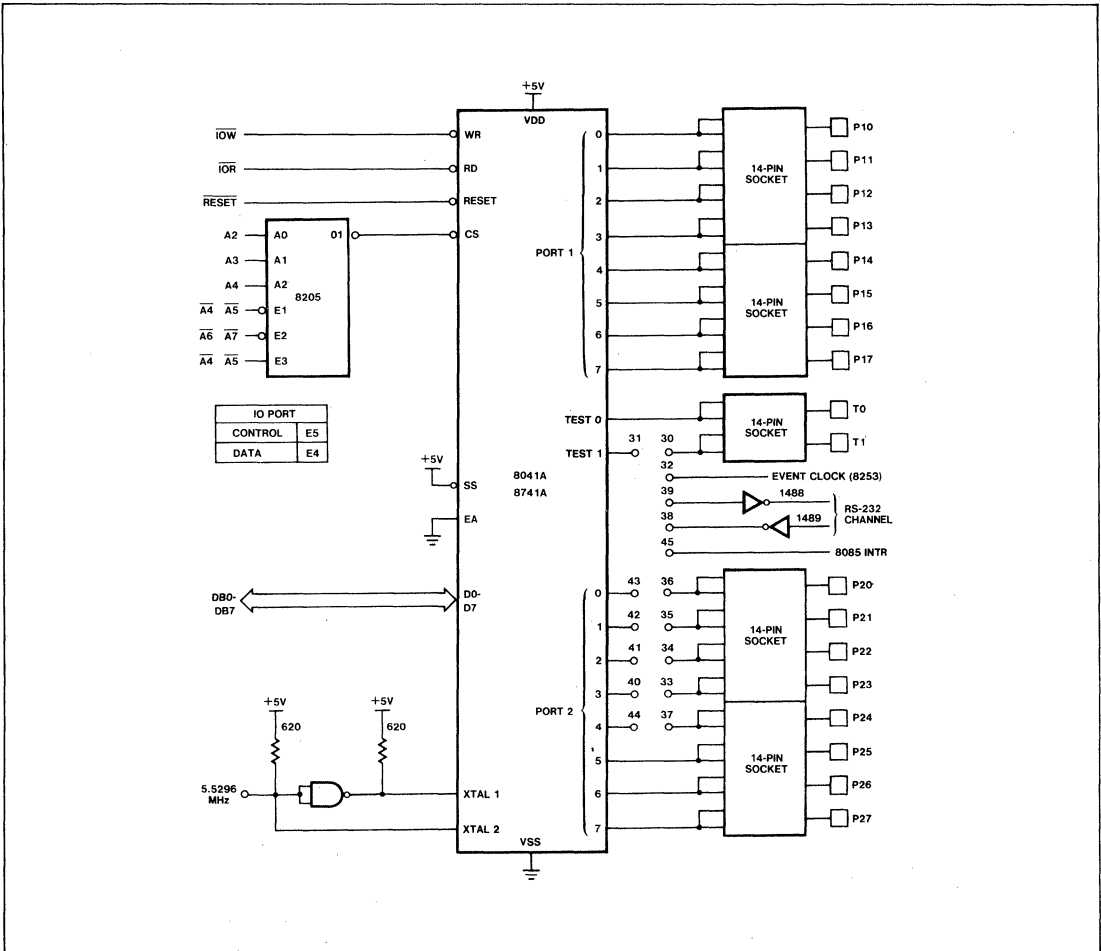


Figure 11. UPI Interface on iSBC 80/30

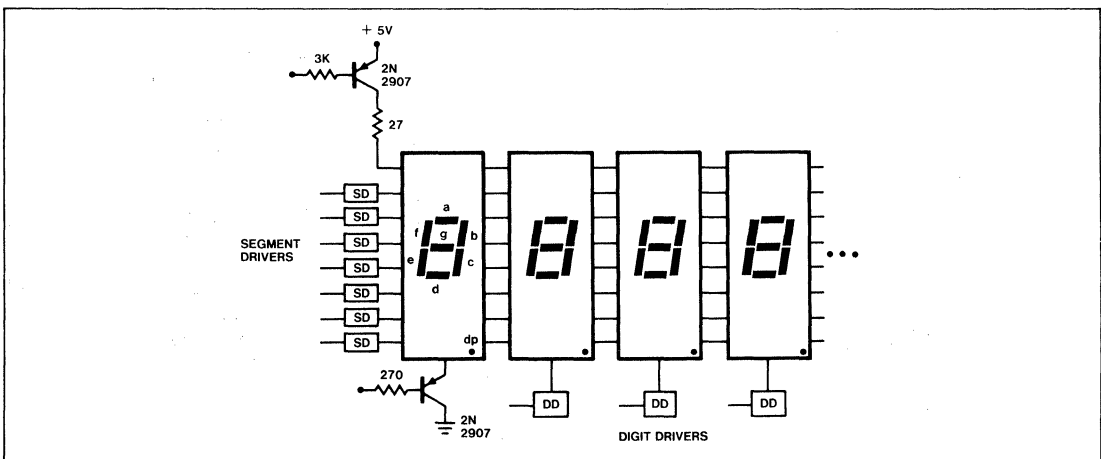
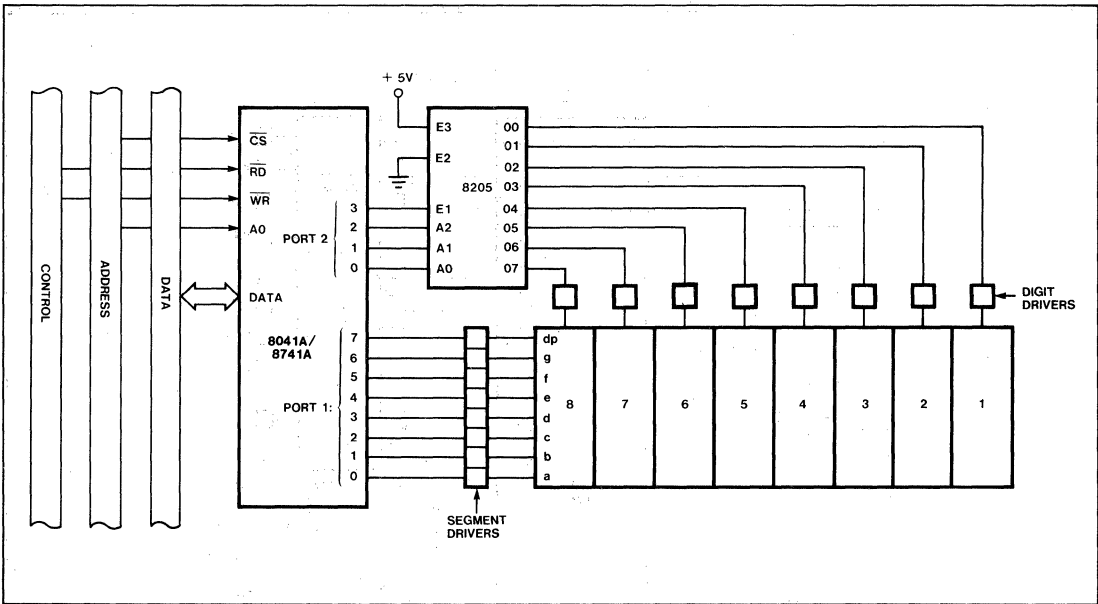
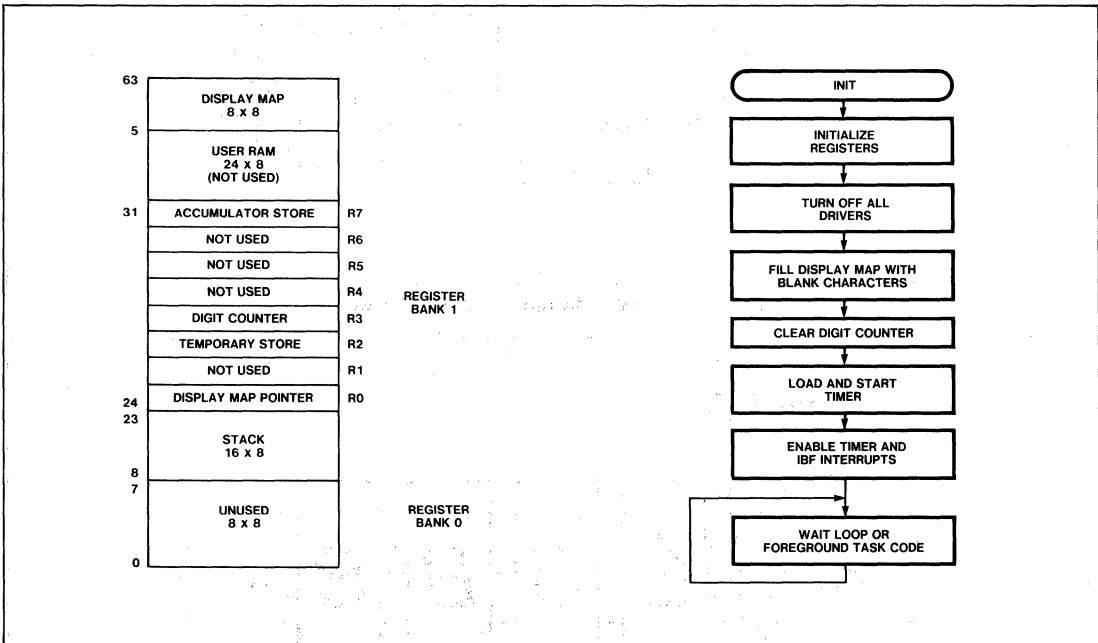


Figure 12. LED Multiplexing

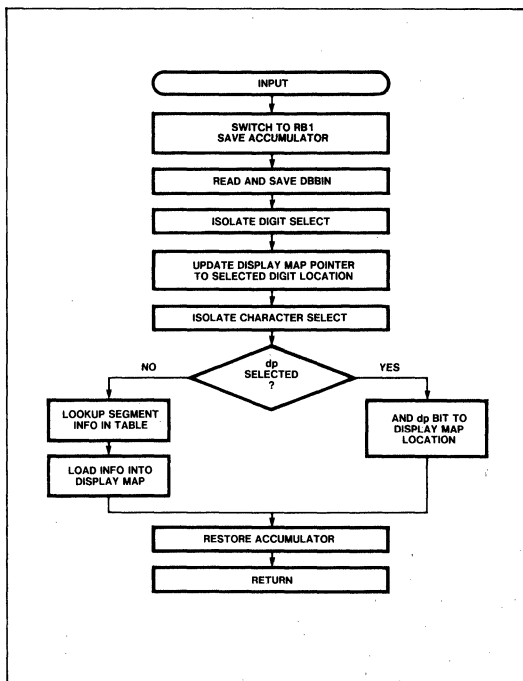


**Figure 13. UPI Controlled 8-Digit LED Display**



**Figure 14. LED Display Controller Data Memory Allocation**

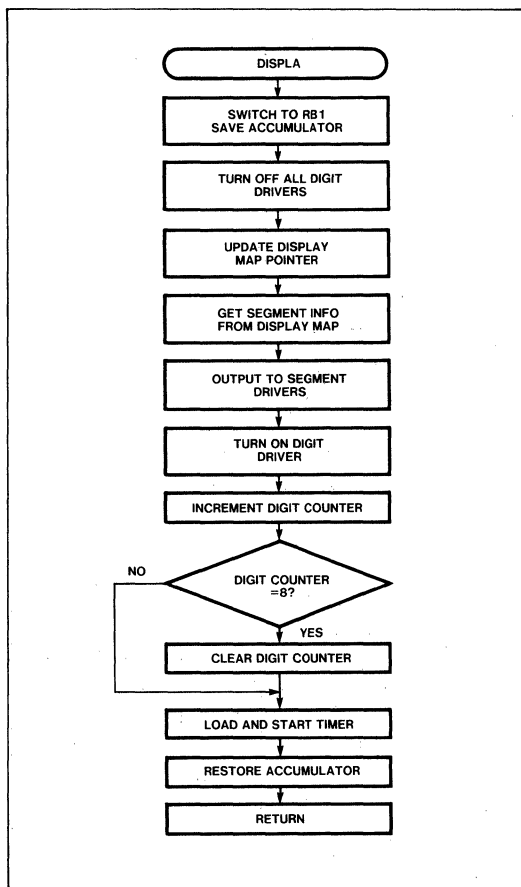
**Figure 14A. INIT Routine Flow**



**Figure 14B. INPUT Routine Flow**

The INPUT routine handles the character input. It is called when an IBF interrupt occurs. After the usual swapping of register banks and saving of the accumulator, DBBIN is read and stored in register R<sub>2</sub>. DBBIN contains the Display Data Word. The format for this word, Figure 15, has two fields: Digit Select and Character Select. The Digit Select field selects the digit number into which the character from the Character Select field is placed. Notice that the character set is not limited strictly to numerics, some alphanumeric capability is provided. Once DBBIN is read, the offset for the selected digit is computed and placed in the Display Map Pointer R<sub>0</sub>. Next the segment information for the selected character is found through a look-up table starting in page 3 of the program memory. This segment information is then stored at the location pointed at by the Display Map Pointer. If the Character Select field specified a decimal point, the segment corresponding to the decimal point is ANDed into the present segment information for that digit. After the accumulator is restored, execution is returned to the main program.

The DISPLA routine simply implements the multiplexing actions described earlier. It is called whenever a timer interrupt occurs. After saving pre-



**Figure 14C. DISPLA Routine Flow**

interrupt status by switching register banks and storing the Accumulator, all digit drivers are turned off. The Display Map Pointer is then updated using the Current Digit Register to point at that digit's segment information in the Display Map. This information is output to PORT 1; the segment drivers. The number of the current digit, R<sub>3</sub>, is then sent to the digit select decoder and the decoder is enabled. This turns on the current digit. The digit counter is incremented and tested to see if all eight digits have been refreshed. If so, the digit counter is reset to zero. If not, nothing is done. Finally, the timer is loaded and restarted, the Accumulator is restored, and the routine returns execution to the main program. Thus DISPLA refreshes one digit each time it is CALLED by the timer interrupt. The digit remains on until the next time DISPLA is executed.

The UPI software listing is included as Appendix A1. Appendix A2 shows the 8085A test routine used



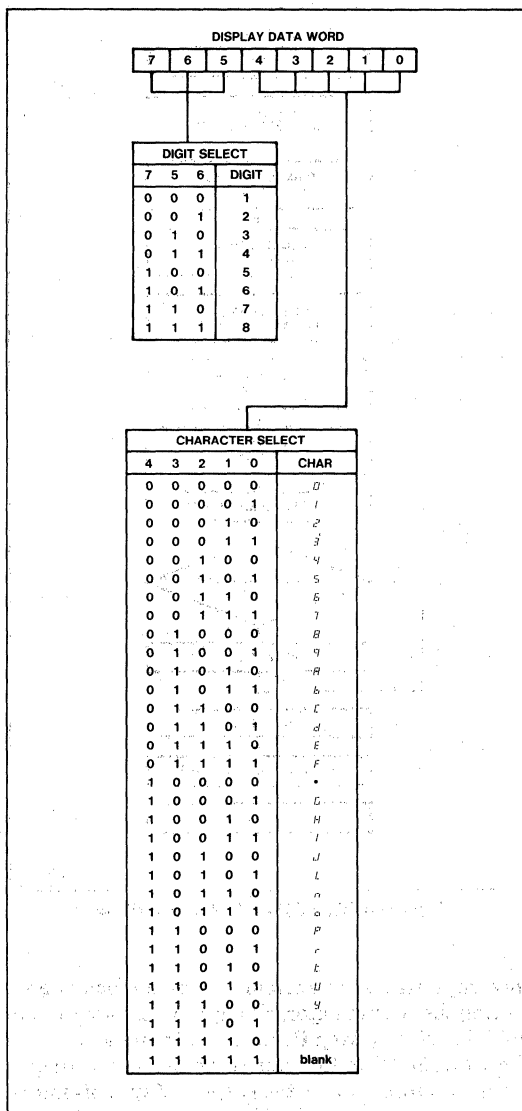


Figure 15. LED Display Controller Display Data Word Format

to display the contents of a display buffer on the display. The 8085A software takes care of the display digit numbering. Since the application is input-only for the UPI, the only protocol required is that the master must test IBF before writing a Display Data Word into DBBIN.

On the iSBC 80/30, the UPI frequency is at 5.5296 MHz. To obtain a flicker-free display, the whole display must be refreshed at a rate of 50 Hz or greater.

If we assume a 50 Hz refresh rate and an 8-digit display, this means the DISPLA routine must be CALLED 50×8 or 400 times/sec. This transfers, using the timer interval of 87  $\mu$ s at 5.5296 MHz, to a timer count of 227. (Recall from the UPI-41A *User's Manual* that the timer is an "8-bit up-counter".) Hence the TIME equate of 227D in the UPI listing. Obviously, different frequency sources or display lengths would require that this equate be modified.

With the UPI running at 5.5296 MHz, the instruction cycle time is 2.713  $\mu$ s. The DISPLA routine requires 28 instruction cycles, therefore, the routine executes in 76  $\mu$ s. Since DISPLA is CALLED 400 times/sec, the total time spent refreshing the display during one second is then 30 ms or 3% of the total UPI time. This leaves 97.0% for any foreground tasks that could be added.

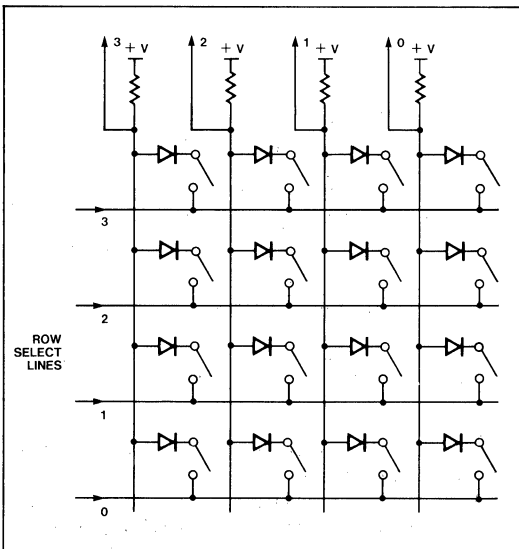
While the basic UPI software is useful just as it stands, there are several enhancements that could be incorporated depending on the application. Auto-incrementing of the digit location could be added to the input routine to alleviate the need for the master to keep track of digit numbers. This could be (optionally) either right-handed or left-handed entry a la TI or HP calculators. The character set could be easily modified by simply changing the lookup table. The display could be expanded to 16 digits at the expense of one additional PORT 2 digit select line, the replacement of the 3-to-8 decoder with a 4-to-16 decoder, and 8 more Display Map locations.

Now let's move on to a slightly more complex application that is UPI output-only—a sensor matrix controller.

### Sensor Matrix Controller

Quite often a microprocessor system is called upon to read the status of a large number of simple SPST switches or sensors. This is especially true in a process or industrial control environment. Alarm systems are also good examples of systems with a large sensor population. If the number of sensors is small, it might be reasonable to dedicate a single input port pin for each sensor. However, as the number of sensors increase, this technique becomes very wasteful. A better arrangement is to configure the sensors in a matrix organization like that shown in Figure 16. This arrangement of 16 sensors requires only 4 input and 4 output lines; half the number needed if dedicated inputs were used. The line saving becomes even more substantial as the number of sensors increases.

In Figure 16, the basic operation of the matrix involves scanning individual row select lines in sequence while reading the column return lines. The state of any particular sensor can then be determined by decoding the row and column information. The typical configuration pulls up the column return lines and the selected row is held low. Deselected rows are held high. Thus a return line remains high for an open sensor on the selected row and is pulled low for a closed sensor. Diode isolation is used to prevent a phantom closure which would occur when a sensor is closed on a selected row and there are two or more closures on a deselected row. Germanium diodes are used to provide greater noise margin at the return line input.

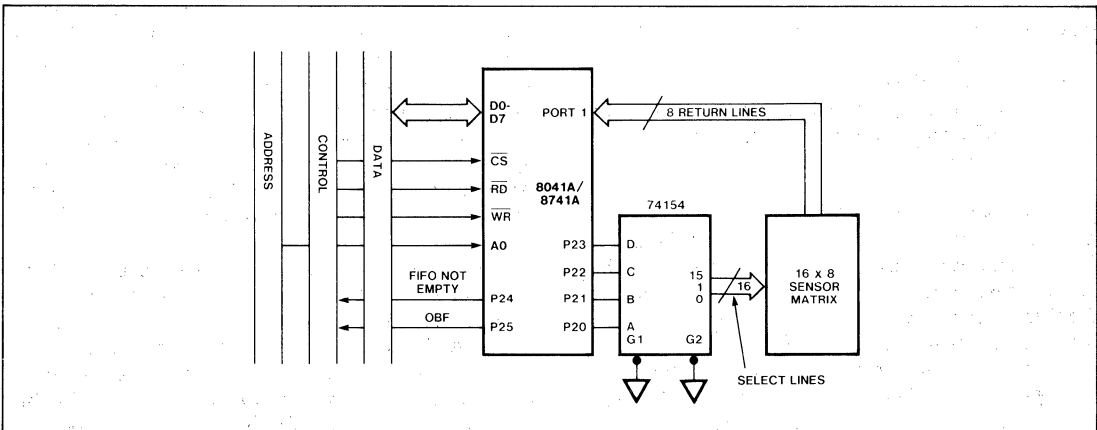


**Figure 16. 4x4 Sensor Matrix**

If the main processor was required to control such a matrix it would periodically have to output at the row port and then read the column return port. The processor would need to maintain in memory a map of the previous state of the matrix. A comparison of the new return information to the old information would then be made to determine whether a sensor change had occurred. Any changes would be processed as needed. A row counter and matrix map pointer also require maintenance each scan. Since in most applications sensors change very slowly compared to most processing actions, the processor probably would scan the rows only periodically with other tasks being processed between scans.

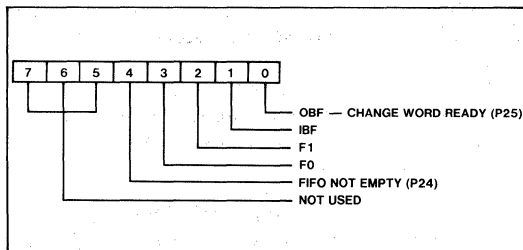
Rather than require the processor to handle the rather mundane tasks of scanning, comparing, and decoding the matrix, why not use a dedicated processor? The UPI is perfect.

Figure 17 shows a UPI configuration for controlling up to 128 sensors arranged in a 16x8 matrix. The 4-to-16 line decoder is used as the row selector to save port pins and provides the expansion to 128 sensors over the maximum of 64 sensors if the port had been used directly. It also helps increase the port drive capability. The column return lines go directly into PORT 1. Features of this design include complete matrix management. As the UPI scans the matrix it compares its present status to the previous scan. If any change is detected, the location of the change is decoded and loaded, along with the sensor's present state, into DBBOUT. This byte is called a Change Word. The Master processor has only to read one byte to determine the status and coordinate of a changed sensor. If the master had not read a previous Change Word in DBBOUT (OBF=1) before a new sensor change is detected, the new Change

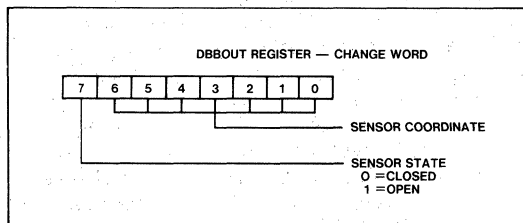


**Figure 17. 128 Sensor Matrix Controller**

Word is loaded into an internal FIFO. This FIFO buffers up to 40 changes before it fills. The status of the FIFO and OBF is made available to the master either by polling the UPI STATUS register, Figure 18A, or as interrupt sources on port pins P<sub>24</sub> and P<sub>25</sub> respectively, Figure 17. The FIFO NOT EMPTY pin and bit are true as long as there are changes not yet read in the FIFO. As long as the FIFO is not empty, the UPI monitors OBF and loads new Change Words from the FIFO into DBBOUT. Thus, the UPI provides complete FIFO management.



**Figure 18A. Sensor Matrix Status Register Format**

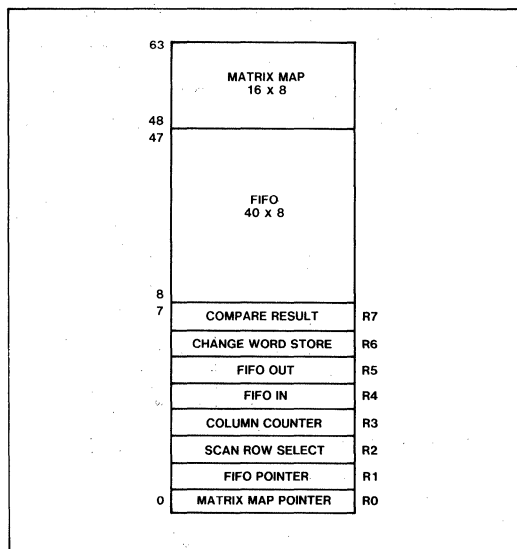


**Figure 18B. Sensor Matrix Change Word Format**

Internally, the matrix scanning software is programmed to run as a foreground task. This allows the timer/counter to be used by any background task although the hardware configuration leaves only 2 inputs (TEST 0 and TEST 1) plus 2 I/O port pins available. Also, to add a background task, the FIFO would have to be made smaller to accommodate the needed register and data memory space. (It would be possible however to turn the table here and make the scanning software timer/counter interrupt-driven where the timer times the scan interval.)

The data memory organization for this application is shown in Figure 19. The upper 16 bytes form the Matrix Map and store the sensor states from the previous scan; one bit for each sensor. The Change Word FIFO occupies the next 40 locations. (The top and bottom addresses of this FIFO are treated as equate variables in the program so that the FIFO size may easily be changed to accommodate the register needs of other tasks.) Register R<sub>0</sub> serves as a pointer into the matrix map area for comparisons

and updates of the sensor status. R<sub>1</sub> is a general FIFO pointer. The FIFO is implemented as a circular buffer with In and Out pointer registers which are stored in R<sub>4</sub> and R<sub>5</sub> respectively. These registers are moved into FIFO pointer R<sub>1</sub> for actual transfers into or out of the FIFO. R<sub>2</sub> is the Row Select Counter. It stores the number of the row being scanned.



**Figure 19. Sensor Matrix Data Memory Map**

Register R<sub>3</sub> is the Column Counter. This counter is normally set to 00H; however, when a change is detected somewhere in a particular row, it is used to inspect each sensor status bit individually for a change. When a changed counter sensor bit is found, the Row Select Counter and Column Counter are combined to give the sensor's matrix coordinate. This coordinate is temporarily stored in the Change Word Store, register R<sub>6</sub>. Register R<sub>7</sub> is the Compare Result. As each row is scanned, the return information is Exclusive-OR'd with the return information from the previous scan of that row. The result of this operation is stored in R<sub>7</sub>. If R<sub>7</sub> is zero, there have been no changes on that row. A non-zero result indicates at least one changed sensor.

The basic program operation is shown in the flow chart of Figure 20. At RESET, the software initializes the working registers, the ports, and clears the STATUS register. To get a starting point from which to perform the sensor comparisons, the current status of the matrix is read and stored in the Matrix Map. At this point, the UPI begins looking for changed sensors starting with the first row.

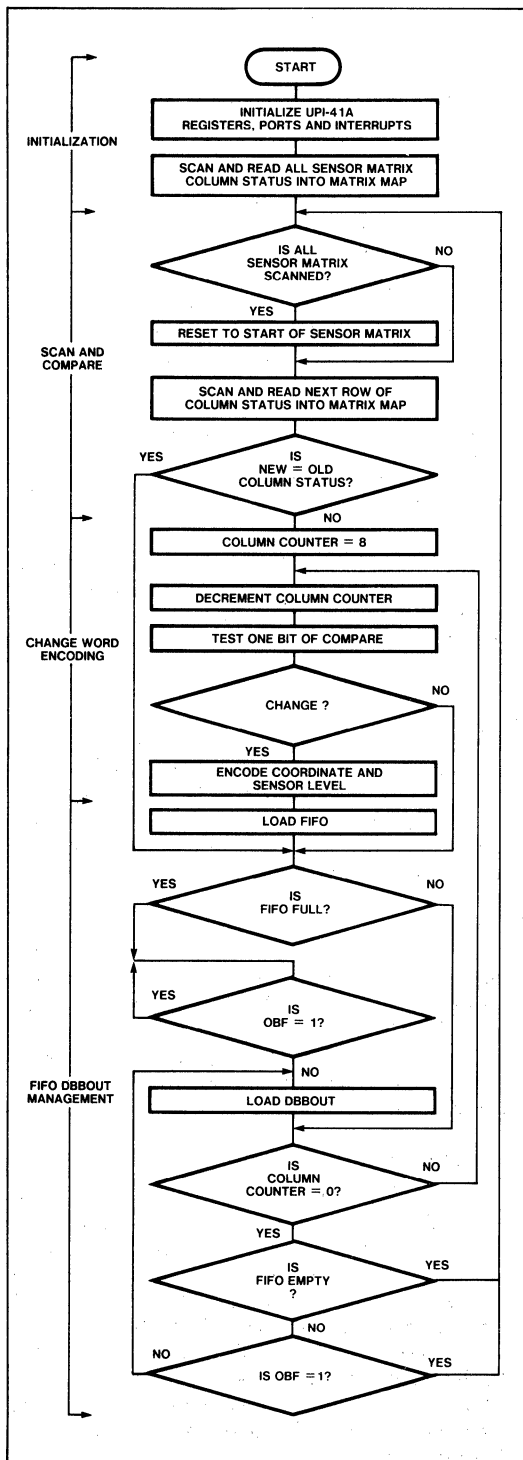


Figure 20. Sensor Matrix Controller Flow Chart

Before delving further into the flow, let's pause to describe the general format of the operation. The UPI scans the matrix one row at a time. If no changes are detected on a particular row, the UPI simply moves to the next row after checking the status of DBBOUT and the FIFO. If a change is detected, the UPI must check each bit (sensor) within the row to determine the actual sensor location. (More than one sensor on the scanned row could have changed.) Rather than test all 8 bits of the row before checking the DBBOUT and FIFO status again, the UPI performs the status check in between each of the bit tests. This ensures the fastest response to the master reading previous Change Words from DBBOUT and the FIFO.

With this general overview in mind, let's go first thru the flow chart assuming we are scanning a row where no changes have occurred. Starting at the Scan-and-Compare section, the UPI first checks if the entire matrix has been scanned. If it has, the various pointers are reset. If not, the address of the next row is placed on PORTs 20 thru 23. This selects the desired row. The state of the row is then read on PORT 1; the column return lines. This present state is compared to the previous state by retrieving the previous state from the matrix map and performing an Exclusive-OR with the present state. Since we are assuming that no change has occurred, the result is zero. No coordinate decoding is needed and the flow branches to the FIFO-DBBOUT Management section.

The FIFO-DBBOUT Management section simply maintains the FIFO and loads DBBOUT whenever Change Words are present in the FIFO and DBBOUT is clear (OBF=0). The section first tests if the FIFO is full. (If we assume our "no-change" row is the first row scanned, the FIFO obviously would not be full.) If it is, the UPI waits until OBF=0, at which point the next Change Word is retrieved from the FIFO and placed in DBBOUT. This "unfills" the FIFO making room for more Change Words. At this point, the Column Counter, R3, is checked. For rows with no changes, the Column Counter is always zero so the test simply falls through. (We cover the case for changes shortly.) Now the FIFO is tested for being empty. If it is, there is no sense in any further tests so the flow simply goes back up to scan the next row. If the FIFO is not empty, DBBOUT is tested again through OBF. If a Change Word is in DBBOUT waiting for the master to read it, nothing can be done and the flow likewise branches up for the next row. However, if the DBBOUT is free and remembering that the previous test showed that the FIFO was not empty, DBBOUT is loaded with the next Change Word and the last two conditional tests repeat.

Now let's assume the next row contains several changed sensors. Like before, the row is selected, the return lines read, and the sensor status compared to the previous scan. Since changes have occurred, the Exclusive-OR result is now non-zero. Any 1's in the result reflect the positions of the changed sensors. This non-zero result is stored in the Compare Result register, R7. At this point, the Column Counter is preset to 8. To determine the changed sensors' locations, the Compare Result register is shifted bit-by-bit to the left while decrementing the Column Counter. After each shift, BIT 7 of the result is tested. If it is a one, a changed sensor has been found. The Column Counter then reflected the sensor's matrix column position while the Scan Row Select register holds it row position. These registers are then combined in R6, the Change Word Store, to form the sensor's matrix coordinate section of the Change Word. The 8th bit of the Change Word Store is coded with the sensor's present state (Figure 18). This byte forms the complete Change Word. It is loaded into the next available FIFO position. If BIT 7 of the Compare Result had been zero, that particular sensor had not changed and the coordinate decoding is not performed.

In between each shift, test, and coordinate encode (if necessary), the FIFO-DBBOUT Management is performed. It is the Column Counter test within this section that routes the flow back up to the Change Word Encoding section if the entire Compare Result (row) has not been shifted and tested.

The FIFO is implemented as a circular buffer with IN and OUT pointers (R4 and R5 respectively). The operations of the FIFO is best understood using an example, Figure 21. This series of figures show how the FIFO, DBBOUT, and OBF interact as changes are detected and Change Words are read by the master. The letters correspond to sequential Change Words being loaded into the FIFO. Note that the figures show only a 4x8 FIFO however, the principles are the same in the 40x8 FIFO.

Figure 21A shows the condition where no Change Words have been loaded into the FIFO or DBBOUT. In Figure 21B a change, "A", has been detected, decoded, and loaded into the FIFO at the location equal to the value of the FIFO-IN pointer. The FIFO-OUT pointer is reset to the bottom of the FIFO since it had reached the FIFO top. Now that a Change Word is in the FIFO, OBF is checked to see if DBBOUT is empty. Because OBF=0, DBBOUT is empty and the Change Word is loaded from the FIFO location pointed at by the FIFO-OUT pointer. This is shown in Figure 21C. Loading DBBOUT automatically sets OBF. OBF remains set until the

master reads DBBOUT. Figures 21D and 21E show two more Change Words loaded into the FIFO. In Figure 21F the first Change Word is finally read by the master resetting OBF. This allows the next Change Word to be loaded into DBBOUT. Note that each time the FIFO is loaded, the FIFO-IN pointer increments. Each time DBBOUT is read the FIFO-OUT pointer increments unless there are no more Change Words in the FIFO. Both pointers wrap-around to the bottom once they reach the FIFO top. The remaining figures show more Change Words being loaded into the FIFO. When the entire FIFO fills and DBBOUT can not be loaded (OBF=1), scanning stops until the master reads DBBOUT making room for more Change Words.

As was mentioned earlier, two interrupt outputs to the master are available: Change Word Ready (P25, OBF) and FIFO NOT EMPTY (P24). The Change Word Ready interrupt simply reflects OBF and is handled automatically by the UPI since an EN FLAGS instruction is executed during initialization. The FIFO NOT EMPTY interrupt is generated and cleared as appropriate, each pass through the FIFO management code.

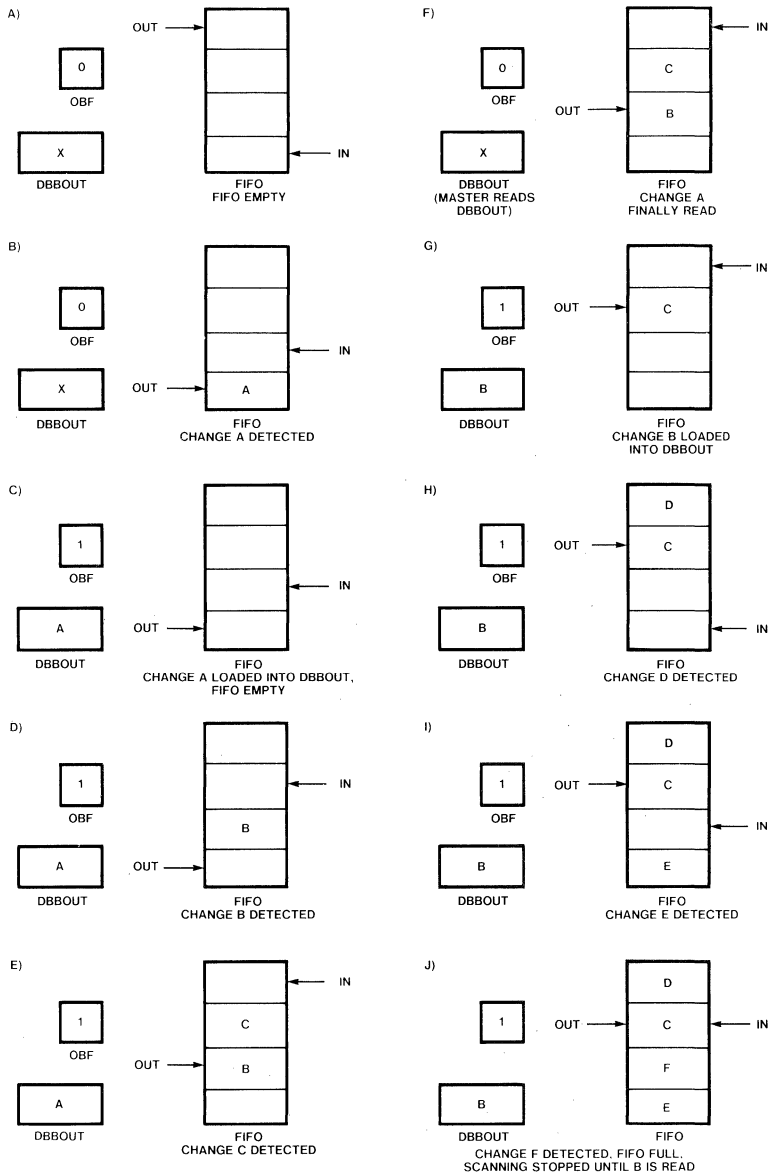
No debouncing is provided although it could be added. Rather, the scan time is left as an equate variable so that it could be varied to account for both debounce time and expected sensor change rates. The minimum scan time for this application is 2msec when using a 6MHz clock. Since the matrix controller is coded as a foreground task, scan time simply uses a software delay loop.

The UPI software is included as Appendix B1. Appendix B2 is 8085A test software which builds a Change Word buffer starting at BUFSRT. This software simply polls the STATUS register looking for Change Word Ready to go true. DBBOUT is then read and loaded into the buffer. Now let's move on to an application which combines both the foreground and background concepts.

## Combination I/O Device

The final UPI application was designed especially to add additional serial and parallel I/O ports to the iSBC 80/30. This UPI simulates a full-duplex UART (Universal Asynchronous Receiver/Transmitter) combined with an 8-bit parallel I/O port. Features of the UART include: software selectable baud rates (110, 300, 600, or 1200 baud), double buffering for both the transmitter and receiver, and receiver testing for false start bit, framing, and overrun errors. For parallel I/O, one 8-bit port is programmable for either input or output. The output port is statically latched and the input port is sampled.

# APPLICATIONS



**Figure 21A-J. FIFO Operation Example**

Figure 22 shows the interface of this combination I/O device to the dedicated UPI socket on the iSBC 80/30. The only external requirement is a 76.8 kHz source which serves as the baud rate standard. The internal baud rates are generated as multiples of this external clock. This clock is obtained from one of the 8253 counters. Otherwise, an RS-232 driver and receiver already available for UPI use in serial I/O applications. Sockets are also provided for termination of the parallel port.

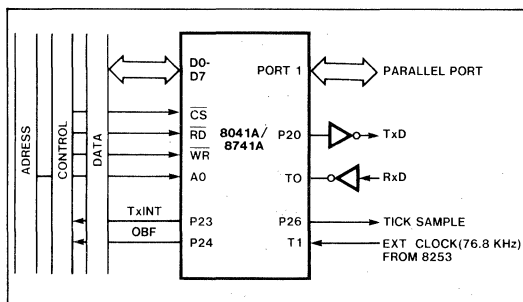


Figure 22. Combination I/O Device

There are three commands for this application. Their format is shown in Figure 23. The CONFIGURE command specifies the serial baud rate and the parallel I/O direction. Normally this command is issued once during system initialization. The I/O command causes a parallel I/O operation to be performed. If the parallel port direction is out, the UPI expects the data byte immediately following an I/O command to be data for the output port. If the port is in the input direction, an I/O command causes the port to be read and the data placed in DBBOUT. The RESET ERROR command resets the serial receiver error bits in the STATUS register.

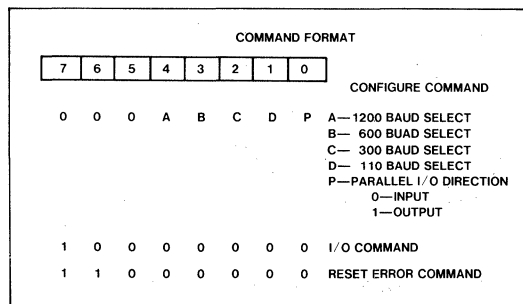


Figure 23. Combination I/O Command Format

The STATUS register format is shown in Figure 24. Looking at each bit, BIT 0 (OBF) is the DATA AVAILABLE flag. It is set whenever the UPI places data into DBBOUT. Since the data may come from

either the receiver or the parallel input port, the F<sub>0</sub> and F<sub>1</sub> flags (BITS 2 and 3) code the source. Thus, when the master finds OBF set, it must decode F<sub>0</sub> and F<sub>1</sub> to determine the source.

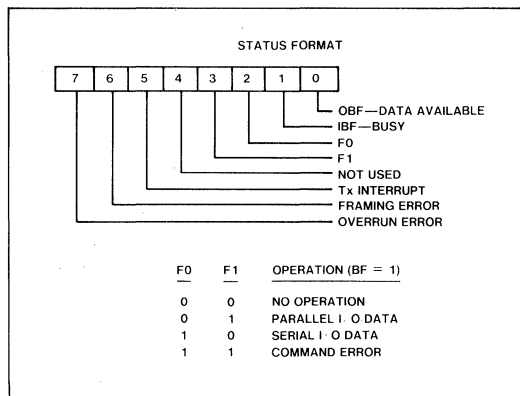


Figure 24. STATUS Register Format

BIT 1 (IBF) functions as a busy bit. When IBF is set, no writes to DBBIN are allowed. BIT 5 is the TxINT (Transmitter Interrupt) bit. It is asserted whenever the transmitter buffer register is empty. The master uses this bit to determine when the transmitter is ready to accept a data character.

BITS 6 and 7 are receiver error flags. The framing error flag, BIT 6, is set whenever a character is received with an invalid stop bit. BIT 7, overrun error, is set if a character is received before the master has read a previous character. If an overrun occurs, the previous character is overwritten and lost. Once an error occurs, the error flag remains set until reset by a RESET ERROR command. A set error flag does not inhibit receiver operation however.

Figure 25 shows the port pin definition for this application. PORT 1 is the parallel I/O port. The UART uses PORT 2 and the Test inputs. P<sub>20</sub> is the transmitter data out pin. It is set for a mark and reset for a space. P<sub>23</sub> is a transmitter interrupt output. This pin has the same timing as the TxINT bit in the STATUS register. It is normally used in interrupt-driven systems to interrupt the master processor when the transmitter is ready to accept a new data character.

The OBF flag is brought out on P<sub>24</sub> as a master interrupt when data is available in DBBOUT. P<sub>26</sub> is a diagnostic pin which pulses at four times the selected baud rate. (More about this pin later.) The receiver data input uses the TEST 0 input. One of the PORT 2 pins could have been used, however, the

PORT PIN DEFINITION		
PORT	BIT	FUNCTION
1	0-7	PARALLEL I/O
2	0	Tx Data
	1	NOT USED
	2	NOT USED
	3	Tx INTERRUPT
	4	OBF INTERRUPT
	5	NOT USED
	6	NOT USED (TICK SAMPLE)
	7	NOT USED
T0		Rx DATA
T1		EXTERNAL CLOCK (76.8 kHz)

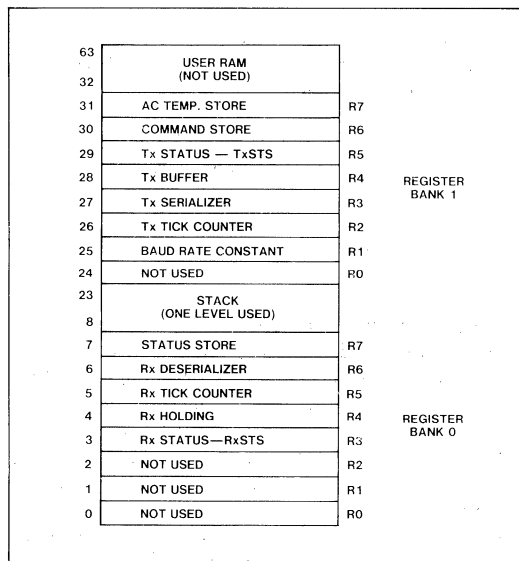
**Figure 25. Combination I/O Port Definition**

software can test the TEST 0 in one instruction without first reading a port.

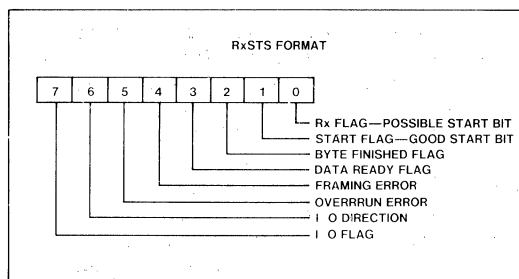
The TEST 1 input is the baud rate external source. The UART divides this input to determine the timing needed for the selected baud rate. The input is a non-synchronous 76.8 kHz source.

Internally, when the CONFIGURE command is received and the selected baud rate is determined, the internal timer/counter is loaded with a baud rate constant and started in the event counter mode. Timer/counter interrupts are then enabled. The baud rate constant is selected to provide a counter interrupt at four times the desired baud rate. At each interrupt, both the transmitter and receiver are handled. Between interrupts, any new commands and data are recognized and executed.

As a prelude to discussing the flow charts, Figure 26 shows the register definition. Register Bank 0 serves the UART receiver and parallel I/O while Register Bank 1 handles the UART transmitter and commands. Looking at RB0 first, R<sub>3</sub> is the receiver status register, RxSTS. Reflected in the bits of this register is the current receiver status in sequential order. Figure 27 shows this bit definition. BIT 0 is the Rx flag. It is set whenever a possible start bit is received. BIT 1 signifies that the start bit is good and character construction should begin with the next received bit. BIT 1 is the Good Start flag. BIT 2 is the Byte Finished flag. When all data bits of a character are received, this flag is set. When all the bits, data and stop bits are received, the assembled character is loaded into the holding register (R<sub>4</sub> in Figure 27) BIT 3, the Data Ready flag, is set. The foreground routine which looks for commands and data continuously, looks at this bit to determine when the receiver has received a character. BITS 4 and 5 signify any error conditions for a particular character.



**Figure 26. Combination I/O Register Map**



**Figure 27. RxSTS Register**

The parallel I/O port software uses BITS 6 and 7. BIT 6 codes the I/O direction specified by the last CONFIGURE command. BIT 7 is set whenever an I/O command is received. The foreground routine tests this bit to determine when an I/O operation has been requested by the master.

As was mentioned, R<sub>4</sub> is the receiver holding register. Assembled characters are held in this register until the foreground routine finds DBBOUT free, at which time the data is transferred from R<sub>4</sub> to DBBOUT. R<sub>5</sub> is the receiver tick counter. Recall that counter interrupts occur at four times the baud rate. Therefore, once a start bit is found, the receiver only needs to look at the data every four interrupts or tick counts. R<sub>5</sub> holds the current tick count.

R<sub>6</sub> is the receiver de-serializing register. Data characters are assembled in this register. R<sub>6</sub> is preset to 80H when a good start bit is received. As each bit is



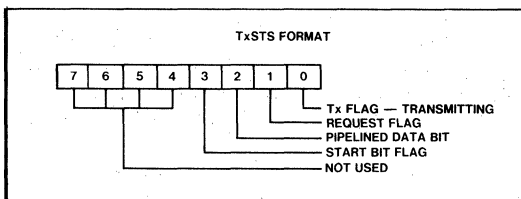
sampled every four timer ticks, they are rotated into the leftmost bit of R<sub>6</sub>. The software knows the character assembly is complete when the original preset bit rotates into the carry.

An image of the upper 4 bits of the STATUS register is stored in R<sub>7</sub>. These bits are the TxINT, Framing and Overrun bits. This image is needed since the UPI may load the upper 4 STATUS register bits from its accumulator; however, it cannot read STATUS directly.

In Register Bank 1 (Figure 26), R<sub>1</sub> holds the baud rate constant which is found from decoding the baud rate select bits of the CONFIGURE command. The counter is reloaded with this constant every timer tick. Like the receiver, the transmitter only needs to update the transmitter output every four ticks. R<sub>2</sub> holds the transmitter tick count. The value of R<sub>2</sub> determines which portion of the data is being transmitted; start bit, data bits, or stop bit. The transmit serializer is R<sub>3</sub>. R<sub>3</sub> holds the data character as each character bit is transmitted.

R<sub>4</sub> is the transmitter holding register. It provides the double buffering for the transmitter. While transmitting one character, it is possible to load the next character into R<sub>4</sub> via DBBIN. The TxINT bit in STATUS and pin on PORT 2 reflect the "fullness" of R<sub>4</sub>. If the holding register is empty, the interrupt bit and pin are set. They are reset when the master writes a new data byte for the transmitter into DBBIN. The transmitter status register (TxSTS) is R<sub>5</sub>. Like RxSTS, TxSTS contains flag bits which indicate the current state of the transmitter. This flag bit format is shown in Figure 28.

TxSTS BIT 0 is the Tx flag. It is set whenever the transmitter is transmitting a character. It is set from the beginning of the start bit until the end of the stop bit. BIT 1 is the Tx request flag. This bit is set by the foreground routine when it transfers a new character from DBBIN to the Tx holding register, R<sub>4</sub>. The transmitter software uses this flag to tell if new data is available. It is reset when the transmitter transfers the character from the holding register to the serializer.



**Figure 28. TxSTS Register**

BIT 2 is the pipelined Tx data bit. The transmitter uses a pipelining technique which sets up the next output level in BIT 2 after processing the current timer tick. The output level is always changed at the same point after a timer tick interrupt. This technique ensures that no bit timing distortion results from different length processing paths through the receiver and transmitter routines.

BIT 3 of TxSTS is the Start Bit flag. It is set by the transmitter when the start bit space is set up in the pipelined data bit. This allows the transmitter to differentiate between the start bit and the data bits on following timer ticks.

The flow charts for this application are shown in Figures 29A-F. At reset, the INIT routine is executed which initializes the registers and port pins. After initialization, IBF and OBF are tested in MNLOOP. These flags are tested continually in this loop. If IBF is set, F<sub>1</sub> is tested for command or data and execution is transferred to the appropriate routine (CMD or DATA). If IBF=0, OBF is checked. If OBF=0 (DBBOUT is free), the Rx data ready and I/O flags in RxSTS are tested. If Rx data ready is set, the received data is retrieved from the Rx holding register and transferred to DBBOUT. Any error flags associated with that data are also transferred to STATUS. If the I/O flag is set and the I/O direction is input, PORT 1 is read and the data transferred to DBBOUT. In either case, F<sub>0</sub> and F<sub>1</sub> are set to indicate the data source.

If IBF is set by a command write to DBBIN, CMD reads the command and decodes the desired operation. If an I/O operation is specified, the I/O flag is set to indicate to the MNLOOP and DATA routines that an I/O operation is to be performed. If the command is a CONFIGURE command, the constant for the selected baud rate is loaded into both Baud Rate Constant register and the timer/counter. The timer/counter is started in the event counter mode and timer/counter interrupts are enabled. In addition, the I/O port is initialized to all 1's if the I/O direction bit specifies an input port. If the command is a RESET ERROR command, the two error flags in STATUS are cleared.

If the IBF flag is set by a data write, the DATA routine reads DBBIN and places the data in the appropriate place. If the I/O flag is set, the data is for the output port so the port is loaded. If the I/O flag is reset, the data is for the UART transmitter. Data for the transmitter resets the TxINT bit and pin plus sets the Tx request flag in TxSTS. The data is transferred to the Tx holding register, R<sub>4</sub>.

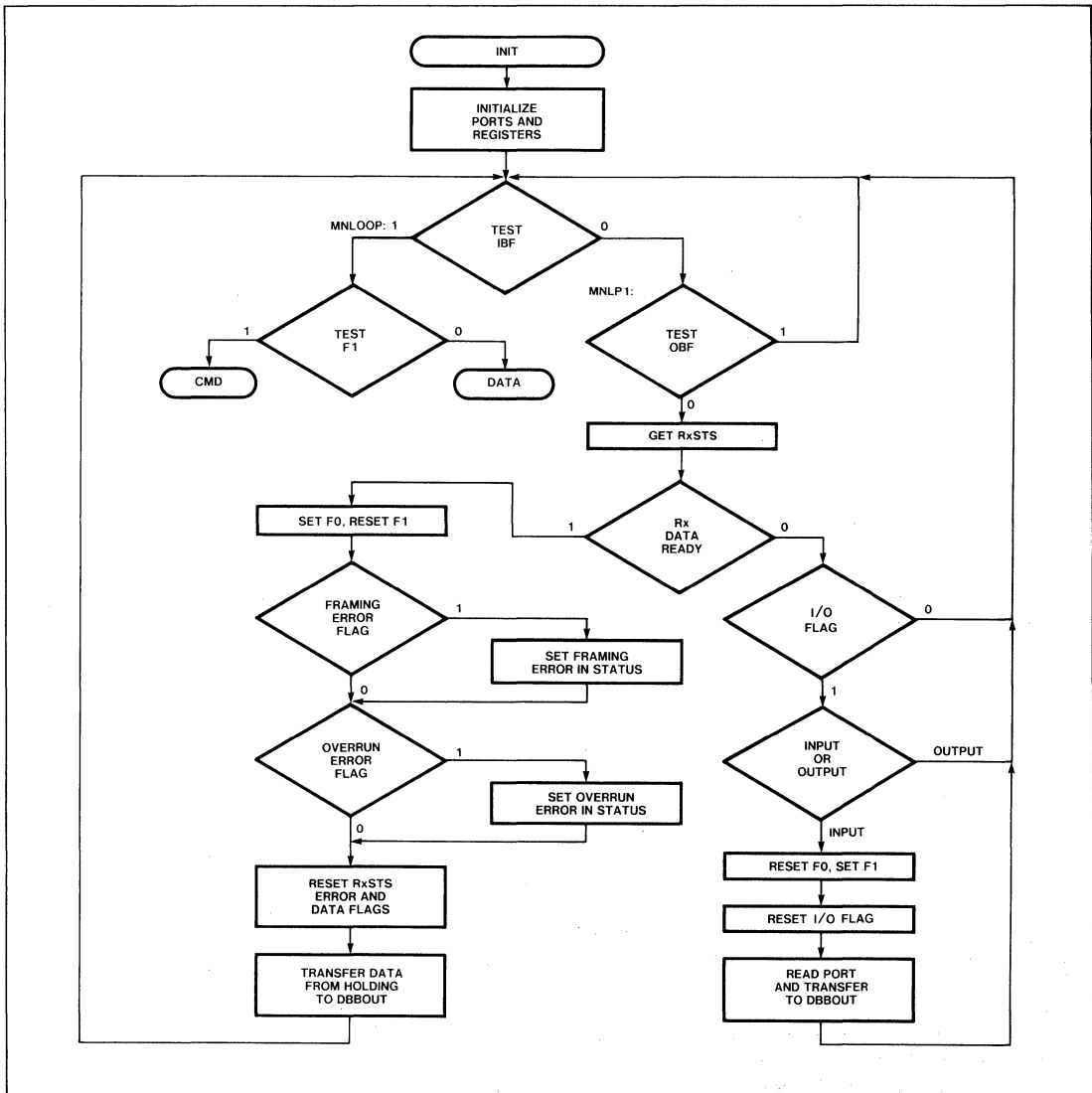


Figure 29A. INIT Flow Chart

Once a CONFIGURE command is received and the counter started, timer/counter interrupts start occurring at four times the selected baud rate. These interrupts cause a vector to the TIMINT routine, Figure 29D. A 76.8 kHz counter input provides a 13.02  $\mu$ s counter resolution. Since it requires several UPI instruction cycles to reload the counter, the counter is set to two counts less than the desired baud rate and the counter is reloaded in TIMINT synchronous with the second low-going transition after the interrupt. Once the counter is reloaded, an output port (P<sub>26</sub>) is toggled to give an external indi-

cation of internal counter interval. This is a helpful diagnostic feature. After the tick sample output, the pipelined transmitter data in TxSTS is output to the TxD pin. Although this occurs every timer tick, the pipelined data is changed only every fourth tick.

The receiver is now handled, Figure 29E. The Rx flag in RxSTS is examined to see if the receiver is currently in the process of receiving a character. If it is not, the RxD input is tested for a space condition which might indicate a possible start bit. If the input is a mark, no start bit is possible and execution

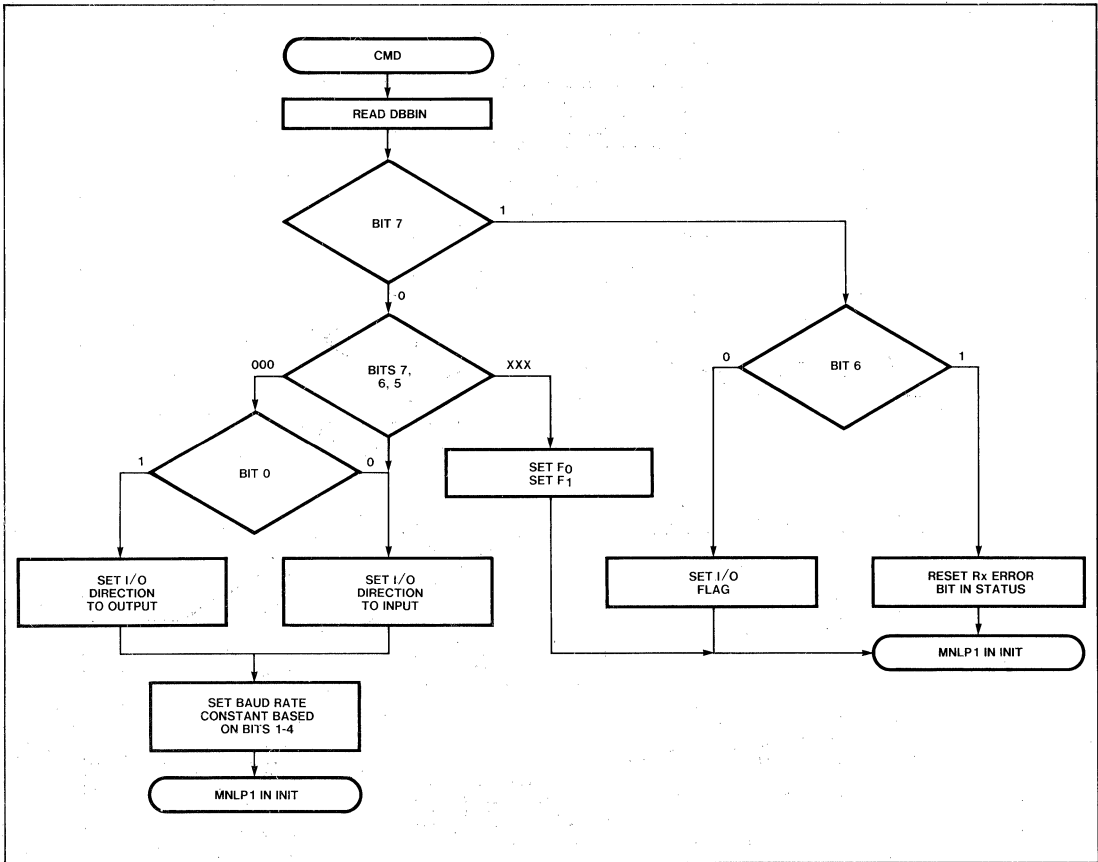


Figure 29B. CMD Flow Chart

branches to the transmitter flow, XMIT. If the input is a space, the Rx flag is set before proceeding with XMIT.

If the Rx flag is found set when entering RCV, the receiver is in the process of receiving a character. If so, the start bit flag is then tested to determine if a good start bit was received. The Rx tick counter is initialized to 4 and the Rx deserializer is set to 80H. A mark indicates a bad start bit; the Rx flag is reset to abort the reception.

If the start bit flag is set, the program is somewhere in the middle of the received character. Since the data should be sampled every fourth timer tick, the tick counter is decremented and tested for zero. If non-zero no sample is needed and execution continues with XMIT. If zero, the tick counter is reset to four. Now the byte finished flag is tested to determine if the data sample is a data or stop bit. If reset, the sample is a data bit. The sample is done and the new bit rotated into the Rx deserializer. If this rotate

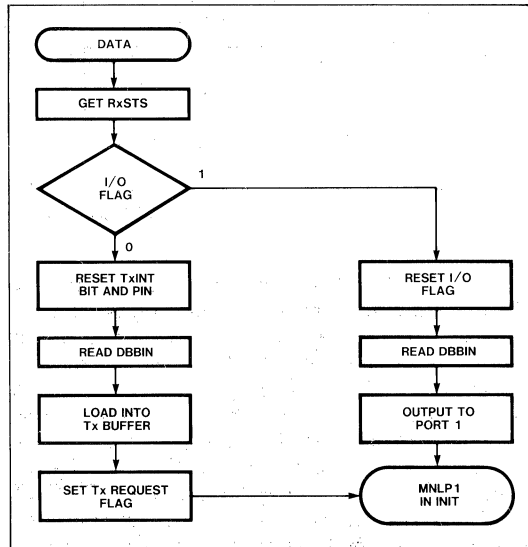


Figure 29C. Data Flow Chart

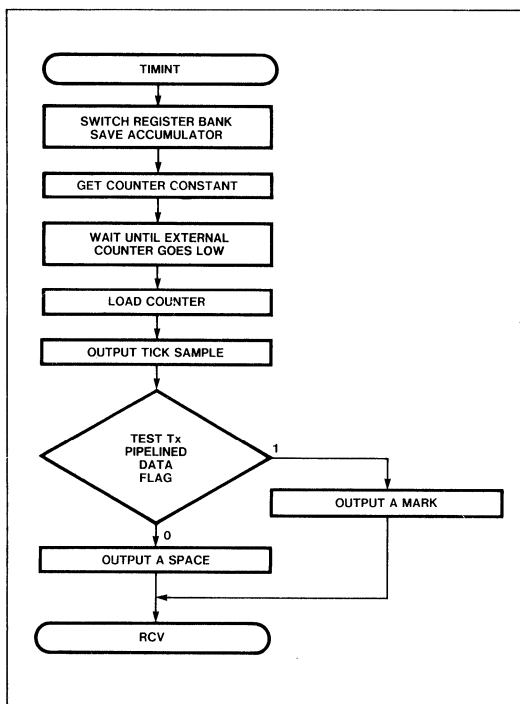


Figure 29D. TIMINT Flow Chart

sets the carry, that data bit was the last so the byte finished flag is set. If the carry is reset, the data bit is not the last so execution simply continues with XMIT.

Had the byte finished flag been set, this sample is for the stop bit. The RxD input is tested and if a space, the framing error flag is set. Otherwise, it is reset. Next, the Rx data ready flag is tested. If it is set, the master has not read the previous character so the overrun error flag is set. Then the Rx data ready flag is set and the received data character is transferred into the Rx holding register. The Rx, start bit, and byte finished flags are reset to get ready for the next character.

Execution of the transmitter routine, XMIT, follows the receiver, Figure 29F. The transmitter starts by checking the start bit flag in TxSTS. Recall that the actual transmit data is output at the beginning of the timer routine. The start bit flag indicates whether the current timer tick interrupt started the start bit. If it is set, the pipelined data output earlier in the routine was the start of the start bit so the flag is reset and the Tx tick counter is initialized. Nothing else is done this timer tick so the routine returns to the foreground.

If the start bit flag is reset, the Tx tick counter is incremented and tested. The test is performed modulo 4. If the counter mod 4 is not zero, it has not been four ticks since the transmitter was handled last so the routine simply returns. If the counter mod 4 is zero, it is time to handle the transmitter and the Tx flag is tested.

The Tx flag indicates whether the transmitter is active. If the transmitter is inactive, no character is currently being transmitted so the Tx request flag is tested to see if a new character is waiting in the Tx buffer. If no character is waiting (Tx request flag=0), the Tx interrupt pin and bit are set before returning to the foreground. If there is a character waiting, it is retrieved from the buffer and placed in the Tx serializer. The Tx request flag is reset while the Tx and start bit flags are set. A space is placed in the Tx pipelined data bit so a start bit will be output on the next tick. Since the Tx buffer is now empty, the Tx interrupt bit and pin are set to indicate the availability of the buffer to the master. The routine then returns to the foreground.

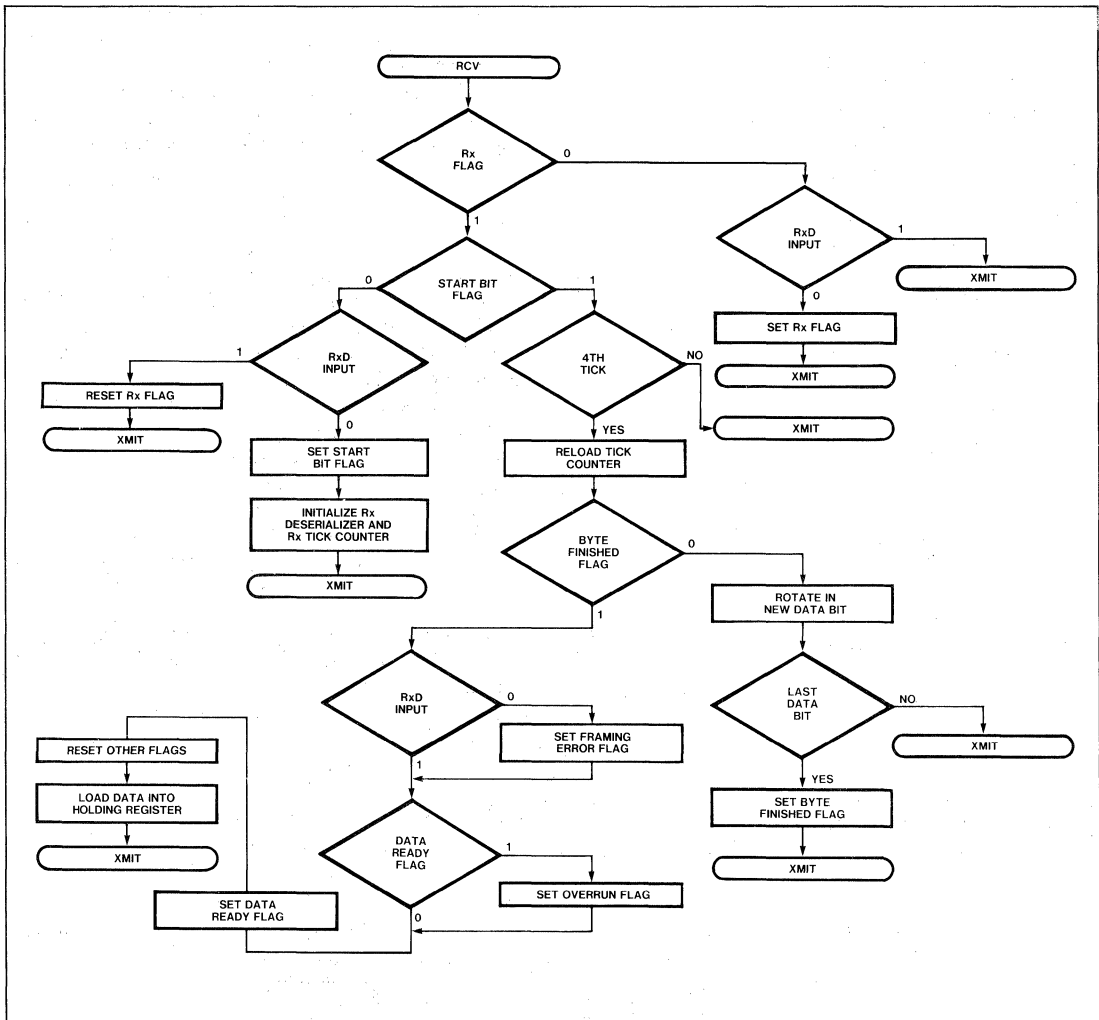
If the tick counter mod 4 is zero and the Tx flag indicates the transmitter is in the middle of a character, the tick counter is checked to see what transmitter operation is needed. If the counter is 28H (40D), all data bits plus the stop bits are complete. The character is therefore done and the Tx flag is reset. If the counter is 24H (36D), the data bits are complete and the next output should be a mark for the stop bit so a mark is loaded into the Tx pipelined data bit.

If neither of the above conditions are met for the counter, the transmitter is some place in the data field, so the next data bit is rotated out of the Tx serializer into the pipelined data bit. The next tick outputs this bit.

At this point the program execution is returned to the foreground.

That completes the discussion of the combination I/O device flow charts. The UPI software listing is shown in Appendix C1. Appendix C2 is example 8085A driver software.

Several observations concerning the drivers are appropriate. Notice that since the receiver and input port of the UPI use the OBF flag and interrupt output, the interrupt and flag are cleared when the master reads DBBOUT. This is not true for the transmitter. There is always some time after a master write of new transmitter data before the transmitter bit and pin are cleared. Thus in an interrupt-driven system, edge-sensitive interrupts should be



**Figure 29E. RCV Flow Chart**

used. For polled-systems, the software must wait after writing new data for IBF=0 before re-examining the Tx interrupt flag in STATUS.

Notice that this application uses none of the user data memory above Register Bank 1 and only 361 bytes of program memory. This leaves the door open for many improvements. Improvements that come to mind are increased buffering of the transmit or received data, modem control pins, and parallel port handshaking inputs.

This completes our discussion of specific UPI applications. Before concluding, let's look briefly at two debug techniques used during the development of

these applications that you might find useful in your own designs.

## DEBUG TECHNIQUES

Since the UPI is essentially a single-chip microcomputer, the classical data, address, and control buses are not available to the outside world during normal operation. This fact normally makes debugging a UPI design difficult; however, certain "tricks" can be included in the UPI software to ease this task.

If a UPI is handling multiple tasks, it is usually easier to code and debug each task individually. This is fairly standard procedure. Since each task usually utilizes only a subset of the total number of I/O pins,

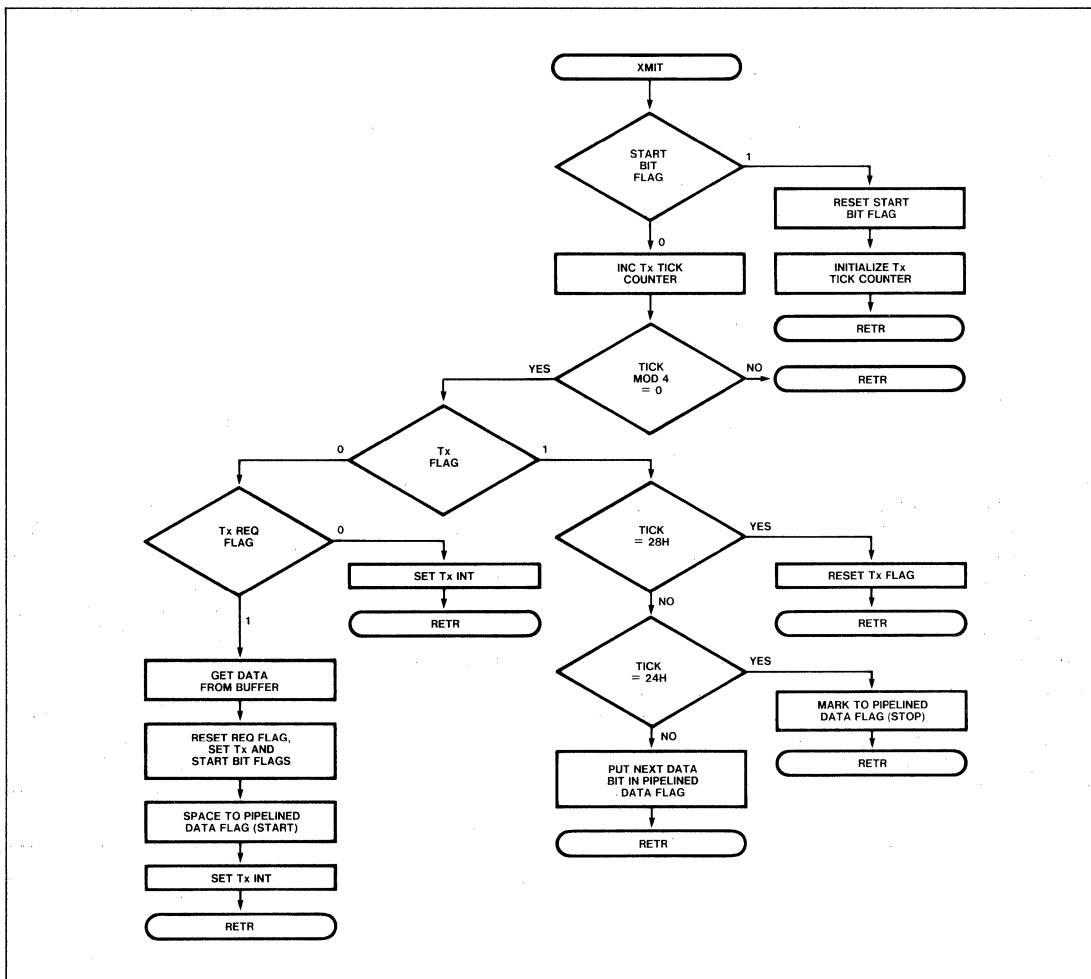


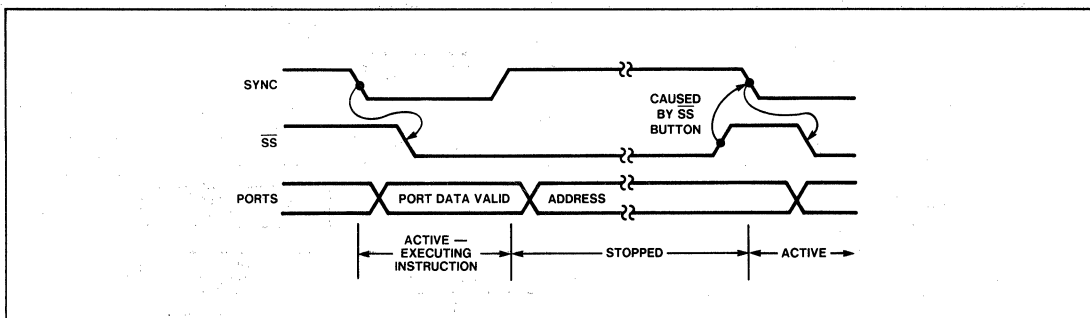
Figure 29F. XMIT Flow Chart

coding only one task leaves some I/O pins free. Port output instructions can then be added in the task code being debugged which toggle these unused pins to determine which section of task code is being executed at any particular time. The task can also be made to "wait" at various points by using an extra pin as an input and adding code to loop until a particular input condition is met.

One example of using an extra pin as an output is included in the combination serial/parallel device code. During initial development the receiver was not receiving characters correctly. Since this could be caused by incorrect sampling, three lines of code were added to toggle BIT 6 of PORT 2 at each tick of the sample clock. This code is at lines 184 and 185 of the listing. Thus by looking at the location of the tick

sample pulse with respect to the received bit, the UPI sampling interval can be observed. The tick sample time was incorrect and the code was modified accordingly. Similar techniques could be applied at other locations in the program.

The EPROM version of the UPI (8741A) also contains another feature to aid in debug: the capability to single step thru a program. The user may step thru the program instruction-by-instruction. The address of the next instruction to be fetched is available on PORT 1 and the lower 2 bits of PORT 2. Figure 30 shows the timing used in the discussion below. When the single step input, SS, is brought low, the internal processor responds by stopping during the fetch portion of the next instruction. This action is acknowledged by the processor raising the SYNC



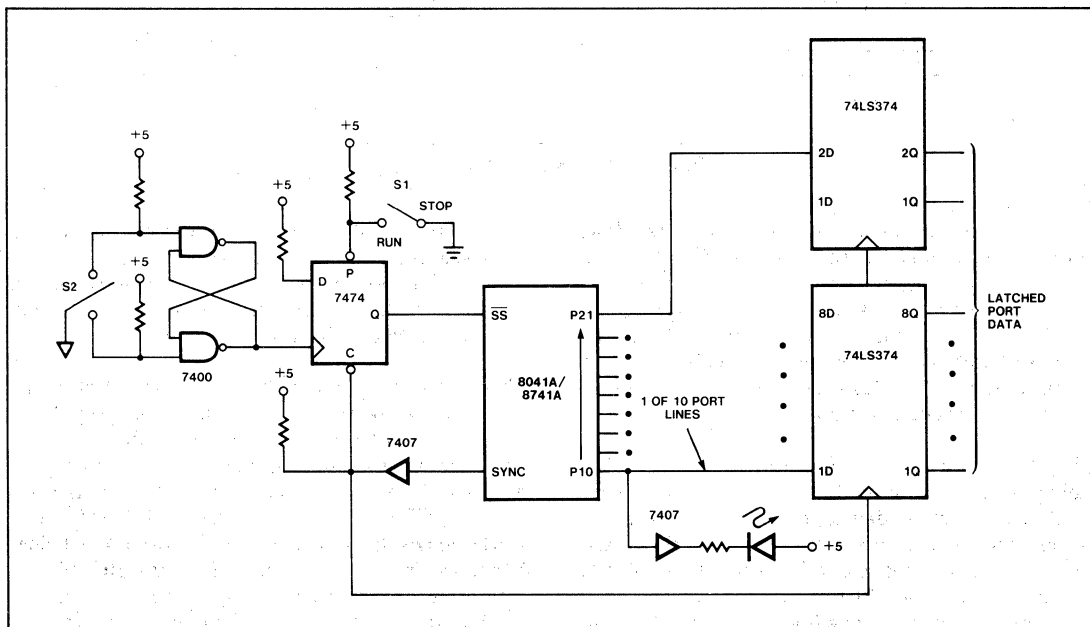
**Figure 30. Single Step Timing**

output. The address of the instruction to be fetched is then placed on the port pins. This state may be held indefinitely. To step to the next instruction,  $\overline{SS}$  is raised high, which causes SYNC to go low, which is then used to return  $\overline{SS}$  low. This allows the processor to advance to the next instruction. If  $\overline{SS}$  is left high, the processor continues to execute at normal speed until  $\overline{SS}$  goes low.

To preserve port functionality, port data is valid while SYNC is low. Figure 31 shows the external circuitry required to implement single step while preserving port functionality.  $S_1$  is the RUN/STOP switch. When in the RUN position, the 7474 is held preset so  $\overline{SS}$  is high and the UPI executes normally. When switched to STOP, the preset is removed and

the next low-going transition of SYNC causes the 7474 to clear, lowering  $\overline{SS}$ . While sync is low, the port data is valid and the current instruction is executing. Low SYNC is also used to enable the tri-state buffers when the ports are used as inputs. When execution is complete, SYNC goes high. This transition latches the valid port data in the 74LS374s. SYNC going high also signifies that the address of the next instruction will appear on the port pins. This state can be held indefinitely with the address data displayed on the LEDs.

When the  $S_2$  is depressed, the 7474 is set which causes  $\overline{SS}$  to go high. This allows the processor to fetch and execute the instruction whose address was displayed. SYNC going low during execution, clears



**Figure 31. Single Step External Circuitry**

the 7474 lowering  $\overline{SS}$ . Thus the processor again stops when execution is complete and the next fetch is started.

All UPI functions continue to operate while single stepping (the processor is actually executing NOPs internally while stopped). Both IBF and timer/counter interrupts can be serviced. The only change is that the interval timer is prescaled on single stepped instructions and, of course, will not indicate the correct intervals in real time. The total number of instruction which would have been executed during a given interval is the same however.

The single step circuitry can be used to step through a complete program; however, this might be a time-consuming job if the program is long or if only a portion is to be examined. The circuitry could easily be modified to incorporate the output toggling technique to determine when to run and stop. If you would like to step thru a particular section of code,

an extra port pin could replace switch S<sub>1</sub>. Extra instructions would then be added to lower the port when entering the code section and raise the port when exiting the section. The program would then stop when that section of code is reached allowing it to be stepped through. At the end of the section, the program would execute at normal speed.

### CONCLUSION

Well, that's it. Machine readable (floppy disk or paper tape) source listings of UPI software for these applications are available in Insite, the Intel library of user-donated programs. Also available in Insite are the source listings for some of Intel's pre-programmed UPI products.

For information about Insite, write to:

Insite  
Intel Corp.  
3065 Bowers Ave.  
Santa Clara, Ca 95051



---

## APPENDIX A

# APPLICATIONS

:F1:ASM48 :F3:LED PRINT(:LP:) NOOBJECT

ISIS-II MCS-48/UPI-41 MACRO ASSEMBLER, V3.0

PAGE 1

```
LOC  OBJ      LINE      SOURCE STATEMENT
1  *MOD41A
2  ;
3  ;          *****
4  ;          *      UPI-41A 8-DIGIT LED DISPLAY CONTROLLER      *
5  ;          *****
6  ;
7  ;
8  ; THIS PROGRAM USES THE UPI-41A AS A LED DISPLAY CONTROLLER
9  ; WHICH SCANS AND REFRESHES EIGHT SEVEN-SEGMENT LED DISPLAYS.
10 ; THE CHARACTERS ARE DEFINED BY INPUT FROM A MASTER CPU IN THE
11 ; FORM OF ONE EIGHT BIT WORD PER DIGIT-CHARACTER SELECTION.
12 ;
13 ;
14 ;
15 ; *****
16 ;
17 ; REGISTER DEFINITIONS:
18 ;     REGISTER          RB1          RBO
19 ;     -----          ---          ---
20 ;     R0                DISPLAY MAP POINTER          NOT USED
21 ;     R1                NOT USED                    NOT USED
22 ;     R2                DATA WORD AND CHARACTER STORAGE NOT USED
23 ;     R3                DIGIT COUNTER              NOT USED
24 ;     R4                NOT USED                    NOT USED
25 ;     R5                NOT USED                    NOT USED
26 ;     R6                NOT USED                    NOT USED
27 ;     R7                ACCUMULATOR STORAGE       NOT USED
28 ; *****
29 ;
30 ; PORT PIN DEFINITIONS:
31 ;     PIN              PORT 1 FUNCTION          PORT 2 FUNCTION
32 ;     ---              -----          -----
33 ;     PO-7            SEGMENT DRIVER CONTROL  DIGIT DRIVER CONTROL
34 ;
35 *EJECT
```

# APPLICATIONS

LOC OBJ

LINE SOURCE STATEMENT

```

36 ; *****
37 ; DISPLAY DATA WORD BIT DEFINITION:
38 ;     BIT                FUNCTION
39 ;     ---                -
40 ;     0-4                CHARACTER SELECT
41 ;     5-7                DIGIT SELECT
42 ;
43 ; CHARACTER SELECT:
44 ;     D4  D3  D2  D1  D0  CHARACTER
45 ;     0   0   0   0   0   0
46 ;     0   0   0   0   1   1
47 ;     0   0   0   1   0   2
48 ;     0   0   0   1   1   3
49 ;     0   0   1   0   0   4
50 ;     0   0   1   0   1   5
51 ;     0   0   1   1   0   6
52 ;     0   0   1   1   1   7
53 ;     0   1   0   0   0   8
54 ;     0   1   0   0   1   9
55 ;     0   1   0   1   0   A
56 ;     0   1   0   1   1   B
57 ;     0   1   1   0   0   C
58 ;     0   1   1   0   1   D
59 ;     0   1   1   1   0   E
60 ;     0   1   1   1   1   F
61 ;     1   0   0   0   0   .
62 ;     1   0   0   0   1   G
63 ;     1   0   0   1   0   H
64 ;     1   0   0   1   1   I
65 ;     1   0   1   0   0   J
66 ;     1   0   1   0   1   L
67 ;     1   0   1   1   0   N
68 ;     1   0   1   1   1   O
69 ;     1   1   0   0   0   P
70 ;     1   1   0   0   1   R
71 ;     1   1   0   1   0   T
72 ;     1   1   0   1   1   U
73 ;     1   1   1   0   0   Y
74 ;     1   1   1   0   1   -
75 ;     1   1   1   1   0   /
76 ;     1   1   1   1   1   "BLANK"
77 ;
78 ; DIGIT SELECT:
79 ;     D7  D6  D5  DIGIT NUMBER
80 ;     0   0   0           1
81 ;     0   0   1           2
82 ;     0   1   0           3
83 ;     0   1   1           4
84 ;     1   0   0           5
85 ;     1   0   1           6
86 ;     1   1   0           7
87 ;     1   1   1           8
88 ; *****
89 ;EJECT
    
```

# APPLICATIONS

```

LOC  OBJ          LINE          SOURCE STATEMENT
          90 ;*****
          91 ;          EQUATES
          92 ;THE FOLLOWING CODE DESIGNATES "TIME" AS A VARIABLE.THIS
          93 ;ADJUSTS THE AMOUNT OF CYCLES THE TIMER COUNTS BEFORE
          94 ;A TIMER INTERRUPT OCCURS AND REFRESHES THE DISPLAY APPROXIMATELY
          95 ;30 TIMES PER SECOND.
FFF1          96 TIME      EQU      -OFH          ;TIMER VALUE 2.5MSEC
          97 ;*****
          98 ;          INTERRUPT BRANCHING
          99 ;THIS PORTION OF MEMORY IS DEDICATED FOR USE OF RESET AND
        100 ;INTERRUPT BRANCHING WHEN THE INTERRUPTS ARE ENABLED THE
        101 ;CODE AT THE FOLLOWING DESIGNATED SPOTS ARE EXECUTED WHEN A
        102 ;RESET OR A INTERRUPT OCCURS.
0000          103          ORG      0
        0000 0409          104          JMP      START          ;RESET
        0002 00          105          NOP
        0003 0436          106          JMP      INPUT          ;IBF INTERRUPT
        0005 00          107          NOP
        0006 00          108          NOP
        0007 041D          109          JMP      DISPLA          ;TIMER INTERRUPT
        110 ;*****
        111 ;          INITIALIZATION
        112 ;THE FOLLOWING CODE SETS UP THE UPI-41 AND DISPLAY HARDWARE
        113 ;INTO OPERATIONAL FORMAT. THE DISPLAY IS TURNED OFF, THE DISPLAY
        114 ;MAP IS FILLED WITH "BLANK" CHARACTERS, THE TIMER SET AND THE
        115 ;INTERRUPTS ARE ENABLED.
        116 ;
        0009 D5          117 START:  SEL      RB1
        000A 8A0B          118          ORL      P2,#0BH          ;TURN DIGIT DRIVERS OFF
        000C 8838          119          MOV      RO,#3BH          ;DISPLAY MAP POINTER,BOTTOM OF DISPLAY MAP
        000E 23FF          120 BLKMAP: MOV      A,#0FFH          ;FF="BLANK"
        0010 A0          121          MOV      @RO,A          ;BLANK TO DISPLAY MAP
        0011 18          122          INC      RO          ;INCREMENT DISPLAY MAP POINTER
        0012 FB          123          MOV      A,RO          ;DISPLAY MAP POINTER TO ACCUMULATOR
        0013 B20E          124          JBS      BLKMAP          ;BLANK DISPLAY MAP TILL FILLED
        0015 B800          125          MOV      R3,#00H          ;SET DIGIT_COUNTER TO 0
        0017 23F1          126          MOV      A,#TIME          ;TIMER VALUE
        0019 62          127          MOV      T,A          ;LOAD TIMER
        001A 55          128          STRT      T          ;START TIMER
        001B 25          129          EN      TCNTI          ;ENABLE TIMER INTERRUPT
        001C 05          130          EN      I          ;ENABLE IBF INTERRUPT
        131 ;*****
        132 ;          USER PROGRAM
        133 ;A USERS PROGRAM WOULD INITIALIZE AT THIS POINT. THE FOLLOWING
        134 ;CODE IS UND CONCLUDED WITH
        135 ;SYNC CHARACTERS (0AAH). A CHECKSUM BYTE IMMEDIATELY PRECEEDS THE
        136 ;FINAL SYNC. WHEN READING, THE CONTROLLE*****
        137 *EJECT
    
```

# APPLICATIONS

LOC	OBJ	LINE	SOURCE STATEMENT
138			;*****
139			DISPLAY ROUTINE
140			; THIS PORTION OF THIS PROGRAM IS AN INTERRUPT ROUTINE WHICH IS
141			; ACTED UPON WHEN THE TIMER COUNT IS COMPLETED. THE ROUTINE UPDATES
142			; ONE DISPLAY DIGIT FROM THE DISPLAY MAP PER INTERRUPT SEQUENTIALLY,
143			; THUS EIGHT TIMER INTERRUPTS WILL HAVE REFRESHED THE ENTIRE DISPLAY.
144			; REGISTER BANK 1 IS SELECTED AND THE ACCUMULATOR IS SAVED UPON
145			; ENTERING THE ROUTINE. ONCE THE DISPLAY HAS BEEN REFRESHED THE TIMER
146			; IS RESET AND THE ACCUMULATOR AND PRE-INTERRUPT REGISTER BANK IS RESTORED.
147			;
001D	D5	148	DISPLA: SEL RB1 ; REGISTER BANK 1
001E	AF	149	MOV R7,A ; SAVE ACCUMULATOR
001F	8A0B	150	ORL P2,#0BH ; TURN DIGIT DRIVERS OFF
0021	FB	151	MOV A,R3 ; DIGIT COUNTER TO ACCUMULATOR
0022	433B	152	ORL A,#3BH ; "OR" TO GET DISPLAY MAP ADDRESS
0024	AB	153	MOV RO,A ; DISPLAY MAP POINTER
0025	F0	154	MOV A,@RO ; GET CHARACTER FROM DISPLAY MAP
0026	3F	155	OUTL P1,A ; OUTPUT CHARACTER TO SEGMENT DRIVERS
0027	FB	156	MOV A,R3 ; DIGIT COUNTER VALUE TO ACCUMULATOR
0028	3A	157	OUTL P2,A ; OUTPUT TO DIGIT DRIVERS
0029	1B	158	INC R3 ; INCREMENT DIGIT COUNTER
002A	D307	159	XRL A,#07H ; CHECK IF AT LAST DIGIT
002C	9630	160	JNZ SETIME ; RESET TIMER IN NOT LAST DIGIT
002E	8B00	161	MOV R3,#00H ; RESET DIGIT COUNTER
0030	23F1	162	SETIME: MOV A,#TIME ; TIMER VALUE
0032	62	163	MOV T,A ; LOAD TIMER
0033	55	164	STRT T ; START TIMER
0034	FF	165	MOV A,R7 ; RESTORE ACCUMULATOR
0035	93	166	RETR ; RETURN
167			;*****
168			SEJECT

# APPLICATIONS

LOC	OBJ	LINE	SOURCE STATEMENT
		169	;
		170	;*****
		171	; INPUT CHARACTER AND DIGIT ROUTINE
		172	; THIS PORTION OF THE PROGRAM IS AN INTERRUPT ROUTINE WHICH
		173	; IS ACTED UPON WHEN THE IDF BIT IS SET. THE ROUTINE GETS THE
		174	; DISPLAY DATA WORD FROM THE DBB AND DEFINES BOTH THE DIGIT AND
		175	; THE CHARACTER TO BE DISPLAYED. THIS IS DONE BY MEANS OF A
		176	; CHARACTER LOOP-UP TABLE AND A DISPLAY MAP FOR DIGIT AND CHARACTER
		177	; LOCATION. SPECIAL CONSIDERATION IS TAKEN FOR A DECIMAL POINT WHICH IS
		178	; SIMPLY ADDED TO THE EXISTING CHARACTER IN THE DISPLAY MAP REGISTER
		179	; BANK 1 IS SELECTED AND THE ACCUMULATOR IS SAVED UPON ENTERING
		180	; THE ROUTINE. ONCE THE DATA WORD HAS BEEN FULLY DEFINED THE ACCUMULATOR
		181	; AND THE PRE-INTERRUPT REGISTER BANK IS RESTORED.
		182	;
0036	D5	183	INPUT: SEL RB1 ;REGISTER BANK 1
0037	AF	184	MOV R7,A ;SAVE ACCUMULATOR
0038	22	185	IN A,DBB ;GET DATA
0039	AA	186	MOV R2,A ;SAVE DATA WORD
003A	47	187	SWAP A ;DEFINE DIGIT LOCATION
003B	77	188	RR A ;
003C	5307	189	ANL A,#07H ;
003E	433B	190	ORL A,#3BH ;
0040	AB	191	MOV RO,A ;DIGIT LOCATION IN DIGIT POINTER
0041	FA	192	MOV A,R2 ;SAVED DATA WORD TO ACCUMULATOR
0042	531F	193	ANL A,#1FH ;DEFINE CHARACTER LOOK-UP-TABLE LOC.
0044	E3	194	MOV3 A,@A ;GET CHARACTER
0045	AA	195	MOV R2,A ;SAVE CHARACTER
0046	D37F	196	XRL A,#7FH ;IS CHARACTER DECIMAL POINT
0048	C64E	197	JZ DPOINT ;
004A	FA	198	MOV A,R2 ;SAVED CHARACTER TO ACCUMULATOR
004B	A0	199	MOV @RO,A ;CHARACTER TO DISPLAY MAP
004C	0451	200	JMP RETURN ;
004E	FA	201	DPOINT: MOV A,R2 ;SAVED CHARACTER TO ACCUMULATOR
004F	50	202	ANL A,@RO ;"AND" WITH OLD CHARACTER
0050	A0	203	MOV @RO,A ;BACK TO DISPLAY MAP
0051	FF	204	RETURN: MOV A,R7 ;RESTORE ACCUMULATOR
0052	93	205	RETR ;
		206	;*****
		207	;\$EJECT

# APPLICATIONS

```

LOC  OBJ      LINE      SOURCE STATEMENT
208 ; *****
209 ;          LOOK-UP TABLE
210 ; THIS LOOK-UP TABLE ORIGINATES IN PAGE 3 OF THE UPI-41 PROGRAM
211 ; MEMORY. IT IS USED TO DEFINE THE CORRECT LEVEL OF EACH SEGMENT
212 ; AND DECIMAL POINT FOR A SELECTED CHARACTER FROM THE INPUT ROUTINE.
213 ; INVERSE LOGIC IS USED BECAUSE OF THE SPECIFIC DRIVER CIRCUITRY, THUS
214 ; A 1 ON A GIVEN SEGMENT MEANS IT IS OFF AND A 0 MEANS IT IS ON.
215 ;
216 ; *****SEGMENTS*****
0300  217          ORG      300H          ;DP 0 F E D C B A
0300  C0      218 CHO:   DB      0C0H          ; 1 1 0 0 0 0 0 0
0301  F9      219 CH1:  DB      0F9H          ; 1 1 1 1 1 0 0 1
0302  A4      220 CH2:  DB      0A4H          ; 1 0 1 0 0 1 0 0
0303  B0      221 CH3:  DB      0B0H          ; 1 0 1 1 0 0 0 0
0304  99      222 CH4:  DB      99H           ; 1 0 0 1 1 0 0 1
0305  92      223 CH5:  DB      92H           ; 1 0 0 1 0 0 1 0
0306  B2      224 CH6:  DB      B2H           ; 1 0 0 0 0 0 1 0
0307  FB      225 CH7:  DB      0FBH          ; 1 1 1 1 1 0 0 0
0308  B0      226 CH8:  DB      B0H           ; 1 0 0 0 0 0 0 0
0309  98      227 CH9:  DB      98H           ; 1 0 0 1 1 0 0 0
030A  BB      228 CHA:  DB      BBH           ; 1 0 0 0 1 0 0 0
030B  B3      229 CHB:  DB      B3H           ; 1 0 0 0 0 0 0 1
030C  C6      230 CHC:  DB      0C6H          ; 1 1 0 0 0 0 1 0
030D  A1      231 CHD:  DB      0A1H          ; 1 0 1 0 0 0 0 1
030E  B6      232 CHE:  DB      B6H           ; 1 0 0 0 0 1 1 0
030F  BE      233 CHF:  DB      BEH           ; 1 0 0 0 1 1 1 0
0310  7F      234 CHDP: DB      7FH           ; 0 1 1 1 1 1 1 1
0311  C2      235 CH9:  DB      0C2H          ; 1 1 0 0 0 0 1 0
0312  B9      236 CHH:  DB      B9H           ; 1 0 0 0 1 0 0 1
0313  FB      237 CHI:  DB      0FBH          ; 1 1 1 1 1 0 0 1
0314  E1      238 CHJ:  DB      0E1H          ; 1 1 1 0 0 0 0 1
0315  C7      239 CHL:  DB      0C7H          ; 1 1 0 0 0 1 1 1
0316  AB      240 CHN:  DB      0ABH          ; 1 0 1 0 1 0 1 1
0317  A3      241 CHO:  DB      0A3H          ; 1 0 1 0 0 0 1 1
0318  BC      242 CHF:  DB      B8H           ; 1 0 0 0 1 1 0 0
0319  AF      243 CHR:  DB      0AFH          ; 1 0 1 0 1 1 1 1
031A  B7      244 CHT:  DB      B7H           ; 1 0 0 0 0 1 1 1
031B  C1      245 CHU:  DB      0C1H          ; 1 1 0 0 0 0 0 1
031C  91      246 CHY:  DB      91H           ; 1 0 0 1 0 0 0 1
031D  BF      247 CHDASH: DB 0BFH          ; 1 0 1 1 1 1 1 1
031E  FD      248 CHAPOS: DB 0FDH          ; 1 1 1 1 1 1 0 1
031F  FF      249 BLANK: DB 0FFH          ; 1 1 1 1 1 1 1 1
250 ; *****
251          END

```

USER SYMBOLS

BLANK	031F	BLKMAP	000E	CHO	0300	CH1	0301	CH2	0302	CH3	0303	CH4	0304	CH5	0305
CH6	0306	CH7	0307	CH8	0308	CH9	0309	CHA	030A	CHAPOS	031E	CHB	030B	CHC	030C
CHD	030D	CHDASH	031D	CHDP	0310	CHE	030E	CHF	030F	CHG	0311	CHH	0312	CHI	0313
CHJ	0314	CHL	0315	CHN	0316	CHO	0317	CHP	0318	CHR	0319	CHT	031A	CHU	031B
CHY	031C	DISPLA	001D	DPOINT	004E	INPUT	0036	RETURN	0051	SETIME	0030	START	0009	TIME	FFF1

ASSEMBLY COMPLETE, NO ERRORS

# APPLICATIONS

!F1:ASM4B !F3: SENSOR NOOBJECT PRINT(,LP,)

ISIS-II MCS-4B/UPI-41 MACRO ASSEMBLER, V3.0

PAGE 1

```

LOC  OBJ      LINE      SOURCE STATEMENT
-----
1  $MOD41A
2  ; *****
3  ; *      UPI-41A SENSOR MATRIX CONTROLLER      *
4  ; *****
5  ;
6  ;      THIS PROGRAM USES THE UPI-41A AS A SENSOR MATRIX CONTROLLER.
7  ; IT HAS MONITORING CAPABILITIES OF UP TO 128 SENSORS.  THE COORDINATE
8  ; AND SENSOR STATUS OF EACH DETECTED CHANGE IS AVAILABLE TO THE MASTER
9  ; MICROPROCESSOR IN A SINGLE BYTE.  A 40XB FIFO QUEUE IS PROVIDED FOR
10 ; DATA BUFFERING.  BOTH HARDWARE OR POLLED INTERRUPT METHODS CAN BE USED
11 ; TO NOTIFY THE MASTER OF A DETECTED SENSOR CHANGE.
12 ;
13 ; *****
14 ;
15 ; REGISTER DEFINITIONS:
16 ;
17 ;      REGISTER      RBQ      RBI
18 ;      -----      ---      ---
19 ;      R0      MATRIX MAP POINTER      NOT USED
20 ;      R1      FIFO POINTER      NOT USED
21 ;      R2      SCAN ROW SELECT      NOT USED
22 ;      R3      COLUMN COUNTER      NOT USED
23 ;      R4      FIFO-IN      NOT USED
24 ;      R5      FIFO-OUT      NOT USED
25 ;      R6      CHANGE WORD      NOT USED
26 ;      R7      COMPARE      NOT USED
27 ; *****
28 ;
29 ; PORT PIN DEFINITIONS:
30 ;
31 ;      PIN      PORT 1 FUNCTION      PIN      PORT 2 FUNCTION
32 ;      ---      -----      ---      -----
33 ;      P0-7      COLUMN LINE INPUTS      P0-3      ROW SELECT OUTPUTS
34 ;      P4      FIFO NOT EMPTY INTERRUPT
35 ;      P5      OBF INTERRUPT
36 ;      P6-7      NOT USED
37 ;
38 ; *****
39 ;
40 $EJECT

```



# APPLICATIONS

ISIS-II MCS-4B/UPI-41 MACRO ASSEMBLER, V3.0

PAGE 2

```
LOC  OBJ      LINE      SOURCE STATEMENT
41 ; *****
42 ;
43 ; CHANGE WORD BIT DEFINITION:
44 ;
45 ;             BIT             FUNCTION
46 ;             ----            -
47 ;             D0-6           SENSOR COORDINATE
48 ;             D7             SENSOR STATUS
49 ;
50 ; *****
51 ;
52 ; STATUS REGISTER BIT DEFINITION:
53 ;
54 ;             BIT             FUNCTION
55 ;             ----            -
56 ;             D0             OBF
57 ;             D1-3           IBF, FO, F1 (NOT USED)
58 ;             D4             FIFO NOT EMPTY
59 ;             D5-7           USED DEFINED (NOT USED)
60 ;
61 ; *****
62 ;
63 ;             EQUATES
64 ;
65 ; THE FOLLOWING CODE DESIGNATES THREE VARIABLES; SCANTM, FIF0BA
66 ; AND FIF0TA. SCANTM ADJUSTS THE LENGTH OF A DELAY BETWEEN
67 ; SCANNING SWITCH. THIS SIMULATES DEBOUNCE FUNCTIONS. FIF0BA
68 ; IS THE BOTTOM ADDRESS OF THE FIFO. FIF0TA IS THE TOP ADDRESS
69 ; OF THE FIFO. THIS MAKES IT POSSIBLE TO HAVE A FIFO 3 TO 40
70 ; BYTES IN LENGTH.
71 ;
72 ; *****
73 ;
000F 74 SCANTM EQU 0FH           ; SCAN TIME ADJUST
000B 75 FIF0BA EQU 0BH         ; FIFO BOTTOM ADDRESS
002F 76 FIF0TA EQU 2FH         ; FIFO TOP ADDRESS
77
78 $EJECT
```

# APPLICATIONS

LOC	OBJ	LINE	SOURCE STATEMENT
		79	*****
		80	
		81	INITIALIZATION
		82	
		83	THE PROGRAM STARTS AT THE FOLLOWING CODE UPON RESET. WITHIN
		84	THIS INITIALIZATION SECTION THE REGISTERS THAT MAINTAIN THE MATRIX
		85	MAP, FIFO AND ROW SCANNING ARE SET UP. PORT 1 IS SET HIGH FOR USE
		86	AS AN INPUT PORT FOR THE COLUMN STATUS. BIT 4 OF STATUS REGISTER IS
		87	WRITTEN TO CONVEY A FIFO EMPTY CONDITION. THE INITIAL COLUMN STATUS
		88	OF ALL THE ROWS IN THE SENSOR MATRIX IS THEN READ INTO THE MATRIX
		89	MAP. ONCE THE MATRIX MAP IS FILLED THE OBF INTERRUPT (PORT 2-4) IS
		90	ENABLED.
		91	
		92	*****
		93	
0000		94	ORG 0
0000	BB3F	95	INITMX: MOV R0, #3FH ; MATRIX MAP POINTER REGISTER, TOP ADDRESS
0002	BA0F	96	MOV R2, #0FH ; SCAN ROW SELECT REGISTER, TOP ROW
0004	BC0B	97	MOV R4, #FIFOBA ; FIFO INPUT ADDRESS REGISTER, BOTTOM OF FIFO
0006	BD2F	98	MOV R5, #FIFOTA ; FIFO OUTPUT ADDRESS REGISTER, TOP OF FIFO
0008	B9FF	99	ORL P1, #0FFH ; INITIALIZE PORT 1 HIGH FOR INPUTS
000A	2300	100	MOV A, #00H ; INITIALIZE STATUS REGISTER, FIFO EMPTY
000C	90	101	MOV STS, A ; WRITE TO STATUS REGISTER, BITS 4-7
000D	FA	102	FILLMX: MOV A, R2 ; SCAN ROW SELECT TO ACCUMULATOR
000E	3A	103	OUTL P2, A ; OUTPUT SCAN ROW SELECT TO PORT 2
000F	09	104	IN A, P1 ; INPUT COLUMN STATUS PORT 1
0010	A0	105	MOV @R0, A ; LOAD MATRIX MAP WITH COLUMN STATUS
0011	FA	106	MOV A, R2 ; CHECK SCAN ROW SELECT REGISTER VALUE FOR 0
0012	C61B	107	JZ OBFINT ; IF 0 ENABLE OBF INTERRUPT
0014	CB	108	DEC R0 ; DECREMENT TO NEXT MATRIX MAP ADDRESS
0015	CA	109	DEC R2 ; DECREMENT TO SCAN NEXT ROW
0016	040D	110	JMP FILLMX ; FILL NEXT MATRIX MAP ADDRESS
001B	BA10	111	OBFINT: MOV R2, #10H ; BIT 4 HIGH IN ROW SCAN SELECT REGISTER
001A	FA	112	MOV A, R2 ; ROW SCAN SELECT VALUE TO ACCUMULATOR
001B	3A	113	OUTL P2, A ; INITIALIZE PORT 2, BIT 4 FOR "EN FLAGS"
001C	F5	114	EN FLAGS ; ENABLE OBF INTERRUPT PORT 2, BIT 4
		115	
		116	*EJECT

# APPLICATIONS

```

LOC  OBJ          LINE      SOURCE STATEMENT
117 ; *****
118 ;
119 ;              SCAN AND COMPARE
120 ;
121 ; THE FOLLOWING CODE IS THE SCAN AND COMPARE SECTION OF THE PROGRAM.
122 ; UPON ENTERING THIS SECTION A CHECK IS MADE TO SEE IF THE ENTIRE MATRIX
123 ; HAS BEEN SCANNED.  IF SO THE REGISTERS THAT MAINTAIN THE MATRIX MAP AND ROW
124 ; SCANNING ARE RESET TO THE BEGINNING OF THE SENSOR MATRIX.  IF THE ENTIRE
125 ; MATRIX HASNT BEEN SCANNED THE REGISTERS INCREMENT TO SCAN THE NEXT ROW.
126 ; FROM THIS POINT ON THE ROW SCAN SELECT REGISTER IS USED FOR TWO FUNCTIONS.
127 ; BITS 0-3 FOR SCANNING AND BITS 4 AND 5 FOR THE EXTERNAL INTERRUPTS.  THUSLY
128 ; ALL USAGE OF THE REGISTERS IS DONE BY LOGICALLY MASKING IT SO AS TO ONLY
129 ; AFFECT THE FUNCTION DESIRED.  ONCE THE REGISTERS ARE RESET, ONE ROW OF THE
130 ; SENSOR MATRIX IS SCANNED.  A DELAY IS EXECUTED TO ADJUST FOR SCAN TIME
131 ; (DEBOUNCE).  A BYTE OF COLUMN STATUS IS THEN READ INTO THE MATRIX MAP.
132 ; AT THE TIME THE NEW COLUMN STATUS IS COMPARED TO THE OLD.  THE RESULT IS
133 ; STORED IN THE COMPARE REGISTER.  THE PROGRAM IS THEN ROUTED ACCORDING TO
134 ; WHETHER OR NOT A CHANGE WAS DETECTED.
135 ;
136 ; *****
137 ;
001D FA          138  ADJREQ:  MOV     A, R2           ; SCAN ROW SELECT TO ACCUMULATOR
001E 330F        139          ANL     A, #0FH          ; CHECK FOR 0 SCAN VALUE ONLY, NOT INTERRUPT
0020 C626        140          JZ      RSETRG          ; IF 0 RESET REGISTERS
0022 CB          141          DEC     R0             ; DECREMENT MATRIX MAP POINTER
0023 CA          142          DEC     R2             ; DECREMENT SCAN ROW SELECT
0024 042C        143          JMP     SCANMX          ; SCAN MATRIX
0026 BB3F        144  RSETRG:  MOV     R0, #3FH        ; RESET MATRIX MAP POINTER REGISTER, TOP ADDRESS
0028 FA          145          MOV     A, R2             ; SCAN ROW SELECT TO ACCUMULATOR
0029 430F        146          ORL     A, #0FH          ; RESET SCAN ROW SELECT, NO INTERRUPT CHANGE
002B AA          147          MOV     R2, A             ; SCAN ROW SELECT REGISTER
002C FA          148  SCANMX:  MOV     A, R2             ; SCAN ROW SELECT TO ACCUMULATOR
002D 3A          149          OUTL    P2, A             ; OUTPUT SCAN ROW SELECT TO PORT 2
002E BB0F        150          MOV     R3, #SCANTH        ; SET DELAY FOR OUTPUT SCAN TIME
0030 EB30        151  DELAY2:  DJNZ    R3, DELAY2      ; DELAY
0032 09          152          IN      A, P1             ; INPUT COLUMN STATUS FROM PORT 1 TO ACCUMULATOR
0033 20          153          XCH     A, @R0             ; STORE NEW COLUMN STATUS SAVE OLD IN ACCUMULATOR
0034 D0          154          XRL     A, @R0             ; COMPARE OLD WITH NEW COLUMN STATUS
0035 AF          155          MOV     R7, A             ; SAVE COMPARE RESULT IN COMPARE REGISTER
0036 C669        156          JZ      CHFFUL           ; IF THE SAME, CHECK IF FIFO IS FULL
157 ;
158 *EJECT

```

# APPLICATIONS

LOC	OBJ	LINE	SOURCE STATEMENT
		159	*****
		160	;
		161	;
		162	CHANGE WORD ENCODING
		163	;
		164	THE FOLLOWING CODE IS THE CHANGE WORD ENCODING SECTION. THIS
		165	SECTION IS ONLY EXECUTED IF A CHANGE WAS DETECTED. THE COLUMN COUNTER
		166	IS SET AND DECREMENTED TO DESIGNATE EACH OF THE 8 COLUMNS. THE COMPARE
		167	REGISTER IS LOOKED AT ONE BIT AT A TIME TO FIND THE EXACT LOCATION OF
		168	THE CHANGE(S). WHEN A CHANGE IS FOUND IT IS ENCODED BY GIVING IT A
		169	COORDINATE FOR ITS LOCATION. THIS IS DONE BY COMBINING THE PRESENT VALUE
		170	IN THE ROW SCAN SELECT REGISTER AND THE COLUMN COUNTER. THE ACTUAL STATUS
		171	OF THAT SENSOR IS ESTABLISHED BY LOOKING AT THE CORRESPONDING BYTE IN
		172	THE MATRIX MAP. THIS STATUS IS COMBINED WITH THE COORDINATE TO ESTABLISH
		173	THE CHANGE WORD. THE CHANGE WORD IS THEN STORED IN THE CHANGE WORD REGISTER
		174	;
		175	*****
003B	8B0B	176	MOV R3,#0BH ;SET COLUMN COUNTER REGISTER TO 8
003A	CB	177	RRLOOK: DEC R3 ;DECREMENT COLUMN COUNTER
003B	FO	178	MOV A,@R0 ;COLUMN STATUS TO ACCUMULATOR
003C	77	179	RR A ;ROTATE COLUMN STATUS RIGHT
003D	A0	180	MOV @R0,A ;ROTATED COLUMN STATUS BACK TO MATRIX MAP
003E	FF	181	MOV A,R7 ;COMPARE REGISTER VALUE TO ACCUMULATOR
003F	77	182	RR A ;ROTATE COMPARE VALUE RIGHT
0040	AF	183	MOV R7,A ;ROTATED COMPARE VALUE TO COMPARE REGISTER
0041	F245	184	JB7 ENCODE ;TEST BIT 7 IF CHANGE DETECTED ENCODE CHANGE WORD
0043	0469	185	JMP CHFFUL ;IF NO CHANGE IS DETECTED CHECK FOR FIFO FULL
0045	FA	186	ENCODE: MOV A,R2 ;SCAN ROW SELECT TO ACCUMULATOR 0000XXXX
0046	530F	187	ANL A,#0FH ;ROTATE ONLY SCAN VALUE
0048	E7	188	RL A ;ROTATE LEFT 000XXXXX
0049	E7	189	RL A ;ROTATE LEFT 00XXXXX0
004A	E7	190	RL A ;ROTATE LEFT 0XXXXX00
004B	4B	191	ORL A,R3 ;ESTABLISH MATRIX COORDINANT 0XXXXXXX
		192	;(OR) COLUMN COUNTER VALUE WITH ACCUMULATOR
004C	AE	193	MOV R6,A ;SAVE COORDINANT IN CHANGE WORD REGISTER
004D	FO	194	MOV A,@R0 ;COLUMN STATUS FROM MATRIX MAP TO ACCUMULATOR
004E	53B0	195	ANL A,#0BH ;0 ALL BITS BUT BIT 7
0050	4E	196	ORL A,R6 ;(OR) SENSOR STATUS WITH COORDINATE FOR COMPLETED CHANGE WORD
0051	AE	197	MOV R6,A ;SAVE CHANGE WORD XXXXXXXX
		198	;
		199	*EJECT

# APPLICATIONS

LOC	OBJ	LINE	SOURCE STATEMENT
		200	*****
		201	
		202	FIFO-DBBOUT MANAGEMENT
		203	
		204	THE FOLLOWING CODE IS THE FIFO-DBBOUT MANAGEMENT SECTION OF THE
		205	PROGRAM. THIS SECTION TAKES AN ENCODED CHANGE WORD AND LOADS IT INTO
		206	THE FIFO. THE FIFO NOT EMPTY INTERRUPT IS THEN SET AND THE FIFO-IN
		207	POINTER GETS UPDATED. A FIFO FULL CONDITION IS THEN CHECKED FOR AND
		208	ROUTED ACCORDINGLY. IF BOTH THE FIFO AND OBF HAVE CHANGE WORDS THE
		209	PROGRAM LOCKS UP UNTIL THIS HAS CHANGED. IF THE FIFO ISNT FULL COLUMN
		210	COUNTER= 0, FIFO EMPTY AND OBF CONDITIONS ARE CHECKED. THE FIFO-OUT
		211	POINTER IS SET AND DBBOUT IS LOADED IF THE FIFO ISNT EMPTY AND OBF ISNT
		212	SET. IF THIS ISNT THE SITUATION, PROGRAM FLOW IS ROUTED BACK TO THE
		213	THE SCAN AND COMPARE SECTION TO SCAN THE NEXT ROW.
		214	
		215	*****
		216	
0052	FC	217	LOADFF: MOV A,R4 ;FIFO INPUT ADDRESS TO ACCUMULATOR
0053	A9	218	MOV R1,A ;FIFO POINTER USED FOR INPUT
0054	FE	219	MOV A,R6 ;CHANGE WORD TO ACCUMULATOR
0055	A1	220	MOV @R1,A ;LOAD FIFO AT FIFO INPUT ADDRESS
0056	2310	221	STATNE: MOV A,#10H ;BIT 4 FOR FIFO NOT EMPTY
0058	90	222	MOV STS,A ;WRITE TO STATUS REGISTER, FIFO NOT EMPTY
0059	8A20	223	INTRH1: ORL P2,#20H ;FIFO NOT EMPTY INTERRUPT PORT 2-5 HIGH
005B	FA	224	MOV A,R2 ;ROW SCAN SELECT TO ACCUMULATOR
005C	4320	225	ORL A,#20H ;SAVE INTERRUPT, NO CHANGE TO SCAN VALUE
005E	AA	226	MOV R2,A ;ROW SCAN SELECT REGISTER
005F	232F	227	ADJFIN: MOV A,#FIFOTA ;FIFO TOP ADDRESS TO ACCUMULATOR
0061	DC	228	XRL A,R4 ;COMPARE WITH CURRENT FIFO INPUT ADDRESS
0062	C667	229	JZ RSFFIN ;IF THE SAME RESET FIFO INPUT REGISTER
0064	1C	230	INC R4 ;NEXT FIFO INPUT ADDRESS
0065	0469	231	JMP CHFFUL ;CHECK FIFO FULL
0067	BC08	232	RSFFIN: MOV R4,#FIFOBA ;RESET FIFO INPUT REGISTER, BOTTOM OF FIFO
0069	FC	233	CHFFUL: MOV A,R4 ;FIFO INPUT ADDRESS TO ACCUMULATOR
006A	DD	234	XRL A,R5 ;COMPARE INPUT WITH OUTPUT FIFO ADDRESS
006B	967D	235	JNZ CHCNTR ;IF NOT SAME CHECK COLUMN COUNTER VALUE
006D	866D	236	CHOBFI: JOBF CHOBFI ;IF OBF IS 1 THEN CHECK OBF
006F	232F	237	ADJFOT: MOV A,#FIFOTA ;FIFO TOP ADDRESS TO ACCUMULATOR
0071	DD	238	XRL A,R5 ;COMPARE TOP TO OUTPUT FIFO ADDRESS
0072	C677	239	JZ RSFFOT ;IF THE SAME RESET FIFO OUTPUT REGISTER
0074	1D	240	INC R5 ;NEXT FIFO OUTPUT ADDRESS
0075	0479	241	JMP LOADDB ;LOAD DBBOUT
0077	BD08	242	RSFFOT: MOV R5,#FIFOBA ;RESET FIFO OUTPUT ADDRESS TO BOTTOM OF FIFO
0079	FD	243	LOADDB: MOV A,R5 ;OUTPUT FIFO ADDRESS TO ACCUMULATOR
007A	A9	244	MOV R1,A ;FIFO POINTER USED FOR OUTPUT
007B	F1	245	MOV A,@R1 ;CHANGE WORD TO ACCUMULATOR
007C	02	246	OUT DBB,A ;CHANGE WORD TO DBBOUT
007D	FB	247	CHCNTR: MOV A,R3 ;COLUMN COUNTER TO ACCUMULATOR
007E	963A	248	JNZ RRL00K ;IF NOT 0 FINISH CHANGE WORD ENCODING
0080	2308	249	CHFFEM: MOV A,#FIFOBA ;FIFO BOTTOM ADDRESS TO ACCUMULATOR
		250	
		251	*EJECT

# APPLICATIONS

ISIS-II MCS-48/UPI-41 MACRO ASSEMBLER, V3 0

PAGE 7

LOC	OBJ	LINE	SOURCE STATEMENT
0082	DC	252	XRL A, R4 ; COMPARE FIFO INPUT ADDRESS WITH FIFO BOTTOM ADD
0083	C6BC	253	JZ ADJFEM ; IF THE SAME, ADJUST TO CHECK FOR FIFO EMPTY
0085	FC	254	MOV A, R4 ; FIFO INPUT ADDRESS TO ACCUMULATOR
0086	07	255	DEC A ; DECREMENT FIFO INPUT ADDRESS IN ACCUMULATOR
0087	DD	256	XRL A, R5 ; COMPARE INPUT TO OUTPUT FIFO ADDRESSES
0088	C691	257	JZ STATMT ; IF SAME, WRITE STATUS REGISTER FOR FIFO EMPTY
008A	049C	258	JMP CH0BF2 ; CHECK OBF
008C	232F	259	ADJFEM MOV A, #FIF0TA ; FIFO TOP ADDRESS TO ACCUMULATOR
008E	DD	260	XRL A, R5 ; COMPARE TOP TO OUTPUT FIFO ADDRESS
008F	969C	261	JNZ CH0BF2 ; IF NOT SAME THEN FIFO IS NOT EMPTY, CHECK OBF
0091	2300	262	STATMT: MOV A, #00H ; CLEAR BIT 0 FOR FIFO EMPTY
0093	90	263	MOV STS, A ; WRITE TO STATUS REGISTER
0094	9ADF	264	INTRLD: ANL P2, #ODFH ; FIFO EMPTY, INTERRUPT PORT 2-5 LOW
0096	FA	265	MOV A, R2 ; SCAN ROW SELECT TO ACCUMULATOR
0097	53DF	266	ANL A, #ODFH ; SAVE INTERRUPT, NO CHANGE TO SCAN VALUE
0099	AA	267	MOV R2, A ; SCAN ROW SELECT REGISTER
009A	041D	268	JMP ADJREG ; ADJUST REGISTERS
009C	861D	269	CH0BF2: J0BF ADJREG ; IF OBF=1 THEN ADJUST REGISTERS
009E	046F	270	JMP ADJFOT ; ADJUST FIFO OUT ADDRESS TO LOAD DBBOUT
		271	
		272	END

USER SYMBOLS

ADJFEM 008C	ADJFIN 005F	ADJFOT 006F	ADJREG 001D	CHCNTR 007D	CHFFEM 0080	CHFFUL 0069	CH0BF1 006D
CH0BF2 009C	DELAY2 0030	ENCODE 0045	FIF0BA 0008	FIF0TA 002F	FILLMX 000D	INITMX 0000	INTRH1 0059
INTRLD 0094	LDADD8 0079	LOADFF 0052	0BFINT 0018	RRL00K 003A	RSETRG 0026	RSFFIN 0067	RSFFOT 0077
SCANMX 002C	SCANTH 000F	STATMT 0091	STATNE 0056				

ASSEMBLY COMPLETE. NO ERRORS

---

# **An 8741A/8041A Digital Cassette Controller**

## **Contents**

<b>INTRODUCTION</b>	<b>2-46</b>
<b>THE CM-600 MINI-DEK®</b>	<b>2-47</b>
<b>RECORDING FORMAT</b>	<b>2-47</b>
<b>THE UPI-41A™ CONTROLLER</b>	<b>2-48</b>
<b>THE HARDWARE INTERFACE</b>	<b>2-50</b>
<b>THE CONTROLLER SOFTWARE</b>	<b>2-52</b>
Write Command	2-54
Read Operation	2-56
Skip Operation	2-59
Rewind Operation	2-60
Abort Operation	2-60
<b>WRAPPING IT UP</b>	<b>2-60</b>
<b>APPENDIX</b>	<b>2-62</b>

## INTRODUCTION

The microcomputer system designer requiring a low-cost, non-volatile storage medium has a difficult choice. His options have been either relatively expensive, as with floppy discs and bubble memories, or non-transportable, like battery backed-up RAMs. The full-sized digital cassette option was open but many times it too was too expensive for the application. Filling this void of low-cost storage is the recently developed digital mini-cassette. These mini-cassettes are similar to, but not compatible with, dictation cassettes. The mini-cassette transports are inexpensive (well under \$100 in quantity), small (less than 25 cu. in.), low-power (one watt), and their storage capacity is a respectable 200K bytes of unformatted data on a 100-foot tape. These characteristics make the mini-cassette perfect for applications ranging from remote datalogging to program storage for hobbyist systems.

The only problem associated with mini-cassette drives is controlling them. While these drives are relatively easy to interface to a microcomputer system, via a parallel I/O port, they can quickly overburden a CPU if other concurrent or critical real-time I/O is required. The cleanest and probably

the least expensive solution in terms of development cost is to use a dedicated single-chip controller. However, a quick search through the literature turns up no controllers compatible with these new transports. What to do? Enter the UPI-41A family of Universal Peripheral Interfaces.

The UPI-41A family is a group of two user-programmable slave microcomputers plus a companion I/O expander. The 8741A is the "flag-chip" of the line. It is a complete microcomputer with 1024 bytes of EPROM program memory, 64 bytes of RAM data memory, 16 individually programmable I/O lines, an 8-bit event counter and timer, and a complete slave peripheral interface with two interrupts and Direct Memory Access (DMA) control. The 8041A is the masked ROM, pin compatible version of the 8741A. Figure 2 shows a block diagram common to both parts. The 8243 I/O port expander completes the family. Each 8243 provides 16 programmable I/O lines.

Using the UPI concept, the designer can develop a custom peripheral control processor for his particular I/O problem. The designer simply develops his peripheral control algorithm using the UPI-41A assembly language and programs the EPROM of

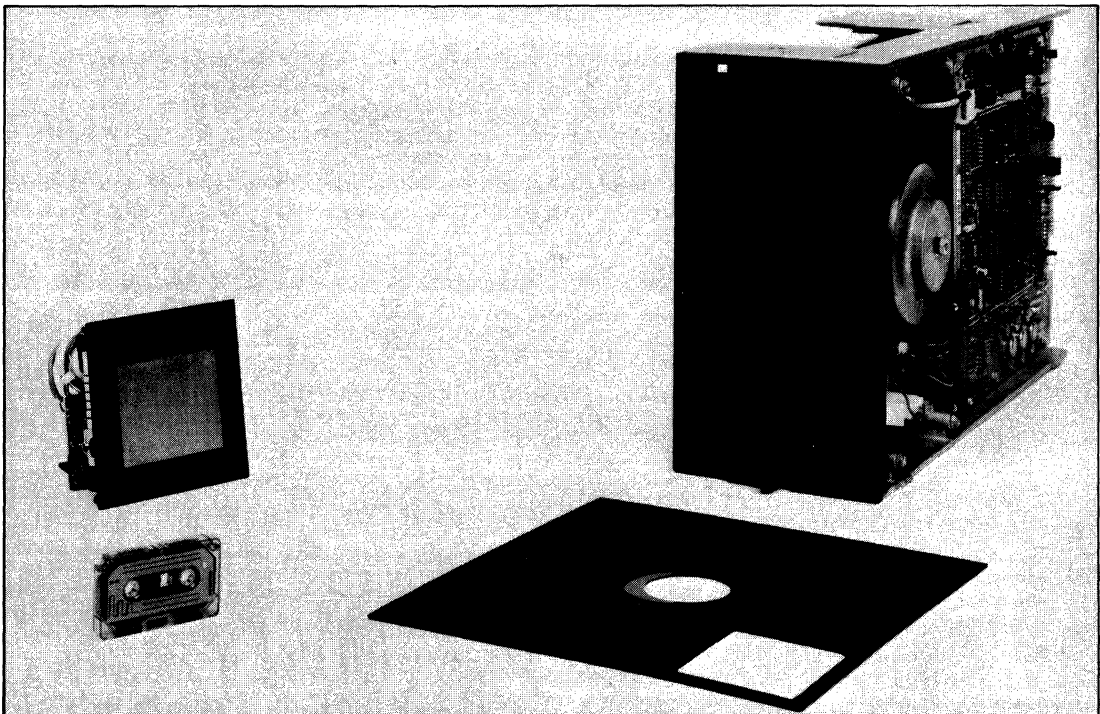
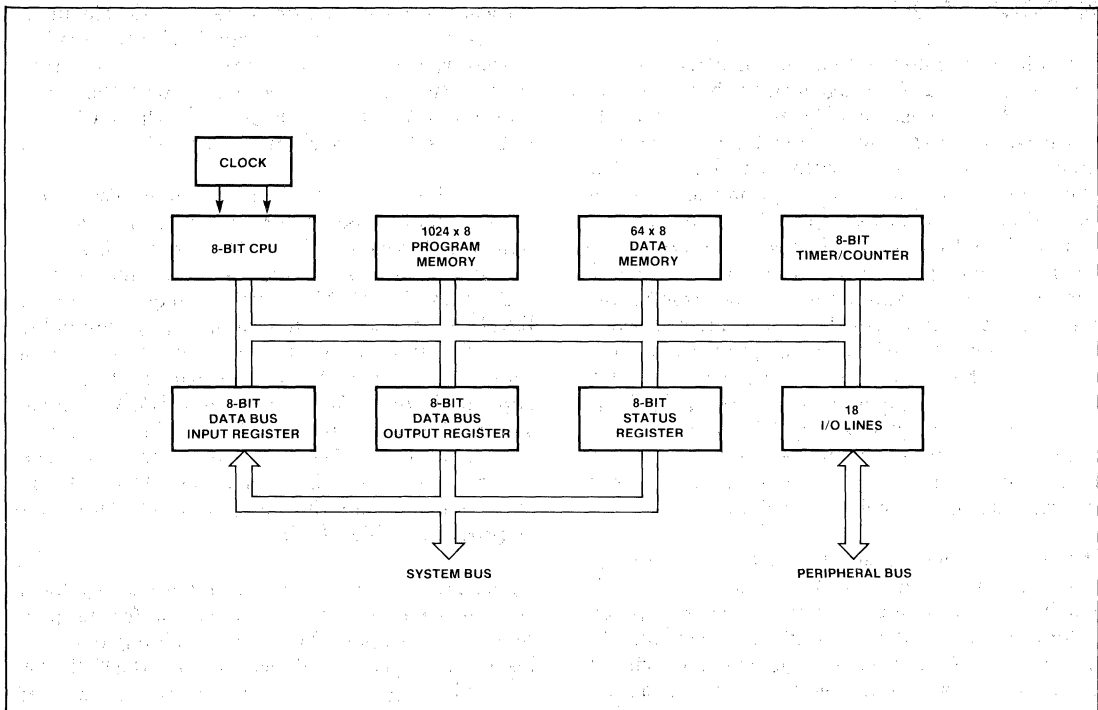


Figure 1. Comparison of Mini-Cassette and Floppy Disk Transports and Media.





**Figure 2. 8741A/8041A Block Diagram**

the 8741A. Voila! He has a single-chip dedicated controller. Testing may be accomplished using either an ICE-41A or the Single-step mode of the 8741A. UPI-41A peripheral interfaces are being used to control printers, keyboards, displays, custom serial interfaces, and data encryption units. Of course, the UPI family is perfect for developing a dedicated controller for digital mini-cassette transports. To illustrate this application for the UPI family let's consider the job of controlling the Braemar CM-600 Mini-Dek®.

### THE CM-600 MINI-DEK®

The Braemar CM-600 is representative of digital mini-cassette transports. It is a single-head, single-motor transport which operates entirely from a single 5-volt power supply. Its power requirements, including the motor, are 200ma for read or write and 700ma for rewind. Tapes speeds are 3 inches per second (IPS) during read or write, 5 IPS fast forward, and 15 IPS rewind. With these speeds and a maximum recording density of 800 bits per inch (BPI), the maximum data rate is 2400 bits per second (BAUD). The data capacity using both sides of a 100-foot tape is 200K bytes. On top of this,

the transport occupies only 22.5 cubic inches (3" x 3" x 2.5").

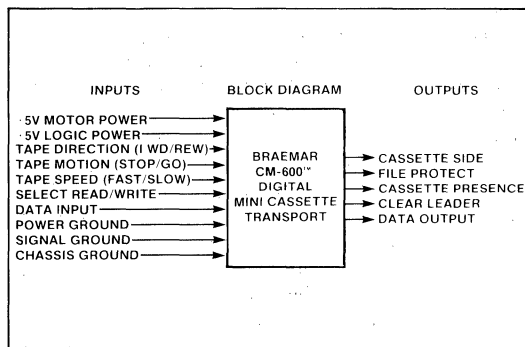
All I/O for the CM-600 is TTL-compatible and can be divided into three groups: motor control, data control, and cassette status. The motor group controls are GO/STOP, FAST/SLOW, and FORWARD/REVERSE. The data controls are READ/ WRITE, DATA IN, and DATA OUT. The remaining group of outputs give the transport's status: CLEAR LEADER, CASSETTE PRESENCE, FILE PROTECT, and SIDE SENSOR. These signals, shown schematically in figure 3 and table 1, give the pin definition of the CM-600 16-pin I/O connector.

### RECORDING FORMAT

The CM-600 does not provide either encoding or decoding of the recorded data; that task is left for the peripheral interface. A multitude of encoding techniques from which the user may choose are available. In this single-chip dedicated controller application, a "self-clocking" phase encoding scheme similar to that used in floppy discs was chosen. This scheme specifies that a logic "0" is a bit cell with no transition; a cell with a transition is a logic "1."

**Table 1. CM-600 I/O Pin Definition**

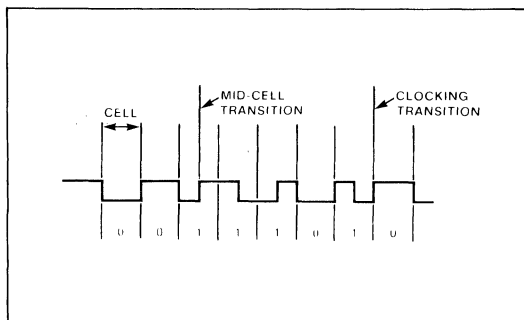
Pin	I/O	Function
1	—	Index pin—not used
2	—	Signal ground
3	O	Cassette side (0—side B, 1—side A)
4	I	Data input (0—space, 1—mark)
5	O	Cassette presence (0—cassette, 1—no cassette)
6	I	Read/Write (0—read, 1—write)
7	O	File protect (0—tab present, 1—tab removed)
8	—	+5v motor power
9	—	Power ground
10	—	Chassis ground
11	I	Direction (0—forward, 1—rewind)
12	I	Speed (0—fast, 1—slow)
13	O	Data output (0—space, 1—mark)
14	O	Clear leader (0—clear leader, 1—off clear leader)
15	I	Motion (0—go, 1—stop)
16	—	+5v logic power



**Figure 3. Braemar CM-600™ Block Diagram**

Figure 4 illustrates the encoding of the character 3AH assuming the previous data ended with the data line high. (The least significant bit is sent first.) Notice that there is always a "clocking" transition at the beginning of each cell. Decoding is simply a matter of triggering on this "clocking" transition, waiting 3/4 of a bit cell time, and determining whether a mid-cell transition has occurred. Cells with no mid-cell transitions are data 0's; cells with transitions are data 1's. This encoding technique has all the benefits of Manchester encoding with the added advantage that the encoded data may be "decoded by eyeball:" long cells are always 0's, short cells are always 1's.

Besides the encoding scheme, the data format is also up to the user. This controller uses a variable byte length, checksum protected block format. Every block starts and ends with a SYNC character



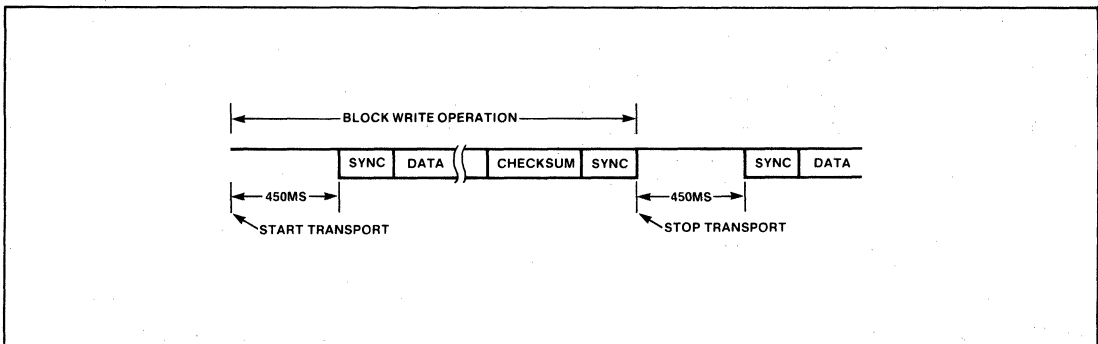
**Figure 4. Modified Phase Encoding of Character 3A Hex**

(AAH), and the character immediately preceding the last SYNC is the checksum. The checksum is capable of catching 2 bit errors. The number of data characters within a block is limited to 64K bytes. Blocks are separated by an Inter-Record Gap (IRG). The IRG is of such a length that the transport can stop and start within an IRG, as illustrated in the data block timing, figure 5. Braemar specifies a maximum start or stop time of 150ms for the transport, thus the controller uses 450ms for the IRG. This gives plenty of margin for controlling the transport and also for detecting IRGs while skipping blocks.

## THE UPI-41A CONTROLLER

The goal of the UPI software design for this application was to make the UPI-41A microcomputer into an intelligent cassette control processor. The host processor (8086, 8088, 8085A, etc.) simply issues a high-level command such as READ-a-block or WRITE-a-block. The 8741A accepts the command, performs the requested operation, and returns to the host system a result code telling the outcome of the operation, eg. Good-Completion, Sync Error, etc. Table 2 shows the command and result code repertoire. The 8741A completely manages all the data transfers for reading and writing.

As an example, consider the WRITE-a-block command. When this command is issued, the UPI-41A expects a 16-bit number from the host telling how many data bytes to write (up to 64K bytes per block). Once this number is supplied in the form of two bytes, the host is free to perform other tasks; a bit in the UPI's STATUS register or an interrupt output will notify the host when a data transfer is required. The 8741A then checks the transport's status to be sure that a cassette is present and not file protected. If either is false, a result code is



**Figure 5. IRG/Block Timing Diagram (not to scale)**

**Table 2. Controller Command/Result Code Set**

Command	Result
Read (01H)	Good-Completion (00H) Buffer Overrun Error (41H) Bad Synch1 Error (42H) Bad Synch2 Error (43H) Checksum Error (44H) Command Error (45H) End of Tape Error (46H)
Rewind (04H)	Good-Completion (00H)
Skip (03H)	Good-Completion (00H) End of Tape Error (47H) Beginning of Tape Error (48H)
Write (02H)	Good-Completion (00H) Buffer Underrun Error (81H) Command Error (82H) End of Tape Error (83H)

returned to the host; otherwise the transport is started. After the peripheral controller checks to make sure that the tape is off the clear leader and past the hole in the tape, it writes a 450ms IRG, a SYNC character, the block of data, the checksum, and the final SYNC character. (The tape has a clear leader at both ends and a small hole 6 inches from the end of each leader.) The data transfers from the host to the UPI-41A slave microcomputer are double buffered. The controller requests only the desired number of data bytes by keeping track of the count internally.

If nothing unusual happened, such as finding clear leader while writing, it returns a Good-Completion result code to the host. If clear leader was encountered, the transport is stopped immediately and an End-of-Tape result code is returned to the host. Another possible error would be if the host is late in supplying data. If this occurs, the controller writes

an IRG, stops the drive, and returns the appropriate Data-Underrun result code.

The READ-a-block command also provides error checking. Once this command is issued by the host, the controller checks for cassette presence. If present, it starts the transport. The data output from the transport is then examined and decoded continuously. If the first character is not a SYNC, that's an error and the controller returns a Bad-First-SYNC result code (42H) after advancing to the next IRG. If the SYNC is good, the succeeding characters are read into an on-chip 30 character circular buffer. This continues until an IRG is encountered. When this occurs, the transport is stopped. The controller then tests that the last character. If it is a SYNC, the controller then compares the accumulated internal checksum to the block's checksum, the second to the last character of the block. If they match, a Good-Completion result code (00H) is returned to the host. If either test is bad, the appropriate error result code is returned. The READ command also checks for the End-of-Tape (EOT) clear leader and returns the appropriate error result code if it is found before the read operation is complete.

The 30 character circular buffer allows the host up to 30 character times of response time before the host must collect the data. All data transfers take place thru the UPI-41A Data Bus Buffer Output register (DBBOUT). The controller continually monitors the status of this register and moves characters from the circular buffer to the register whenever it is empty.

The SKIP-n-blocks command allows the host to skip the transport forward or backward up to 127 blocks. Once the command is issued, the controller expects one data byte specifying the number of

# APPLICATIONS

blocks to skip. The most significant bit of this byte selects the direction of the skip (0=forward, 1=reverse). SKIP is a dual-speed operation in the forward direction. If the number of blocks to skip is greater than 8, the controller uses fast-forward (5 IPS) until it is within 8 blocks of the desired location. Once within 8 blocks, the controller switches to the normal read speed (3 IPS) to allow accurate placement of the tape. The reverse skip uses only the rewind speed (15 IPS). Like the READ and WRITE commands, SKIP also checks for EOT and beginning-of-tape (BOT) depending upon the tape's direction. An error result code is returned if either is encountered before the number of blocks skipped is complete.

The REWIND command simply rewinds the tape to the BOT clear leader. The ABORT command allows the termination of any operation in progress, except a REWIND. All commands, including ABORT, always leave the tape positioned on an IRG.

## THE HARDWARE INTERFACE

There's hardly any hardware design effort required for the controller and transport interface in figure 6. Since the CM-600 is TTL compatible, it connects

directly to the I/O ports of the UPI controller. If the two are separated (i.e. on different PC cards), it is recommended that TTL buffers be provided.) The only external circuitry needed is an LED driver for the DRIVE ACTIVE status indicator.

The 8741A-to-host interface is equally straightforward. It has a standard asynchronous peripheral interface: 8 data lines (D<sub>0</sub>-D<sub>7</sub>), read (RD), write (WR), register select (AO), and chip select (CS). Thus it connects directly to an 8086, 8088, 8085A, 8080, or 8048 bus structure. Two interrupt outputs are provided for data transfer requests if the particular system is interrupt-driven. DMA transfer capability is also available. The clock input can be driven from a crystal directly or with the system clock (6MHz max). The UPI-41A clock may be asynchronous with respect to other clocks within the system.

This application was developed on an Intel iSBC 80/30 single board computer. The iSBC 80/30 is controlled by an 8085A microprocessor, contains 16K bytes of dual-ported dynamic RAM and up to 8K bytes of either EPROM or ROM. Its I/O complement consists of an 8255A Programmable Parallel Interface, an 8251A Programmable Communica-

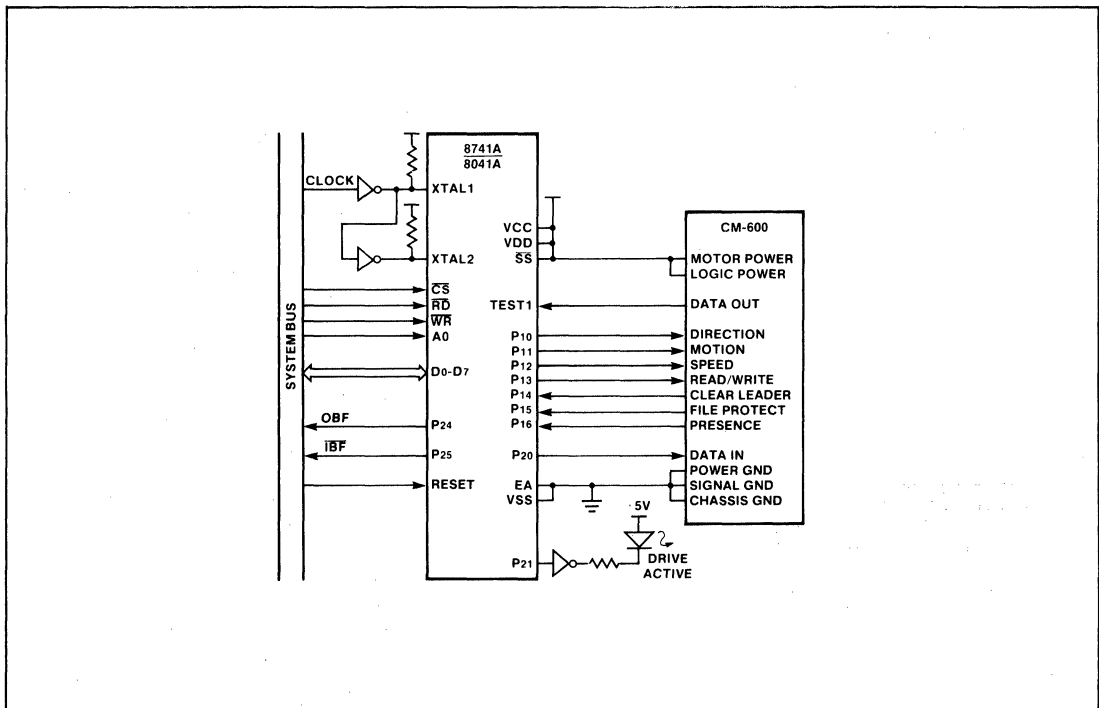
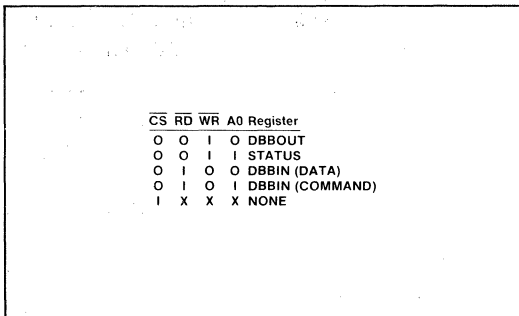


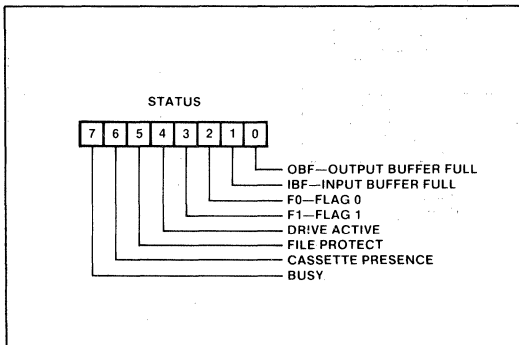
Figure 6. Controller/Transport System Schematic

tions Interface, an 8253 Programmable Interval Timer, and an 8259A Programmable Interrupt Controller. The iSBC 80/30 is especially convenient for UPI development since it contains an uncommitted socket dedicated to either an 8041A or 8741A, complete with buffering for its I/O ports. The iSBC 80/30 to 8741A interface is reflected in figure 8. (Optionally, an iSBC 569 Digital Controller board could be used. The iSBC 569 board contains three uncommitted UPI sockets with an interface similar to that in figure 8.)

Looking at the host-to-controller interface, the host sees the 8741A as three registers in the host's I/O address space: the data register, the command register, and the status register. The decoding of these registers is shown in figure 7. All data and commands for the controller are written into the Data Bus Buffer Input register (DBBIN). The state of the register select input, AO, determines whether a command or data is written. (Writes with AO set to 1 are commands by convention.) All data and results from the controller are read by the host from the Data Bus Buffer Output register (DBBOUT).



**Figure 7. 8741A/8041A Interface Register Decoding**



**Figure 8. Status Register Bit Definition**

The Status register contains flags which give the host the status of various operations within the controller. Its format is given in figure 8. The Input Buffer Full (IBF) and Output Buffer Full (OBF) flags show the Status of the DBBIN and DBBOUT registers respectively. IBF indicates when the DBBIN register contains data written by the host. The host may write to DBBIN only when IBF is 0. Likewise, the host may read DBBOUT only when OBF is set to a 1. These bits are handled automatically by the UPI-41A internal hardware. FLAG 0 (F<sub>0</sub>) and FLAG 1 (F<sub>1</sub>) are general purpose flags used internally by the controller which have no meaning externally.

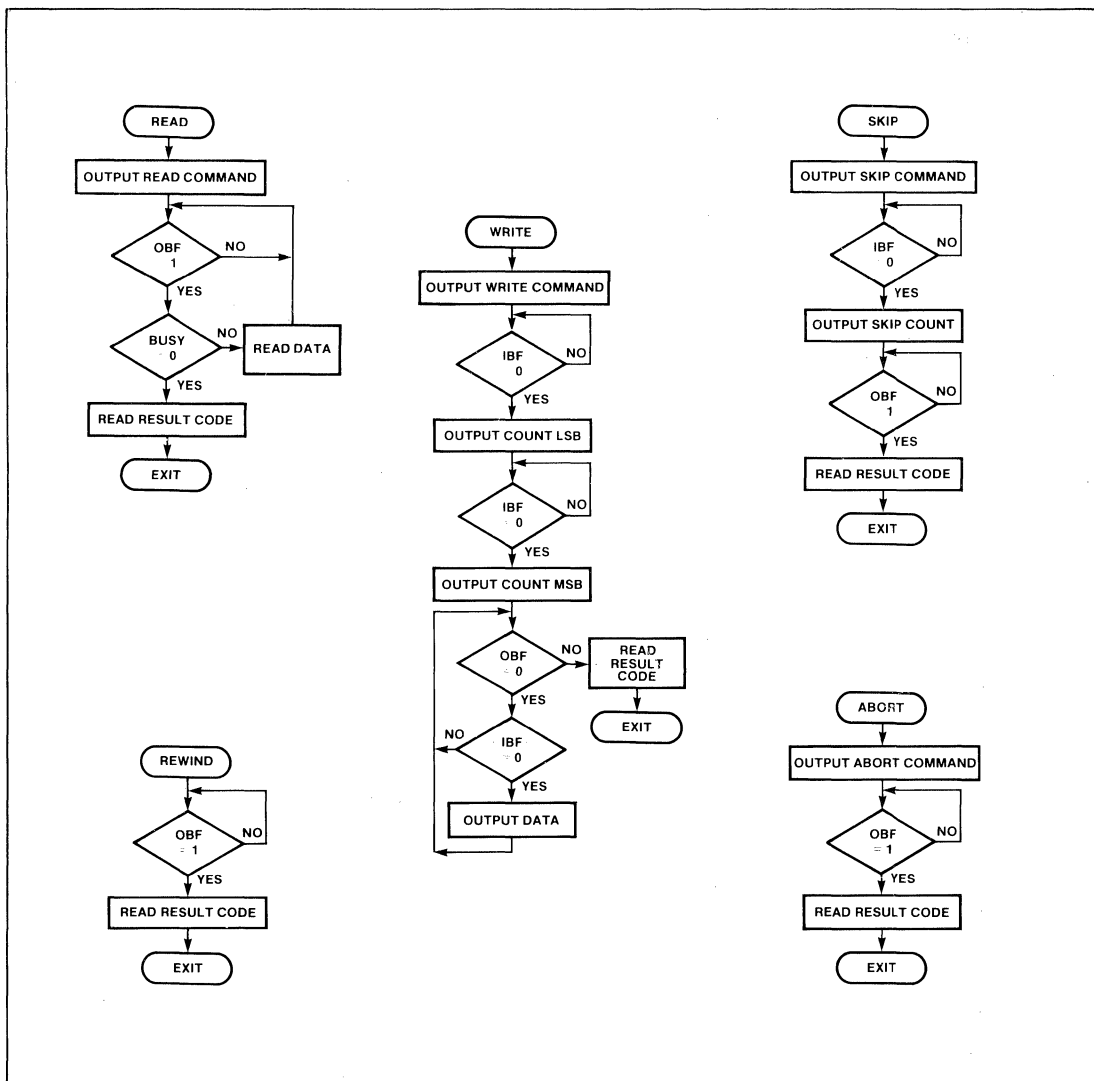
The remaining four bits are user-definable. For this application they are DRIVE ACTIVE, FILE PROTECT, CASSETTE PRESENCE, and BUSY flags. The FILE PROTECT and CASSETTE PRESENCE flags reflect the state of the corresponding I/O lines from the transport. DRIVE ACTIVE is set whenever the transport motor is on and the controller is performing an operation. The BUSY flag indicates whether the contents of the DBBOUT register is data or a result code. The BUSY flag is set whenever a command is issued by the host and accepted by the controller. As long as BUSY is set, any character found in DBBOUT is a result code. Thus whenever the host finds OBF set, it should test the BUSY flag to determine whether the character is data or a result code.

Notice the OBF and  $\overline{\text{IBF}}$  are available as interrupt outputs to the host processor, figure 6. These outputs are self-clearing, that is, OBF is set automatically upon the controller loading DBBOUT and cleared automatically by the host reading DBBOUT. Likewise  $\overline{\text{IBF}}$  is cleared to a 0 by the host writing into DBBIN: set to a 1 when the controller reads DBBIN into the accumulator.

The flow charts of figure 9 show the flow of sample host software assuming a polling software interface between the host and the controller. The WRITE command requires two additional count bytes which form the 16-bit byte count. These extra bytes are "handshaked" into the controller using the IBF flag in the STATUS register. Once these bytes are written, the host writes data in response to IBF being cleared. This continues until the host finds OBF set indicating that the operation is complete and reads the result code from DBBOUT. No testing of BUSY is needed since only the result code appears in the DBBOUT register.

The READ command does require that BUSY be tested. Once the READ command is written into the

# APPLICATIONS



**Figure 9. Host CPU Flow Charts for Commands When Polling is Used**

controller, the host must test BUSY whenever OBF is set to determine whether the contents of DBBOUT is data from the tape or the result code.

The SKIP command requires the skip count byte. This byte is written into DBBIN after IBF has been cleared following the command. The host then waits until OBF is set indicating the operation is complete and the result code is waiting in DBBOUT. The REWIND and ABORT commands only require that the host test OBF. Once set, the result code is ready in DBBOUT.

The flow charts for an interrupt-driven system are simplified since no testing of OBF or IBF is required. The mere fact that an interrupt occurred implies that the corresponding bit in the STATUS REGISTER is set or cleared.

## THE CONTROLLER SOFTWARE

The internal UPI-41A software can be divided roughly into the various commands. (This software is discussed as flow charts. The actual program listing is included in Appendix A.) A command

recognizer simply waits for a command input by the host and then branches to the appropriate command routine. The command routine executes until the entire operation is complete and then branches back to the command recognizer. Since only one command routine is executing at any one time, the working registers change function based upon which command is active. Figure 10 shows the register function and identifying name for each command. Notice that while most registers have completely different meanings depending upon the command, some registers retain their meaning over all commands. All registers were assigned names based on their function to aid programming and to make the listing easier to read.

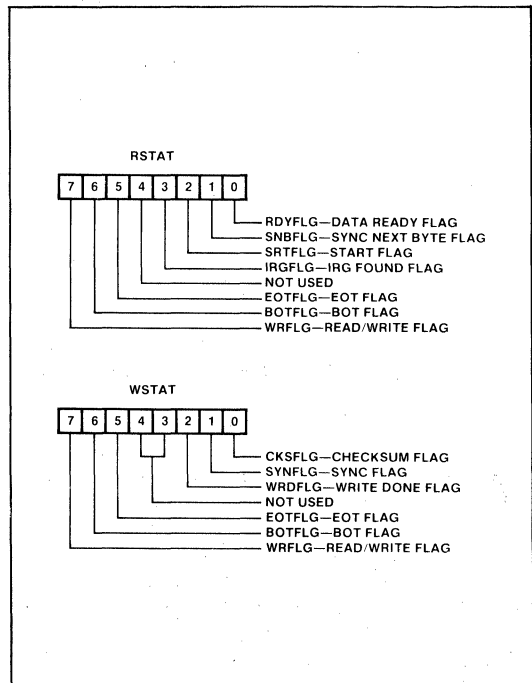
The READ and WRITE commands utilize the internal timer and event counter for all bit timing. This timer provides an internal interrupt on overflow. Thus these commands can be thought of as containing both foreground and background (interrupt service routine) tasks. These tasks communicate via general purpose registers assigned the function of internal status registers: WSTAT and RSTAT for the WRITE and READ commands respectively. The bit definition for these internal status registers is shown in figure 11. We will refer to these bits as the command routines are discussed.

WRITE COMMAND	
• REGISTER BANK 0	
R0	CNTLSB    BYTE COUNT LSB
R1	CNTMSB    BYTE COUNT MSB
R2	CMPSAV    COMMAND SAVE
R3	CHKSUM    CHECKSUM
	ACCUMULATOR
R4	TEMP1     TEMPORARY STORAGE
R5	SOFTWARE DELAY
R6	COUNTERS
R7	STAT      IMAGE OF UPPER 4-BITS OF STATUS
• REGISTER BANK 1	
R0	NOT USED
R1	NOT USED
R2	RESULT    RESULT CODE STORAGE
R3	WSTAT     WRITE STATUS
R4	BITCNT    BIT COUNTER
R5	SERIAL    SERIALIZER
R6	TEMPO     TEMPORARY STORAGE
R7	ASAVE     ACCUMULATOR STORAGE

**Figure 10A. Register Definition for WRITE**

READ/SKIP/REWIND COMMANDS	
• REGISTER BANK 0	
R0	LBOU      BUFFER OUTPUT POINTER
R1	LBRDY     BUFFER READY POINTER
R2	NOT USED
R3	CHKSUM    CHECKSUM ACCUMULATOR
R4	BLKCNT    BLOCK COUNTER FOR SKIP
R5	BLKTIM    COUNTER TO TIME IRG DURING SKIP
R6	BLKSAV    BACKUP FOR BLKTIM
R7	STAT      IMAGE OF UPPER 4-BITS OF STATUS
• REGISTER BANK 1	
R0	LBIN      BUFFER INPUT POINTER
R1	IRGCNT    COUNTER TO TIME IRG DURING READ
R2	RESULT    RESULT CODE STORAGE
R3	RSTAT     READ STATUS REGISTER
R4	BITCNT    BIT COUNTER
R5	DESERL    DE-SERIALIZER
R6	RDATA     READ DATA HOLDING REGISTER
R7	ASAVE     ACCUMULATOR STORAGE

**Figure 10B. Register Definition for READ, SKIP, and REWIND**



**Figure 11. READ and WRITE Internal Status Register Bit Definitions**

## WRITE COMMAND

Let's look at the WRITE command routine first, figure 12. As was mentioned earlier, the WRITE requires two additional data bytes before it can be processed. Once the command recognizer branches to the WRITE routine, the routine waits on IBF until these bytes are written by the host. These count bytes are stored in the CNTLSB and CNTMSB (Count Least and Most Significant Byte) registers. These two registers are concatenated to form the 16-bit byte count. At this point, the routine tests the

transport status lines, CASSETTE PRESENCE and FILE PROTECT. If there is no cassette present or the tape is write protected, the routine exits immediately after resetting BUSY and loading DBBOUT with the appropriate error result code. Assuming the transport status is correct, the other registers required by the routine are initialized: the bit counting register (BITCNT) is set to 8; the checksum accumulator (CHKSUM) is cleared; the data holding register (SERIAL) is loaded with the first SYNC character. The internal timer counter is then loaded with a value which will cause an internal timer interrupt in one half of a bit-cell time, but not activated.

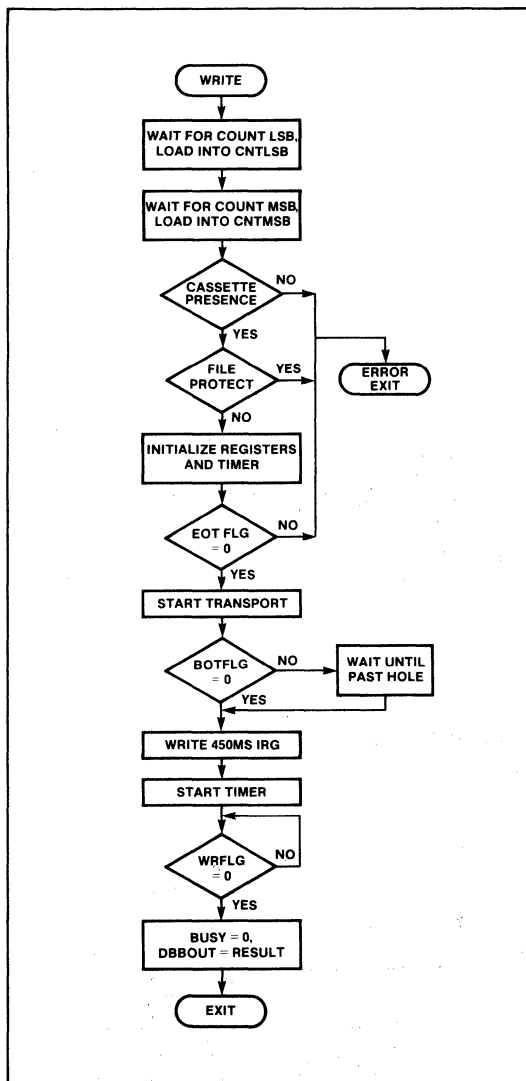
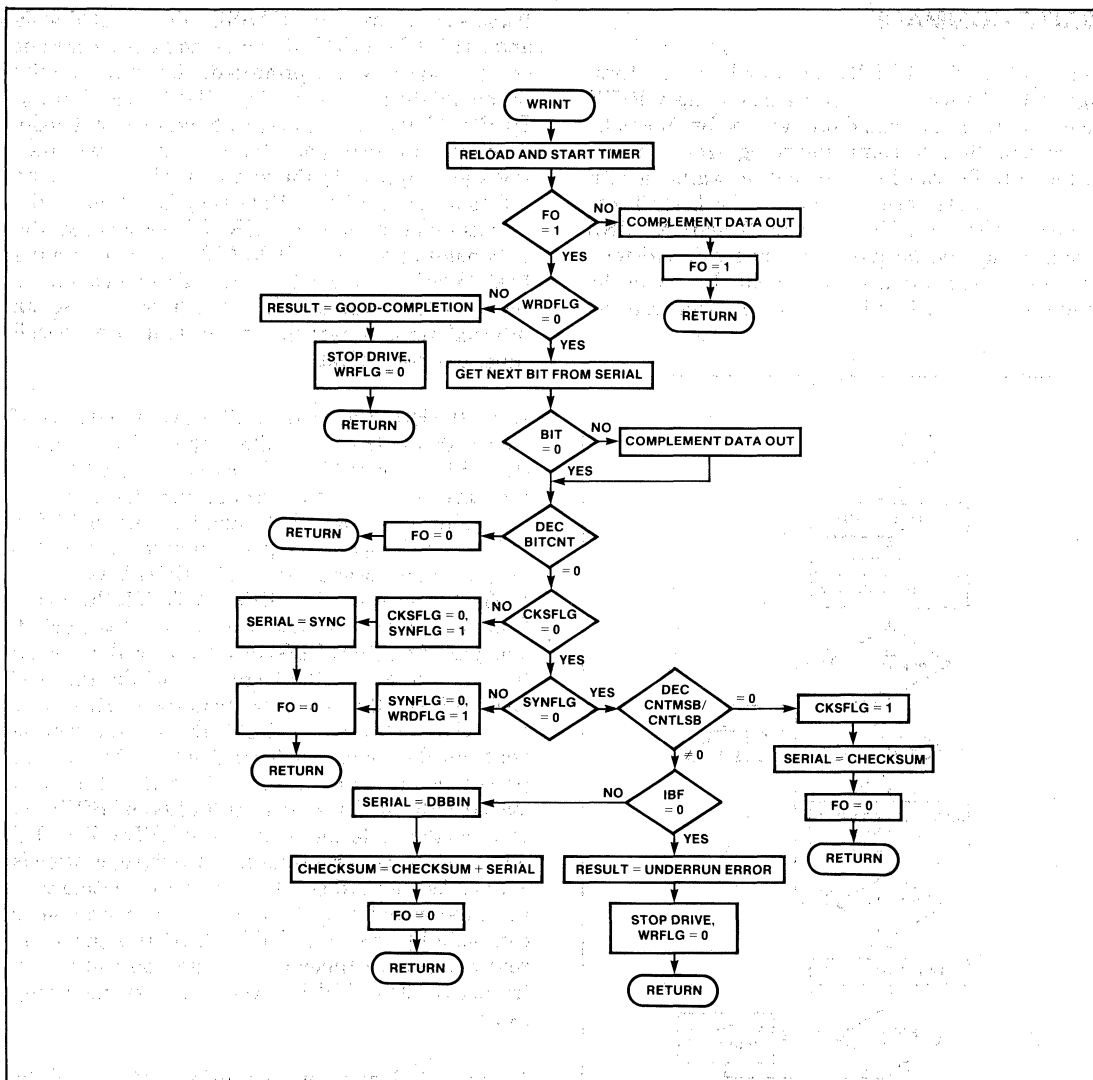


Figure 12. WRITE Command Flow Chart

Next, the EOT flag in WSTAT is examined to see if we are trying to write while at the end of the tape. (EOTFLG is set to 1 if EOT was encountered during the last operation.) If an error occurred, the routine exits after resetting BUSY and loading DBBOUT with the EOT-while-write result error code (83H) via the result storage register, RESULT. Assuming EOTFLG is not set, the DRIVE ACTIVE flag in the Status register is set and the transport is started. The BOT flag (BOTFLG) in WSTAT is then tested to see if we are at the beginning of the tape.) If BOTFLG is 0, the routine writes a 450ms IRG using a software delay loop. If BOTFLG is 1, the routine waits until the clear leader and hole in the tape are passed before starting the IRG. WSTAT is then loaded with 80H. This resets EOTFLG and BOTFLG and sets the write and read flag, WRFLG. WRFLG tells the interrupt routines that a write operation is active. As we shall see, the interrupt routine tells the foreground task that the write operation is complete by resetting WRFLG. At this point the routine starts the timer and enters a loop continually testing WRFLG. If WRFLG is 1, the routine simply loops.

Now let's look at the write routine that does all the work: the write timer interrupt service routine. When the timer interrupt occurs half a bit-cell time later, an automatic vector to the INT routine is performed (location 07H in program memory). INT test WRFLG to see whether it's a read or write operation in progress and branches accordingly. Since we are talking about a write operation, the branch is to the WRINT routine, figure 13. WRINT first reloads the timer to provide the timing for the next half cell (the timer continues to run). The F<sub>0</sub> is used to define whether this particular interrupt is for the first or the second half of the bit cell. The phase encoding algorithm used specifies that the beginning of a bit must always have a transition. If F<sub>0</sub> is reset, the data output to the transport is simply





**Figure 13. WRINT—Write Timer Interrupt Routine Flow Chart**

complemented providing the transition. If set, the interrupt is at the mid-cell position. If the data bit is a 1, complement the data output; otherwise, do not change it.  $F_0$  is complemented every interrupt.

The CLEAR LEADER input from the transport is also tested on every interrupt. If it was encountered, the transport is stopped, the EOTFLG in WSTAT is set, WRFLG is reset, and RESULT is loaded with the EOT-while-write error result code (83H). WRINT returns to the main write loop.

The data contained in the SERIAL register is shifted out bit-by-bit at every other timer interrupt (those interrupts with  $F_0$  set to a 1) until the BITCNT register indicates that all 8 bits have been shifted out. When this occurs, a 16-bit decrement operation on the CNTMSB and CNTLSB registers is performed. If the result is non-zero, the routine transfers the next data byte from DBBIN to SERIAL. If the host is late in getting the next byte into DBBIN, a Write-Underrun error (81H) occurs. Like the other error conditions, WRFLG in WSTAT is reset and

the Write-Underrun error result code is loaded into the result holding register, RESULT, before returning to the main write loop. If the data is ready in DBBIN, it is transferred to SERIAL and added to the accumulating checksum. The routine then returns to the write main foreground task. (Remember that the foreground task is doing nothing more than testing WRFLG.)

If the decrement result is zero, all data transfers are complete. The accumulated checksum value is loaded into Serial and the Checksum flag, CKSFLG, is set in WSTAT before exiting the interrupt routine. This causes the checksum value to be written onto the tape. Sixteen timer interrupts later the checksum is complete; it is now time to write the final SYNC. CKSFLG is reset, a SYNC character is loaded into SERIAL, and the SYNC flag (SYNFLG), is set in WSTAT. Sixteen more timer interrupts later the SYNC is written to the tape and the block is almost finished. One more interrupt is needed to finish the last bit. The write done flag (WRDFLG) is set to indicate that this is the last interrupt for this block. WRDFLG is detected as being set to a 1 on the next interrupt and the transport is stopped. WRFLG in WSTAT is reset and the Good-Completion result code is loaded into the RESULT register before exiting to the foreground task.

All this occurs while the foreground task is testing WRFLG. When WRFLG is cleared, the foreground task "knows" that the background task is finished; BUSY is reset and the result code stored in RESULT is loaded into DBBOUT. The program then returns to the command recognizer.

## READ OPERATION

In the case of the read command, figure 14, the RSTAT register provides the communication between the foreground and background tasks. The read command routine starts out by initializing the registers it requires: the checksum accumulator, CHKSUUM, is cleared; pointers for the circular buffer, DBIN, LBRDY, and LBOUT are set to the start of the buffer; and the bit counter, BITCNT, is set to 8.

The circular buffer has three pointers: LBIN to point to the next free buffer location, LBOUT to point to the next location from which to retrieve data, and LDRDY to trail LBIN by two locations. LBRDY trails LBIN to ensure that the host does not get the received checksum or last SYNC bytes as data. The buffer is empty whenever LDRDY equals LBOUT. The buffer is full whenever LBOUT minus 1 equals LBIN. Data is placed in the buffer by

loading it into the location pointed at by LBIN and then LBIN is incremented. Data is removed from the buffer at the location pointed at by LBOUT and the LBOUT is incremented by one. The data memory locations 20H thru 3FH form the circular buffer. Any pointer increment or decrement operation limits the pointers to this range. (If a pointer at 3FH is incremented, the result wraps around to 20H.)

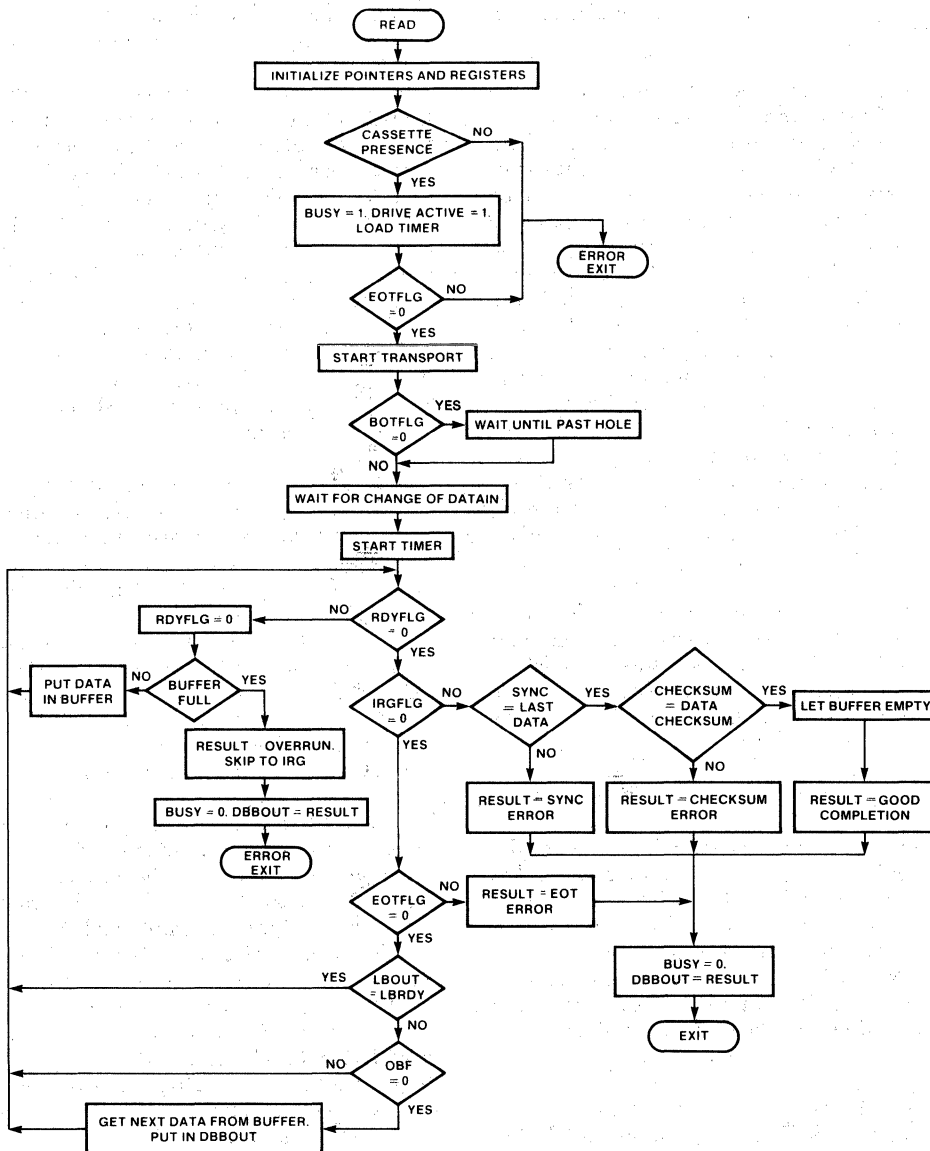
Once the registers have been initialized, the timer is loaded, but not started, with a value that corresponds to 3/4 of a bit-cell time. Next, the EOT test is performed on the EOTFLG in RSTAT. The routine exits with an EOT-while-read error result code if an attempt to read is made while at EOT. If not, the transport is started and the BOTFLG is tested to see if it must move past the clear leader and hole. Once past the clear leader and hole, if necessary, the SYNC-Next-Byte flag (SNBFLG), and the Start flag (SRTFLG), are set in RSTAT. SNBFLG informs the software that the next received byte should be a SYNC. SRTFLG prevents LBRDY from being incremented prematurely.

As soon as a transition from mark (1) to space (0) is detected, the timer is started. The routine enters a loop which tests the data ready flag (RDYFLG), the IRG found flag (IRGFLG), and the EOT detected flag (EOTFLG) in RSTAT. These flags are set by the background task to communicate with the foreground. RDYFLG is set when a character has been assembled and is waiting in the holding register, RDATA. IRGFLG is set when an IRG has been found by the background task. EOTFLG has the same meaning as with the write command; the clear leader at the end of the tape has been found.

If none of these flags are set, the foreground then looks at the circular buffer to see if it contains any data to output to the host. The buffer contains data when LBIN does not equal LBOUT. If these pointers are equal, the buffer is empty and the foreground task just continues to loop. If they are not equal, there is some data left in the buffer. OBF is tested to see if DBBOUT is free to accept more data. If it is free, the character pointed at by LBOUT is transferred to DBBOUT and LBOUT is incremented to the next location. If DBBOUT still contains previously loaded data (OBF set), the foreground continues to test the flags in RSTAT.

When the foreground task finds RDYFLG set, data is available in RDATA. Before transferring this data into the buffer, it first compares LBIN and LBOUT. If LBOUT is one less than LBIN, the buffer is full and no more data can be loaded. This is an error condition; the read operation is aborted and the

# APPLICATIONS



**Figure 14. READ Command Routine Flow Chart**

# APPLICATIONS

transport is moved to the next IRG using the SKIP routine discussed below. Once at the next IRG, BUSY is reset and the Read-Overrun error result code (41H) is placed in DBBOUT. This terminates the read operation and the routine branches back to the command recognizer.

If the buffer is not full, the data is transferred from RDATA to the location pointed to by LBIN. LBIN is incremented and the RDYFLG in RSTAT is reset. LBRDY is also incremented if LBIN has been incremented twice already. (SRTFLG set prevents LBRDY from being incremented. SRTFLG is reset when LBIN is incremented to the second buffer position.) This ensures that LBRDY will point to the last data byte once an IRG is detected. The data is also added to the accumulated checksum, CHKSUM. The foreground then goes back to test the RSTAT flags. When IRGFLG is found set, the background task has found an IRG and stopped the transport. This indicates that the block read is complete. Since the IRG occurs after the checksum and final SYNC characters, these two bytes are in the circular buffer. To test them the foreground task then decrements LBIN to point at the final SYNC and checks if it is a SYNC character. If not, a Bad-Sync2 error result code (43H) is placed in RESULT and the routine branches to the read exit routine. If it is okay, a SYNC is removed from the accumulated checksum. LBIN is decremented again to point to the received checksum. Since this character is also in the accumulated checksum, it is subtracted out. Now the accumulated checksum reflects only the received data so it is compared with the received checksum. If they are equal, the data is presumed good and a Good-Completion result code (00H) is loaded into result. If not, an error has occurred and the RESULT is loaded with the Bad-Checksum result code (44H).

Although the actual read operation is complete with respect to the transport, there may still be data remaining in the buffer of the controller. The read exit routine loops testing LBOUT and LBRDY and transferring data from the buffer into DBBOUT until the buffer is empty. Once the buffer is empty, BUSY is reset and the result code is transferred from RESULT to DBBOUT, completing the read operation.

The timer interrupt routine, RDINT, for the read operation is shown in figure 15. The phase decoding algorithm specifies that the timer start at the beginning transition of the bit cell. It waits for 3/4 of a bit cell before sampling the data input. If the data input is the same as immediately after the beginning transition, the data bit is a 0. If it is different, the

data bit is a 1. The timer interrupt routine compares the present state of the data input to the state immediately following the beginning transition.  $F_0$  stores this value and shifts it into the de-serializing register (DESERL). Once 8 bits have been accumulated, the RDYFLG is set to inform the foreground that a character is complete. This character is then transferred from DESERL to the holding register, RDATA.

After the interrupt routine has sampled and shifted in the bit, it looks for the beginning transition of the

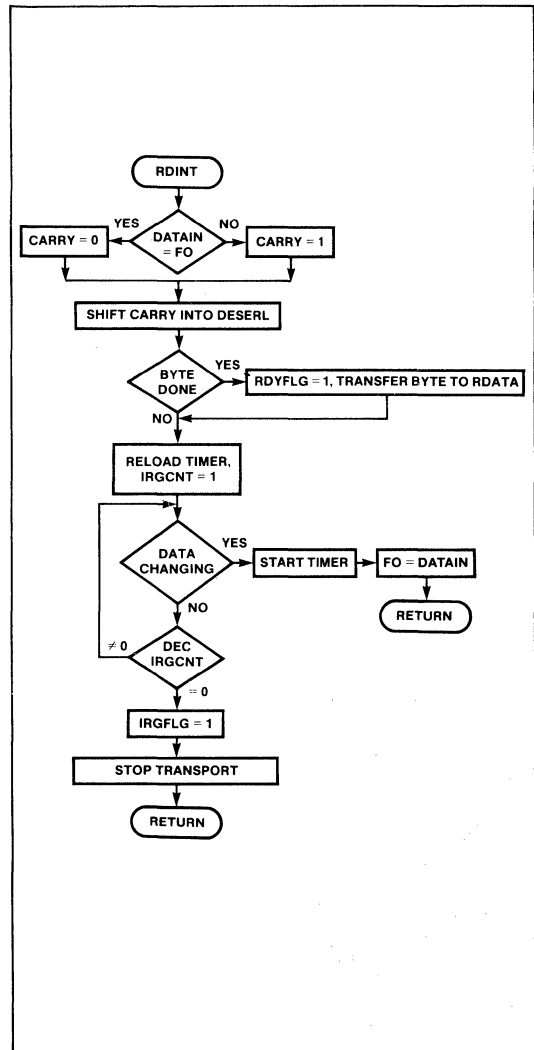


Figure 15. RDINT—Read Timer Interrupt Routine Flow Chart

next bit cell. While looking for this transition, it keeps track of time by decrementing a counter called IRGCNT. If this counter reaches zero, no transition has occurred within a certain amount of time (this application used two bit cell times); this is defined as the beginning of an IRG. When an IRG is found, the transport is stopped and the IRGFLG is set in RSTAT before exiting the interrupt service routine. If a transition is found before the counter times out, the routine exits setting  $F_0$  to the data input state after the transition.  $F_0$  is used for storing the state while in the foreground. As in the write operation, the CLEAR LEADER input is also

tested every interrupt. If an EOT is found, EOTFLG is set and the transport is stopped.

## SKIP OPERATION

The same technique for finding IRGs is used in the SKIP command routine. The SKIP command, figure 16, causes the transport to skip forward or reverse a specified number of IRGs. The number of IRGs to skip is indicated by the byte following the SKIP command byte acceptance (i.e. BUSY has been set and IBF is 0). The SKIP command routine waits, looping on IBF, until the IRG skip count is loaded by

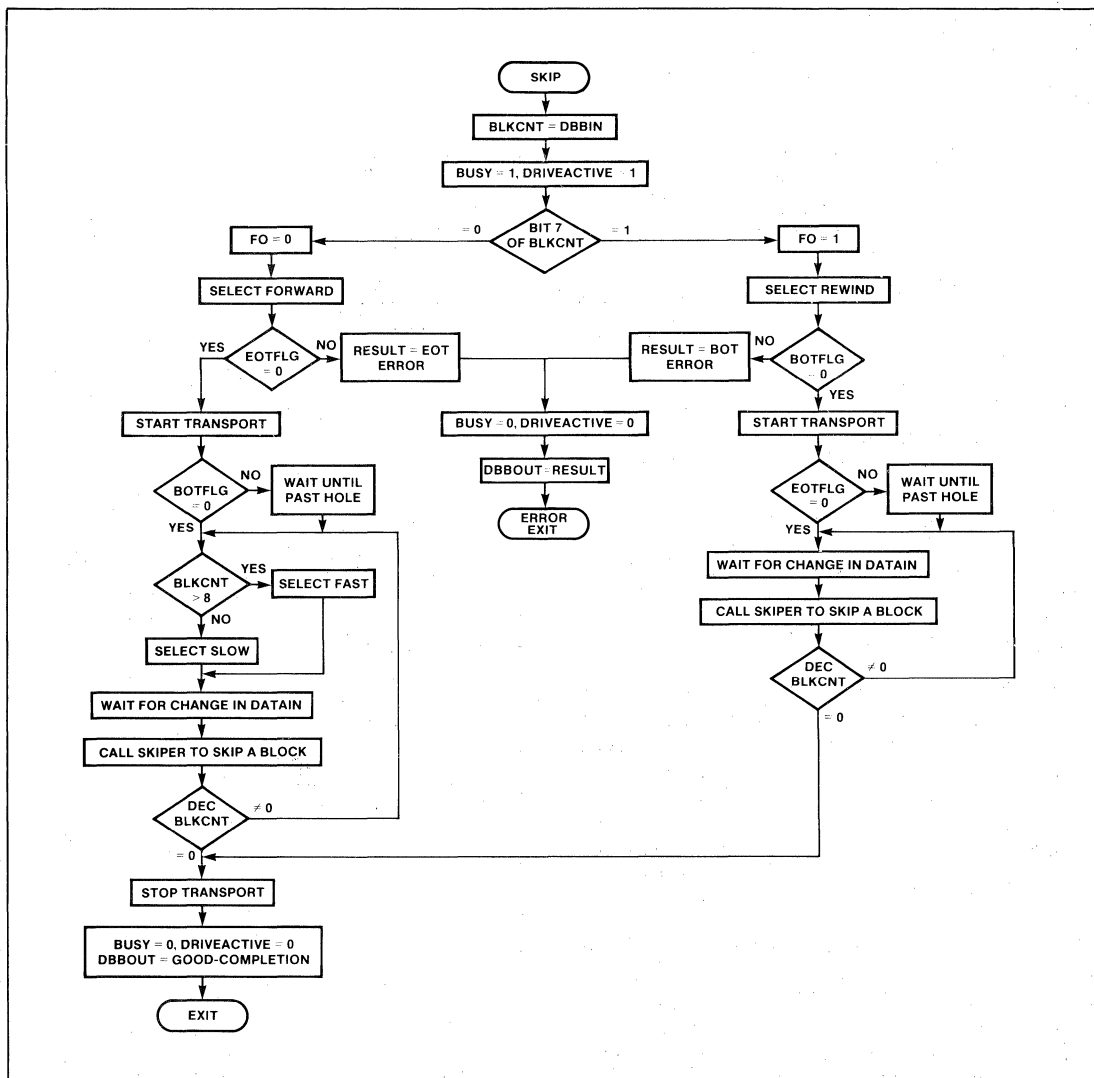


Figure 16. SKIP Command Routine Flow Chart

the host. Then it is transferred from DBBIN to the skip count register, BLKCNT. The usual EOT and BOT tests are performed to ensure it doesn't skip forward when at EOT or reverse when at BOT. The transport is then started in the direction indicated by bit 7 of the BLKCNT value. (This bit is masked off after the initial direction test.)

For reverse skips, the skipping subroutine, SKIPER, is called. It advances the transport to the next IRG using the IRGCNT technique described above. When SKIPER returns, BLKCNT is decremented and tested for zero. If non-zero, SKIPER is called repeatedly until BLKCNT is zero. Once zero, the transport is stopped and BUSY is reset. DBBOUT is loaded with a Good-Completion result code (00H).

When doing forward skips, the software takes advantage of the fact that the transport can recognize IRGs during fast forward. If the BLKCNT is greater than 8, fast forward is selected instead of slow and SKIPER is called. (The IRGCNT value is modified to take into account the faster tape speed.) When SKIPER returns, BLKCNT is decremented and tested both for being less than 8 or equal to zero. Once BLKCNT is less than 8, the slow speed is selected. Once BLKCNT reaches 0, the operation is terminated like the reverse skips. The transport is stopped and BUSY is cleared. DBBOUT is loaded with a Good-Completion result code.

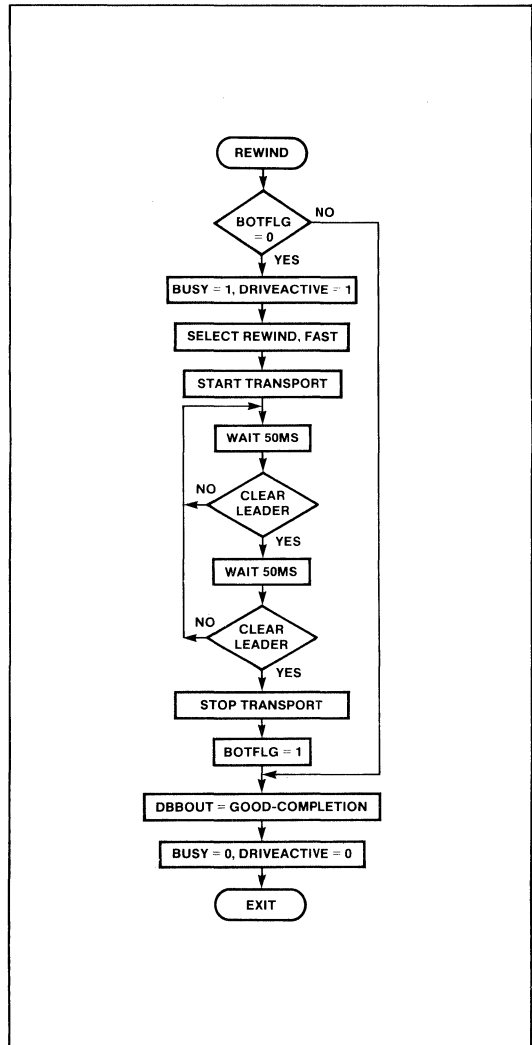
As with both READ and WRITE commands, the clear leader test is made periodically to ensure that no skips are made past the end or beginning of the tape. The appropriate error result code is issued if CLEAR LEADER is found set. RSTAT is loaded with the appropriate EOTFLG bit set.

## REWIND OPERATION

The REWIND command routine, figure 17, simply sets the transport to fast rewind and loops until clear leader is found for greater than 50ms. (The hole at the ends of the tape is guaranteed not to cause the clear leader input to be active for more than 50ms.) Once the tape's clear leader is found; the transport is stopped; BUSY is reset. A Good-Completion result code is loaded into DBBOUT. Also, since the transport is now at the BOT, the BOTFLG is RSTAT is set.

## ABORT OPERATION

The final command is the ABORT command. It does not have a separate flow chart of its own. All other commands monitor IBF periodically during



**Figure 17. REWIND Command Routine Flow Chart**

their execution. If a command is found, the command is compared to the ABORT command code. If it is found, the routine in execution is stopped and BUSY is reset. The Abort-Complete result code is placed in DBBOUT. The aborted routine does ensure that it exits gracefully. An aborted READ or SKIP advances to the next IRG before stopping; WRITE records an IRG before stopping.

## WRAPPING IT UP

The program listing follows in Appendix A. For more information on the UPI-41A family, see the

## APPLICATIONS

---

referenced manuals on the cover of this application note. For those readers who would like to use or modify this program but don't want to type in nearly 1K bytes of code, source files are available through the Intel User's Library, INSITE. (Contact your local Intel sales office for information on INSITE.) A sample of other UPI-41A programs available thru the INSITE library are:

Seiko printer controller  
Olivetti printer controller  
LRC printer controller (8295)  
Sensor matrix controller  
LED display controller  
Combination serial/parallel I/O  
Programmable keyboard/display controller  
GPIB controller (8292)

**APPENDIX**



# APPLICATIONS

ASM48 : F1: DIGCAS. ASM NOOBJECT PRINT(:LP:)

ISIS-II MCS-4B/UPI-41 MACRO ASSEMBLER, V3 0  
DIGITAL CASSETTE CONTROLLER REV 1 0 - 26 MARCH 80

```

LOC  OBJ          LINE          SOURCE STATEMENT
1  *MACROFILE MDD41A TITLE('DIGITAL CASSETTE CONTROLLER REV 1 0 - 26 MARCH 80')
2  ;
3  ; *****
4  ;
5  UPI-41A DIGITAL CASSETTE CONTROLLER FOR THE BRAEMAR CM-600
6  ;
7  ; *****
8  ;
9  ; THIS UPI-41A BASED PROGRAM CONTROLS A BRAEMAR CM-600 MINI-CASSETTE
10 ; THE PROGRAM ALLOWS THE HOST CPU TO SIMPLY ISSUE COMMANDS SPECIFYING
11 ; READ-A-BLOCK, WRITE-A-BLOCK, SKIP FORWARD OR REVERSE N BLOCKS,
12 ; REWIND, AND ABORT. THE UPI-41A HANDLES ALL DATA REQUESTS AND MONITORS
13 ; THE CASSETTE DRIVE FOR ERRORS, EQ WRITING TO THE END-OF-TAPE, ETC
14 ; EACH COMMAND SETS THE CONTROLLER IN THE BUSY CONDITIONS. ONCE THE
15 ; OPERATION IS COMPLETE, THE UPI-41A RESETS IT'S BUSY FLAG AND LOADS THE
16 ; OUTPUT DATA BUFFER WITH A RESULT BYTE WHICH INDICATES THE RESULT
17 ; OF THE REQUESTS OPERATION. THE COMMANDS AND RESULT CODES ARE SHOWN
18 ; IN THE SYSTEMS EQUATES.
19 ;
20 ; THE CONTROLLER USES A MODIFIED PHASE ENCODING WHERE DATA 0'S ARE LONG
21 ; (FULL BIT TIME) CELLS AND DATA 1'S HAVE TRANSITIONS AT THE MID-BIT CELL
22 ; POSITION. WHEN WRITING, ALL BLOCKS ARE PREFACED AND CONCLUDED WITH
23 ; SYNC CHARACTERS (OAAH). A CHECKSUM BYTE IMMEDIATELY PRECEEDS THE
24 ; FINAL SYNC. WHEN READING, THE CONTROLLER TESTS THE VALIDITY OF
25 ; BOTH SYNC CHARACTERS AND THE CHECKSUM
26 ; INTER-RECORD GAPS (IRG) ARE WRITTEN WITH ALL MARK (DATA OUT = 1).
27 ;
28 ; THE WRITE-A-BLOCK OPERATION IS DOUBLE BUFFERED WHILE THE READ-A-BLOCK
29 ; OPERATION USES A 30-CHARACTER CIRCULAR BUFFER TO MINIMIZE CPU
30 ; RESPONSE TIME REQUIREMENTS.
31 ;
32 $EJECT
33 ; *****
34 ;
35 ; REGISTER EQUATES - THE WRITE AND READ/SKIP OPERATIONS ARE DISTINCT
36 ; THEREFORE THE SAME PHYSICAL REGISTER MAY BE USED FOR DIFFERENT
37 ; PURPOSES IN EACH OPERATION. EACH OPERATION USES DIFFERENT REGISTER
38 ; LABELS FOR CLARITY
39 ;
40 ; *****
41 ;
42 ; WRITE - RBO
43 ; -----
44 CNTLSB EQU R0 ; BYTE COUNTER LSB
45 CNTMSB EQU R1 ; BYTE COUNTER MSB
46 CMDSAV EQU R2 ; COMMAND SAVER
47 CHKSUM EQU R3 ; CHECKSUM REGISTER
48 TEMP1 EQU R4 ; TEMPORARY STORAGE
49 ; R5 ; DELAY REGISTER
50 ; R6 ; DELAY REGISTER
0007 51 STAT EQU R7 ; STS IMAGE
52 ;
53 ; WRITE - RB1
54 ; -----
55 ; R0 ; NOT USED
56 ; R1 ; NOT USED
0002 57 RESULT EQU R2 ; RESULT STORAGE
0003 58 WSTAT EQU R3 ; WRITE STATUS REGISTER
0004 59 BITCNT EQU R4 ; WRITE BIT COUNTER
0005 60 SERIAL EQU R5 ; WRITE SERIALIZER
0006 61 TEMPO EQU R6 ; TEMPORARY STORAGE
0007 62 ASAVE EQU R7 ; ACCUMULATOR SAVE
63 ;
64 ; *****
65 ;
66 ; READ/SKIP - RBO
67 ; -----
0000 68 LBOUT EQU R0 ; NEXT BYTE OUTPUT POINTER
0001 69 LBRDY EQU R1 ; NEXT BYTE AVAILABLE POINTER
70 ; R2 ; NOT USED
71 ; CHKSUM EQU R3 ; CHECKSUM REGISTER (SAME FOR WRITE)
0004 72 BLKCNT EQU R4 ; BLOCK COUNTER FOR SKIP
0005 73 BLKTIM EQU R5 ; BLOCK IRG TIMER FOR SKIP
0006 74 BLKSAM EQU R6 ; BLOCK IRG TIMER SAVE
75 ; STAT EQU R7 ; STS IMAGE (SAME FOR WRITE)
76 ;
77 ; READ/SKIP - RB1
78 ; -----
0000 79 LBIN EQU R0 ; NEXT BYTE INPUT POINTER
0001 80 IRCNT EQU R1 ; IRG TICK TIMER
81 ; RESULT EQU R2 ; RESULT STORAGE (SAME FOR WRITE)
0003 82 RSTAT EQU R3 ; READ STATUS REGISTER
83 ; BITCNT EQU R4 ; READ BIT COUNTER (SAME FOR WRITE)
0005 84 DESERL EQU R5 ; READ DE-SERIALIZER
0006 85 RDATA EQU R6 ; READ DATA BUFFER
86 ; ASAVE EQU R7 ; ACCUMULATOR SAVE (SAME FOR WRITE)
87 ;

```

# APPLICATIONS

ISIS-II MCS-4B/UPI-41 MACRO ASSEMBLER, V3.0  
 DIGITAL CASSETTE CONTROLLER REV 1.0 - 26 MARCH 80

```

LOC  OBJ          LINE          SOURCE STATEMENT
      ;
      ; *****
88 ; *****
89 ;
90 ; STATUS REGISTER BIT DEFINITIONS:
91 ; THE MAJOR OPERATIONS, WRITE AND READ, USE THE TIMER TO DETERMINE
92 ; ALL BIT-CELL TIMING AND TO PERFORM THE SERIAL-TO-PARALLEL CONVERSIONS.
93 ; THE TIMER INTERRUPT SERVICE AND MAIN ROUTINES COMMUNICATE VIA
94 ; GENERAL PURPOSE REGISTERS USED AS STATUS REGISTERS.
95 ;
96 ; STAT IS THE STS REGISTER IMAGE SINCE THE UPI CAN'T READ STS DIRECTLY.
97 ; (ONLY THE HIGH ORDER 4-BITS OF STAT ARE USED.)
98 ; THE LOWER 4-BITS ARE NOT USER-DEFINABLE.
99 ;
100 ; *****
101 ;
102 ; WSTAT - WRITE STATUS REGISTER
103 ;
104 ;          WSTATO  CHECKSUM FLAG (CKSFGL) - CHECKSUM BYTE BEING SENT
105 ;             1  SYNC FLAG (SYNFLG) - FINAL SYNC BYTE BEING SENT
106 ;             2  WRITE DONE FLAG (WRDFLG) - FINAL SYNC IS BEING SENT
107 ;                                     (ENSURES LAST BIT IS COMPLETE)
108 ;             3  NOT USED
109 ;             4  NOT USED
110 ;             5  END OF TAPE FLAG (EOTFLG) - EOT WAS FOUND, TAPE IS NOW AT EOT
111 ;             6  BEGINNING OF TAPE FLAG (BOTFLG) - BOT WAS FOUND, TAPE IS NOW AT BOT
112 ;             7  WRITE/READ FLAG (WRFLG) - WRITE OR READ OPERATION IS ACTIVE
113 ;
114 ; RSTAT - READ STATUS REGISTER
115 ;
116 ;          RSTATO  DATA READY FLAG (RDYFLG) - NEXT BYTE IS READY IN RDATA
117 ;             1  SYNC NEXT BYTE FLAG (SNBFLG) - NEXT BYTE SHOULD BE A SYNC
118 ;             2  START FLAG (SRFLG) - BEGINNING OF READ, DON'T INC LBRDY
119 ;                                     UNTIL LBIN=22
120 ;             3  IRG FOUND FLAG (IRGFLG) - IRG WAS FOUND BY TIMER INTERRUPT ROUTINE
121 ;             4  NOT USED
122 ;             5  END OF TAPE FLAG (EOTFLG) - EOT WAS FOUND, TAPE IS NOW AT EOT
123 ;             6  BEGINNING OF TAPE FLAG (BOTFLG) - BOT WAS FOUND, TAPE IS NOW AT BOT
124 ;             7  WRITE/READ FLAG (WRFLAG) - WRITE OR READ OPERATION IS ACTIVE
125 ;
126 ; STAT - STS IMAGE
127 ;
128 ;          STATO   OBF - OUTPUT BUFFER FULL
129 ;             1  IBF - INPUT BUFFER FULL
130 ;             2  FO - GENERAL PURPOSE FLAG (USED INTERNALLY)
131 ;             3  F1 - COMMAND/DATA FLAG
132 ;             4  DRIVE ACTIVE - MOTOR ON
133 ;             5  FILE PROTECT - DRIVE STATUS
134 ;             6  CASSETTE PRESENCE - DRIVE STATUS
135 ;             7  BUSY - CONTROLLER PERFORMING OPERATION
136 ;
137 ; *EJECT
138 ; *****
139 ;
140 ; PORT DEFINITION:
141 ;
142 ; *****
143 ;
144 ; PORT10 - DIRECTION (0-FORWARD, 1-REWIND)
145 ;     11 - MOTION (0-GO, 1-STOP)
146 ;     12 - SPEED (0-FAST, 1-SLOW)
147 ;     13 - READ/WRITE (0-READ, 1-WRITE)
148 ;     14 - CLEAR LEADER (0-OFF LEADER, 1-ON LEADER)
149 ;     15 - FILE PROTECT (0-TAB PRESENT, 1-NO TAB)
150 ;     16 - PRESENCE (0-TAPE IN WITH DOOR CLOSED, 1-NO TAPE)
151 ;     17 - NOT USED
152 ;
153 ; PORT20 - DATA OUT TO CASSETTE (0-SPACE, 1-MARK)
154 ;     21 - DRIVE ACTIVE LED (0-ON, 1-OFF)
155 ;     22 - NOT USED
156 ;     23 - NOT USED
157 ;     24 - OBF INTERRUPT OUTPUT
158 ;     25 - IBF/ INTERRUPT OUTPUT
159 ;     26 - NOT USED
160 ;     27 - NOT USED
161 ;
162 ; TEST1 - DATA IN FROM CASSETTE
163 ;
164 ; *EJECT
    
```

# APPLICATIONS

ISIS-II MCS-48/UPI-41 MACRO ASSEMBLER, V3.0  
 DIGITAL CASSETTE CONTROLLER REV 1.0 - 26 MARCH 80

LOC	OBJ	LINE	SOURCE STATEMENT
		165	*****
		166	;
		167	SYSTEM EQUATES:
		168	;
		169	*****
		170	;
		171	WRITE SYSTEM EQUATES:
		172	;
0001		173	CKSFLG EQU 01H ;CHECKSUM FLAG IN WRITE STATUS
0002		174	SYNFLG EQU 02H ;SYNC FLAG IN WRITE STATUS
0004		175	WRDFLG EQU 04H ;WRITE DONE FLAG IN WRITE STATUS
0020		176	EDTFLG EQU 20H ;EDT FLAG
0040		177	BOTFLG EQU 40H ;BOT FLAG
0080		178	WRFLG EQU 80H ;READ/WRITE FLAG IN WRITE STATUS
0008		179	WRCNT EQU 08H ;WRITE BIT CONSTANT
FFFC		180	WRTIM EQU -4H ;WRITE TIMER CONSTANT
		181	;
		182	PORT EQUATES:
		183	;
0001		184	REWIND EQU 01H ;DIRECTION MASKS
00FE		185	FORWD EQU 0FEH
0002		186	STP EQU 02H ;START/STOP MASKS
00FD		187	SRT EQU 0FDH
0004		188	SLOW EQU 04H ;SPEED MASKS
00FB		189	FAST EQU 0FBH
0008		190	WR EQU 08H ;WRITE/READ MASKS
00F7		191	RD EQU 0F7H
0001		192	DOHI EQU 01H ;DATA OUTPUT TO DRIVE MASKS
00FE		193	DOLW EQU 0FEH
0002		194	DAOFF EQU 02H ;DRIVE ACTIVE LED MASKS
00FD		195	DAON EQU 0FDH
		196	;
		197	READ SYSTEM EQUATES:
		198	;
0001		199	RDYFLG EQU 01H ;DATA READY FLAG IN READ STATUS
0002		200	SNBFLG EQU 02H ;SYNC NEXT BYTE FLAG IN READ STATUS
0004		201	STRFLG EQU 04H ;START INC READY POINTER FLAG IN READ STATUS
0008		202	IRGFLG EQU 08H ;IRG FOUND FLAG IN READ STATUS
0008		203	RDCNT EQU 08H ;READ TIMER CONSTANT
FFFA		204	RDTIM EQU -6H ;READ BIT CONSTANT
		205	;
		206	STS REGISTER EQUATES:
		207	;
0080		208	BUSY EQU 80H ;BUSY BIT
0010		209	DRACT EQU 10H ;DRIVE ACTIVE BIT
0040		210	TAPIN EQU 40H ;TAPE IN DRIVE BIT
0020		211	FILPRT EQU 20H ;FILE PROTECT BIT
		212	*EJECT
		213	GENERAL RESULT CODES
		214	;
0001		215	ABTCMP EQU 01H ;ABORT COMPLETE CODE
0000		216	GOOD EQU 00H ;GOOD RESULT CODE
0002		217	CMDERR EQU 02H ;COMMAND ERROR CODE
0003		218	NTAPE EQU 03H ;NO TAPE ERROR CODE
0004		219	NWR EQU 04H ;FILE PROTECT ERROR CODE
		220	;
		221	WRITE RESULT CODES
		222	;
0081		223	UNDERW EQU 81H ;UNDERRUN ERROR CODE
0082		224	WCMDER EQU 82H ;COMMAND/DATA ERROR CODE
0083		225	EDTERR EQU 83H ;EDT ERROR CODE
		226	;
		227	READ RESULT CODES
		228	;
0041		229	OVERUN EQU 41H ;UNDERRUN CODE FOR BUFFER
0042		230	SYNC1 EQU 42H ;BAD SYNC1 ERROR CODE
0043		231	SYNC2 EQU 43H ;BAD SYNC2 ERROR CODE
0044		232	BADCHS EQU 44H ;BAD CHECKSUM ERROR CODE
0045		233	RCMDER EQU 45H ;COMMAND/DATA ERROR CODE
0046		234	REOTER EQU 46H ;EDT AT READ ERROR CODE
0047		235	SKPEOT EQU 47H ;EDT AT SKIP ERROR CODE
0048		236	SKPBOT EQU 48H ;BOT AT RSKIP ERROR CODE
		237	;
		238	MISC EQUATES:
		239	;
00AA		240	SYNC EQU 0AAH ;SYNC BYTE
0033		241	SLWIRG EQU 51D ;SLOW IRG COUNT CONSTANT - NO TRANSITION IN 2 BIT TIMES
0020		242	FASTIRG EQU 32D ;FAST IRG COUNT CONSTANT
0020		243	RWDIRG EQU 32D ;REWIND IRG COUNT FOR SKIP
		244	;
		245	COMMANDS
		246	;
0005		247	ABORT EQU 05H ;ABORT COMMAND
0001		248	RDCMD EQU 01H ;READ FROM TAPE COMMAND
0002		249	WRCMD EQU 02H ;WRITE TO TAPE COMMAND
0004		250	RWCMD EQU 04H ;REWIND COMMAND
0003		251	SKCMD EQU 03H ;SKIP BLOCK COMMAND
0000		252	RESCMD EQU 00H ;RESET COMMAND
		253	;
		254	*EJECT

# APPLICATIONS

ISIS-II MCS-48/UPI-41 MACRO ASSEMBLER, V3.0  
DIGITAL CASSETTE CONTROLLER REV 1.0 - 26 MARCH 80

```

LOC  OBJ          LINE      SOURCE STATEMENT
255 ; *****
256 ;
257 ; START OF PROGRAM - JUMPS FOR COLD START (RESET) AND TIMER INTERRUPTS
258 ;
259 ; *****
260 ;
0000 0409      261 RESET:  JMP      BEGIN          ; JUMP OVER TIMER VECTOR LOCATION
262 ;
0007          263          ORG      7H            ; TIMER INTERRUPT VECTOR LOCATION
264 ;
0007 6400      265 TIMINT: JMP      INT           ; JUMP TO TIMER INTERRUPT SERVICE ROUTINE
266 ;
267 ; *****
268 ;
269 ; PROGRAM START - INITIALIZE STATUS REGISTERS, DRIVE OUTPUTS, AND
270 ; WAIT FOR A COMMAND.
271 ;
272 ; *****
273 ;
0009 27        274 BEGIN:  CLR      A              ; INITIALIZE THINGS
000A B5        275          CLR      FO
000B C5        276          SEL      RBO
000C AF        277          MOV      STAT, A        ; CLEAR STS IMAGE
000D 90        278          MOV      STS, A         ; CLEAR STS
000E D5        279          SEL      RB1
000F AB        280          MOV      WSTAT, A       ; CLEAR STATUS
0010 B906      281          ORL      P1, #STP OR SLOW ; STOP DRIVE AND SELECT SLOW FOR STARTERS
0012 99F4      282          ANL      P1, #F0RWD AND RD ; SELECT FORWARD AND READ
0014 BA02      283          ORL      P2, #DAOFF    ; TURN OFF DRIVE ACTIVE LED
0016 F5        284          EN      FLAGS          ; ENABLE FLAG INTERRUPT OUTPUTS
285 ;
286 ; COMMAND RECOGNIZER MAIN LOOP
287 ;
0017 C5        288 B1:    SEL      RBO              ; COMMAND PROCESSING IN RBO
0018 D61F      289          JNIBF     B2            ; TEST IF IBF INPUT
001A 7623      290          JF1      CMDIN        ; YES THERE IS AN INPUT, SO TEST IF ITS A COMMAND
001C 22        291          IN      A, DBB         ; NOPE, ITS DATA SO IGNORE IT
001D 0417      292          JMP      B1             ; JUST GO BACK TO TEST IBF
001F 14AD      293 B2:    CALL     STSUP          ; NO INPUT, UPDATE STS WITH DRIVE STATUS
0021 0417      294          JMP      B1             ; GO BACK TO TEST IBF
295 ;
296 ; *****
297 ;
298 ; COMMAND PROCESSOR - TESTS VALIDITY OF INPUT AND BRANCHES TO THE APPROPRIATE
299 ; ROUTINE.  ILLEGAL COMMANDS ARE FLAGGED AS COMMAND ERRORS.
300 ;
301 ; *****
302 ;
0023 FF        303 CMDIN:  MOV      A, STAT        ; GET STS IMAGE
0024 4380      304          ORL      A, #BUSY       ; SET BUSY FOR ALL COMMAND INPUTS
0026 AF        305          MOV      STAT, A        ; RESTORE IMAGE
0027 90        306          MOV      STS, A         ; UPDATE STS
0028 22        307          IN      A, DBB         ; READ COMMAND FROM DBBIN
0029 AA        308          MOV      CMDSAV, A       ; SAVE IT IN CMDSAV
002A BC06      309          MOV      TEMP1, #6H      ; INITIALIZE ILLEGAL COMMAND COUNTER
002C FA        310 CMDIN1: MOV      A, CMDSAV      ; GET COMMAND FROM CMDSAV
002D 17        311          INC      A
002E DC        312          XRL      A, TEMP1       ; TEST IF VALID
002F C63A      313          JZ      CMDIN2        ; YES, INDIRECT JUMP TO IT
0031 EC2C      314          DJNZ     TEMP1, CMDIN1  ; NO MATCH YET, TRY AGAIN
315 ;
316 ;
317 ;
318 ;
319 ;
320 ;
321 ;
0033 54C4      317 CMDIN3: CALL     NDRACT        ; RESET DRACT AND BUSY (DRACT WAS NEVER SET)
0035 2302      318          MOV      A, #CMDERR      ; COMMAND ERROR CODE
0037 02        319          OUT      DBB, A         ; OUTPUT ERROR CODE
0038 0417      320          JMP      B1             ; GO BACK TO TEST FOR IBF
321 ;
003A FA        322 CMDIN2: MOV      A, CMDSAV      ; IT'S A GOOD COMMAND, GET IT FROM CMDSAV
003B 033E      323          ADD      A, # (LOW CMDJMP) ; ADD OFFSET
003D B3        324          JMPMP    @A                ; INDIRECT JUMP TO THE COMMAND ROUTINE THRU TABLE
325 ; COMMAND JUMP TABLE
326 ;
003E 44        327 CMDJMP: DB      (LOW RESJMP)
003F 46        328          DB      (LOW REDJMP)
0040 48        329          DB      (LOW WRTJMP)
0041 4A        330          DB      (LOW SKPJMP)
0042 4C        331          DB      (LOW REWJMP)
0043 44        332          DB      (LOW ARTJMP)
333 ;
334 ;
0044 044E      334 ARTJMP: JMP      RESCOM
0046 2400      335 RESJMP: JMP      RESCOM
0048 0455      336 REDJMP: JMP      READ
004A 4414      337 WRTJMP: JMP      WRITE
004C 4494      338 SKPJMP: JMP      SKIP
004E          339 REWJMP: JMP      REWIND
340 ;
341 ; IT'S A ABORT COMMAND WHILE IN COMMAND RECOGNIZER LOOP
342 ;
004E 54C4      343 RESCOM: CALL     NDRACT        ; RESET BUSY AND DRACT (DRACT NEVER WAS SET)
0050 2301      344          MOV      A, #ABTCMP      ; GET ABORT COMPLETE CODE
0052 02        345          OUT      DBB, A         ; OUTPUT IT
0053 0400      346          JMP      RESET          ; GO START OVER
347 ;
348 ; EJECT

```

# APPLICATIONS

ISIS-II MCS-48/UPI-41 MACRO ASSEMBLER, V3.0  
DIGITAL CASSETTE CONTROLLER REV 1.0 - 26 MARCH 80

LOC	OBJ	LINE	SOURCE STATEMENT
		349	*****
		350	;
		351	WRITE TO TAPE ROUTINE
		352	;
		353	*****
		354	;
0055	C5	355	WRITE: SEL RBO
0056	B5	356	CLR FO ;CLEAR INT COUNT FLAG
0057	65	357	STOP TCNT ;BE SURE THAT THE TIMER IS STOPPED
0058	35	358	DIS TCNTI ;DISABLE TIMER INTS
0059	1659	359	CLRTF: JTF CLRTF ;BE SURE THAT THE TIMER FLAG IS CLEARED
005B	D65B	360	WR1: JNIBF WR1 ;WAIT FOR BYTE COUNT LSB
005D	22	361	IN A, DBB ;READ COUNT LSB FROM DBBIN
005E	7633	362	JF1 CMDIN3 ;TEST IF COMMAND - ERROR
0060	AB	363	MOV CNTLSB, A ;IT'S DATA SO STORE IT AWAY
0061	D661	364	WR2: JNIBF WR2 ;WAIT FOR BYTE COUNT MSB
0063	22	365	IN A, DBB ;READ COUNT MSB FROM DBBIN
0064	7633	366	JF1 CMDIN3 ;TEST IF COMMAND - ERROR
0066	A9	367	MOV CNTMSB, A ;IT'S DATA SO STORE IT AWAY
0067	FB	368	MOV CNTLSB ;GET COUNT LSB
0068	17	369	INC A ;INC IT TO ACCOUNT FOR SYNC
0069	AB	370	MOV CNTLSB, A ;SAVE IT
006A	966D	371	JNZ NINMSB ;NO OVERFLOW, DON'T INC COUNT MSB
006C	19	372	INC CNTMSB ;OVERFLOW, SO INC COUNT MSB
006D	09	373	NINMSB: IN A, P1 ;GET DRIVE STATUS
006E	D2C5	374	JB6 DRIVER ;TEST IF NO TAPE
0070	B2C5	375	JB5 DRIVER ;TEST IF FILE PROTECTED
		376	;
		377	MOV CHKSUM, #00H ;CLEAR CHECKSUM REGISTER
0072	BB00	378	SEL RB1
0075	8C08	379	MOV BITCNT, #WRCNT ;INITIALIZE WRITE BIT COUNTER
0077	BDAA	380	MOV SERIAL, #SYNC ;LOAD SYNC INTO SERIAL FOR 1ST BYTE
0079	23FC	381	MOV A, #WRTIM ;GET WRITE TIMER CONSTANT (1/2 CELL TIME)
007B	62	382	MOV T, A ;LOAD TIMER BUT DON'T START IT YET
007C	FB	383	MOV A, WSTAT ;GET WRITE STATUS
007D	B2A9	384	JB5 WEDTER ;IF EOTFLG SET, STILL AT END OF TAPE - ERROR
007F	C5	385	SEL RB0
0080	54BC	386	CALL DRACTS ;NOT AT EOT SO SET DRIVE ACTIVE AND CONTINUE
0082	D5	387	SEL RB1
0083	890E	388	ORL P1, #WR OR SLOW OR STP ;SETUP PORT FOR SLOW WRITE
0085	99FC	389	ANL P1, #SRT AND FORWD ;START DRIVE IN FORWARD
0087	FB	390	MOV A, WSTAT ;GET WRITE STATUS AGAIN
0088	37	391	CPL A ;COMP FOR 0 TEST
0089	D2BD	392	JB6 WR3 ;TEST BOTFLG - WRITE OVER HOLE IF SET
008B	54DC	393	CALL PASHOL ;GET OFF CLEAR LEADER AND PAST HOLE IN TAPE
008D	BB80	394	WR3: MOV WSTAT, #B0H ;SETUP WRITE STATUS WITH WRFLG SET
008F	C5	395	SEL RBO
0090	14C1	396	CALL DEL150 ;WAIT 450 MS IRG BEFORE WRITING DATA
0092	14C1	397	CALL DEL150
0094	14C1	398	CALL DEL150
0096	55	399	STR T ;START TIMER
0097	25	400	EN TCNTI ;ENABLE TIMER INTERRUPTS
		401	;
		402	TIMER INTERRUPT ROUTINE DOES ALL THE WORK SO WAIT UNTIL IT RESETS WRFLG
		403	;
009B	C5	404	WR4: SEL RBO
0099	14AD	405	CALL STSUP ;UPDATE STS WHILE WAITING
009B	D5	406	SEL RB1
009C	FB	407	MOV A, WSTAT ;GET WRITE STATUS
009D	F29B	408	JB7 WR4 ;TEST IF WRITE DONE (WRFLG RESET)
		409	;
		410	WRFLG IS RESET SO WRITE OPERATION MUST BE COMPLETE - OUTPUT RESULT
		411	;
009F	C5	412	WR5: SEL RBO
00A0	54C4	413	CALL NDRACT ;RESET DRACT AND BUSY
00A2	D5	414	SEL RB1
00A3	FA	415	MOV A, RESULT ;GET RESULT CODE
00A4	02	416	OUT DBB, A ;OUTPUT IT
00A5	14C1	417	CALL DEL150 ;WAIT FOR DRIVE TO STOP
		418	;
		419	;
00A7	0417	419	JMP B1 ;FULLY BEFORE ACCEPTING NEW COMMAND
		420	;
		421	;
		421	TAPE IS AT EOT WHEN WRITE COMMAND ISSUED - EXIT WITH ERROR
		422	;
00A9	BAB3	423	WEDTER: MOV RESULT, #EOTERR ;EOT ERROR RESULT CODE
00AB	049F	424	JMP WR5 ;GO RESET BUSY AND OUTPUT RESULT
		425	;
		426	;
		426	EJECT

# APPLICATIONS

ISIS-II MCS-4B/UPI-41 MACRO ASSEMBLER, V3.0  
DIGITAL CASSETTE CONTROLLER REV 1.0 - 26 MARCH 80

LOC	OBJ	LINE	SOURCE STATEMENT
		427	;*****
		428	;
		429	;STS UPDATE SUBROUTINE - UPDATES THE CASSETTE PRESENCE AND FILE PROTECT
		430	;BIT IN STS (ENTER AND EXIT IN RBO)
		431	;
		432	;*****
		433	;
00AD	FF	434	STSUP: MOV A,STAT ;GET STS IMAGE
00AE	4360	435	ORL A,#TAPIN OR FILPRT ;SET BOTH PRESENCE AND FILE PROTECT
00B0	AF	436	MOV STAT,A ;RESTORE IMAGE
00B1	09	437	IN A,P1 ;READ INPUT
00B2	439F	438	ORL A,#NOT(TAPIN OR FILPRT) ;SET BITS TO CORRECT STATE
00B4	5F	439	ANL A,STAT
00B5	AF	440	MOV STAT,A ;RESTORE IMAGE
00B6	90	441	MOV STS,A ;UPDATE STS
00B7	83	442	RET
		443	;
		444	;*****
		445	;
		446	;DELAY ROUTINES- ENTER/EXIT IN RBO
		447	;
		448	;*****
		449	;
00B8	BD24	450	DEL50: MOV R5,#36D ;50MS DELAY ROUTINE
00BA	BEFF	451	DEL1: MOV R6,#OFFH
00BC	EEDC	452	DEL2: DJNZ R6,DEL2
00BE	EDBA	453	DJNZ R5,DEL1
00C0	83	454	RET
		455	;
00C1	BD6C	456	DEL150: MOV R5,#108D ;150MS DELAY ROUTINE
00C3	04BA	457	JMP DEL1
		458	;
		459	;*****
		460	;
		461	;DRIVE ERROR EXIT - NO TAPE OR FILE IS PROTECTED FOR WRITE
		462	;
		463	;*****
		464	;
00C5	C5	465	DRIVER SEL RBO
00C6	54C4	466	CALL NDRACT ;RESET DRACT AND BUSY
00CB	09	467	IN A,P1 ;READ DRIVE STATUS
00C9	D2D0	468	JB6 NT ;TEST IF TAPE IS THERE
00CB	2304	469	MOV A,#NWR ;TAPE IS THERE SO ERROR MUST BE FILE PROTECT
00CD	02	470	DR1: OUT DBB,A ;OUTPUT ERROR CODE
00CE	0417	471	JMP B1 ;RETURN TO COMMAND LOOP
00D0	2303	472	NT: MOV A,#NTAPE ;NO TAPE ERROR
00D2	04CD	473	JMP DR1
		474	;
		475	;*****
		476	;
		477	;READ ERROR WITH ADVANCE TO IRG BEFORE STOPPING DRIVE.
		478	;WAIT FOR OBF TO BE FREE BEFORE RESETTING BUSY THEN OUTPUT RESULT.
		479	;RDERR3 LABEL IS EXIT POINT FOR OTHER ROUTINES NEEDING TO WAIT FOR
		480	;OBF TO BE FREE BEFORE OUTPUTTING RESULT.
		481	;ROUTINE EXITS IN RBO.
		482	;
		483	;*****
		484	;
00D4	C5	485	RDERR: SEL RBO
00D5	BC01	486	MOV BLKCNT,#1H ;SET SKIP COUNTER TO ADVANCE TO NEXT IRG
00D7	BD33	487	MOV BLKTIM,#SLWIRG ;SETUP IRG COUNTER
00D9	BE33	488	MOV BLKSAV,#SLWIRG
00DB	5400	489	CALL SKIPER ;DO SKIP TO NEXT IRG
00DD	F6F1	490	JC RDERR2 ;TEST IF EOT FOUND
00DF	8902	491	RDERR3: ORL P1,#STP ;STOP DRIVE WHEN DONE
00E1	8A02	492	ORL P2,#DAOFF ;TURN OFF DRIVE ACTIVE LED
00E3	D6E7	493	RDERR4: JNIBF RDERR5 ;TEST IBF WHILE WAITING FOR OBF
00E5	04F8	494	JMP RDERR6 ;IBF SET - GO TEST INPUT
00E7	84E3	495	RDERR5: JOBF RDERR4 ;TEST OBF, LOOP IF 1, CONTINUE IF 0
00E9	C5	496	SEL RBO
00EA	54C4	497	CALL NDRACT ;RESET DRACT AND BUSY
00EC	D5	498	SEL RB1
00ED	FA	499	MOV A,RESULT ;GET RESULT
00EE	02	500	OUT DBB,A ;OUTPUT RESULT
00EF	0417	501	JMP B1 ;GO BACK TO COMMAND LOOP
		502	;
00F1	D5	503	RDERR2: SEL RB1 ;EOT FOUND WHILE SKIPPING
00F2	BA46	504	MOV RESULT,#REOTER ;RESET RESULT VALUE TO EOT ERROR
00F4	BB20	505	MOV RSTAT,#EOTFLG ;SET EOTFLG IN RSTAT
00F6	04DF	506	JMP RDERR3 ;GO OUTPUT NEW RESULT
		507	;
00FB	22	508	RDERR6: IN A,DBB ;READ INPUT
00F9	D305	509	XRL A,#ABORT ;TEST IF ABORT
00FB	96E7	510	JNZ RDERR5 ;IGNORE IT IF NOT
00FD	044E	511	JMP RESCOM ;IT'S AN ABORT, GO RESET
		512	;
		513	;EJECT

# APPLICATIONS

ISIS-II MCS-48/UPI-41 MACRO ASSEMBLER, V3.0  
DIGITAL CASSETTE CONTROLLER REV 1.0 - 26 MARCH 80

```

LOC  OBJ      LINE      SOURCE STATEMENT
-----
0100
514      ORG      100H
515 ;
516 ;*****
517 ;
518 ;READ FROM TAPE ROUTINE
519 ;
520 ;*****
521 ;
0100 C5    522 READ:  SEL      RBO
0101 65    523      STOP    TCNT      ;BE SURE THE TIMER IS STOPPED
0102 35    524      DIS     TCNTI     ;DISABLE TIMER INTS
0103 1603  525 RCLRTF: JTF     RCLRTF   ;BE SURE THE TIMER FLAG IS CLEARED
0105 85    526      CLR     FO         ;CLEAR LAST DATA FLAG
0106 2320  527      MOV     A,#20H     ;GET POINTER START LOCATION
0108 AB    528      MOV     LBDOUT,A   ;INITIALIZE LBDOUT
0109 A9    529      MOV     LBRDY,A   ;INITIALIZE LBRDY
010A BB00  530      MOV     CHKSUM,#00H ;CLEAR CHECKSUM LOCATION
010C D5    531      SEL     RB1
010D AB    532      MOV     LBIN,A      ;INITIALIZE LBIN
010E BC0B  533      MOV     BITCNT,#RDCNT ;INITIALIZE READ BIT COUNTER
0110 09    534      IN      A,P1      ;READ DRIVE STATUS
0111 D24F  535      JB6     DRIVJ     ;TEST IF TAPE IS THERE TO READ
0113 C5    536      SEL     RBO
0114 548C  537      CALL    DRACTS     ;SET DRIVE ACTIVE
0116 23FA  538      MOV     A,#RDTIM    ;GET READ TIMER CONSTANT (3/4 CELL TIME)
0118 62    539      MOV     T,A         ;LOAD TIMER BUT DON'T START IT YET
0119 25    540      EN     TCNTI     ;ENABLE TIMER INTERRUPTS
011A D5    541      SEL     RB1
011B FB    542      MOV     A,RSTAT    ;GET READ STATUS
011C B24B  543      JB5     REOT      ;TEST IF AT EOT - ERROR IF SO
011E 8904  544      ORL     P1,#SLOW   ;SELECT SLOW
0120 99F4  545      ANL     P1,#RD AND FORWD AND SRT ;START DRIVE, FORWARD AND READ
0122 37    546      CPL     A         ;COMP A FOR 0 TEST (STILL HAVE RSTAT)
0123 D227  547      JB6     RD1      ;TEST FOR AT BOT, IF NOT JUST LOOK FOR MARK
0125 54DC  548      CALL    PASHOL    ;IF BOT, WAIT UNTIL PAST CLEAR LEADER AND HOLE
0127 BB06  549 RD1:  MOV     RSTAT,#06H   ;SETUP READ STATUS - SNBFLG AND STRFLG SET
0129 14C1  550      CALL    DEL150    ;LET DRIVE START UP
012B 462B  552 RD1A: JNT1     RD1A    ;AND WAIT OVER WRITE STOP LOCATION
012D 562D  553 RD2:  JT1      RD2     ;WAIT FOR MARK
012F 55    554      STRT    T         ;START TIMER
555 ;
556 ;LOOP START - LOOK FOR READ STATUS FLAGS BEING SET BY TIMER INTERRUPT ROUTINE
557 ;
0130 D5    558 RD3:  SEL     RB1
0131 D635  559      JNIBF   RD4      ;TEST FOR IBF EVEN WHEN READING
0133 24D5  560      JMP     RDIBF    ;INPUT DURING READ - GO TEST IT
0135 FB    561 RD4:  MOV     A,RSTAT    ;GET READ STATUS
0136 1251  562      JBO     GETDAT    ;TEST DATA READY FLAG (RDYFLG)
0138 7291  563      JRB     IRGFND   ;TEST IRG FLAG (IRGFLG)
013A B24B  564      JB5     REOT      ;EOT FOUND DURING READ (EOTFLG SET) - ERROR
565 ;
566 ;NOTHING FROM TIMER INTERRUPT ROUTINE SO GO HANDLE CIRCULAR BUFFER
567 ;
013C C5    568      SEL     RBO
013D F9    569      MOV     A,LBRDY    ;GET READY POINTER
013E DB    570      XRL     A,LBDOUT  ;COMPARE TO OUT POINTER
013F C630  571      JZ      RD3      ;EMPTY IF THE SAME SO JUST LOOP
572 ;
0141 B630  573      JDBF   RD3      ;NOT EMPTY SO SEE IF NEXT BYTE CAN BE OUTPUT
574 ;
0143 FO    574      MOV     A,@LBDOUT  ;TEST DBBOUT - FULL, LOOP
0144 02    575      OUT     DBB,A     ;DBBOUT FREE - GET DATA
0145 FB    576      MOV     A,LBDOUT  ;OUTPUT IT
0146 54CC  577      CALL    BUMPIT    ;GET OUT POINTER
0148 AB    578      MOV     LBDOUT,A  ;BUMP POINTER
0149 2430  579      JMP     RD3      ;RETURN IT
580 ;
581 ;TAPE AT EOT WHEN READ COMMAND ISSUED - ERROR
582 ;
014B BA46  583 REOT:  MOV     RESULT,#REOTER ;EOT AT READ ERROR CODE
014D 04DF  584      JMP     RDERR3    ;EXIT
585 ;
586 ;OUT OF PAGE JUMP FOR DRIVE ERROR
587 ;
014F 04C5  588 DRIVJ: JMP     DRIVER
589 ;
590 ;TIMER ROUTINE FLAGGED DATA IS READY
591 ;
0151 53FE  592 GETDAT: ANL     A,#NOT RDYFLG ;RESET DATA READY FLAG (RDYFLG)
0153 AB    593      MOV     RSTAT,A    ;RESTORE READ STATUS
0154 3282  594      JBI     SNBTST    ;TEST IF DATA SHOULD BE SYNC (SNBFLG SET)
0156 C5    595      SEL     RBO       ;NO, TRY TO PUT IN BUFFER
0157 FB    596      MOV     A,LBDOUT  ;GET OUT POINTER
0158 D5    597      SEL     RB1
0159 54D3  598      CALL    DUMPIT    ;DUMP IT FOR FULL TEST
015B DB    599      XRL     A,LBIN    ;COMPARE IT TO IN POINTER
015C 7662  600      JNZ     NOFULL    ;IF NOT SAME, THEN BUFFER ISN'T FULL
015E BA41  601      MOV     RESULT,#OVERRUN ;BUFFER IS FULL SO OVERRUN ERROR CODE
0160 04D4  602      JMP     RDERR     ;GO EXIT FROM ERROR, SKIP TO NEXT IRG
0162 FE    603 NOFULL: MOV     A,RDATA ;BUFFER ISN'T FULL SO GET DATA FROM HOLDING
0163 C5    604      SEL     RBO
0164 68    605      ADD     A,CHKSUM  ;ADD IT TO CHECKSUM
0165 AB    606      MOV     CHKSUM,A  ;RESTORE CHECKSUM
0166 D5    607      SEL     RB1

```

# APPLICATIONS

ISIS-II MCS-48/UPI-41 MACRO ASSEMBLER, V3.0  
DIGITAL CASSETTE CONTROLLER REV 1.0 - 26 MARCH 80

LOC	OBJ	LINE	SOURCE STATEMENT	
0167	FE	608	MOV A, RDATA	;GET DATA AGAIN
0168	A0	609	MOV @LBIN, A	;PUT IT IN BUFFER
0169	FB	610	MOV A, LBIN	;GET IN POINTER
016A	54CC	611	CALL BUMPIT	;BUMP IT
016C	AB	612	MOV LBIN, A	;RETURN IT
016D	FB	613	MOV A, RSTAT	;GET READ STATUS
016E	5277	614	JB2 LBTST	;TEST IF LBRDY SHOULD BE
		615		;BUMPED TOO (SRTFLG RESET)
0170	C5	616	SEL RBO	
0171	F9	617	MOV A, LBRDY	;SRTFLG IS RESET SO GET LBRDY
0172	54CC	618	CALL BUMPIT	;BUMP IT
0174	A9	619	MOV LBRDY, A	;RETURN IT
0175	2430	620	JMP RD3	;GO BACK TO LOOK FOR DATA
		621		
		622	;START FLAG IS SET - SEE IF LBIN HAS ADVANCED FAR ENOUGH TO START INC LBRDY	
		623		
0177	FB	624	LBTST: MOV A, LBIN	;GET IN POINTER
0178	D322	625	XRL A, #22H	;TEST IF READY POINTER SHOULD BE BUMPED
017A	9630	626	JNZ RD3	;NO, GO BACK FOR DATA
017C	FB	627	MOV A, RSTAT	;YES, GET READ STATUS
017D	93FB	628	ANL A, #NOT STRFLG	;RESET START FLAG
017F	AB	629	MOV RSTAT, A	;RESTORE STATUS
0180	2430	630	JMP RD3	;GO BACK TO LOOK FOR DATA
		631		
		632	;DATA SHOULD BE SYNC - TEST IT	
		633		
0182	FE	634	SNBTST: MOV A, RDATA	;GET DATA
0183	D3AA	635	XRL A, #SYNC	;COMPARE TO SYNC
0185	C6BB	636	JZ RSNB	;IT'S A SYNC, RESET SNBFLG
0187	BA42	637	MOV RESULT, #SYNC1	;IT'S NOT A SYNC, BAD FIRST SYNC ERROR CODE
0189	04D4	638	JMP RDERR	;EXIT READ ERROR, ADVANCE TO NEXT IRG
018B	FB	639	RSNB: MOV A, RSTAT	;SYNC TEST IS OK, GET READ STATUS
018C	53FD	640	ANL A, #NOT SNBFLG	;RESET SNB FLAG
018E	AB	641	MOV RSTAT, A	;RESTORE STATUS
018F	2430	642	JMP RD3	;GO BACK TO LOOK FOR DATA
		643		
		644	;IRG FOUND - TEST FOR SYNC, CORRECT AND TEST CHECKSUM	
		645		
0191	C5	646	IRGFND: SEL RBO	
0192	FF	647	MOV A, STAT	;GET STS IMAGE
0193	53EF	648	ANL A, #NOT DRACT	;RESET DRIVE ACTIVE
0195	AF	649	MOV STAT, A	;RESTORE IMAGE
0196	90	650	MOV STS, A	;UPDATE STS
0197	BA02	651	ORL P2, #DADFF	;TURN OFF DRIVE ACTIVE LED
0199	D5	652	SEL RB1	
019A	FB	653	MOV A, LBIN	;GET IN POINTER
019B	54D3	654	CALL DUMPIT	;DEC IT TO POINT AT LAST DATA, ALIAS SYNC
019D	AB	655	MOV LBIN, A	;RETURN IT
019E	F0	656	MOV A, @LBIN	;GET LAST DATA
019F	D3AA	657	XRL A, #SYNC	;COMPARE TO SYNC
01A1	96BA	658	JNZ IRGF1	;NOT EQUAL - ERROR
01A3	FB	659	MOV A, LBIN	;GET POINTER AGAIN
01A4	54D3	660	CALL DUMPIT	;DEC IT TO POINT AT 2ND
		661		;TO LAST DATA, ALIAS CHECKSUM
01A6	AB	662	MOV LBIN, A	;RETURN IT
01A7	C5	663	SEL RBO	
01AB	FB	664	MOV A, CHKSUM	;GET ACCUMULATED CHECKSUM
01A9	0356	665	ADD A, #56H	;SUBTRACT OUT SYNC
01AB	AB	666	MOV CHKSUM, A	;RESTORE CHECKSUM
01AC	D5	667	SEL RB1	
01AD	F0	668	MOV A, @LBIN	;GET RECEIVED CHECKSUM
01AE	C5	669	SEL RBO	
01AF	37	670	CPL A	;SUBTRACT IT OUT - MAKE IT MINUS
01B0	17	671	INC A	
01B1	6B	672	ADD A, CHKSUM	;SUBTRACT FROM ACC CHECKSUM
01B2	D5	673	SEL RB1	
01B3	D0	674	XRL A, @LBIN	;COMPARE RESULT TO RECEIVED
01B4	96BE	675	JNZ CKSER	;NOT EQUAL, THEN CHECKSUM ERROR
01B6	BA00	676	MOV RESULT, #GOOD	;EQUAL, THEN GOOD RESULT
01BB	24C0	677	JMP BUFFER	;GO FINISH OFF BUFFER BEFORE OUTPUTTING RESULT
01BA	BA43	678	IRGF1: MOV RESULT, #SYNC2	;2ND SYNC ERROR CODE
01BC	04DF	679	JMP RDERR3	;EXIT
01BE	BA44	680	CKSER: MOV RESULT, #BADCHS	;BAD CHECKSUM ERROR CODE BUT STILL FINISH BUFFER
		681		
		682	;DONE WITH READ - LET BUFFER EMPTY BEFORE OUTPUTTING RESULT	
		683		
01C0	C5	684	BUFFER: SEL RBO	
01C1	FB	685	MOV A, LBOU	;GET OUT POINTER
01C2	D9	686	XRL A, LBRDY	;COMPARE TO READY POINTER
01C3	96C7	687	JNZ BUF1	;NOT EMPTY YET SO GO TEST OBF
01C5	04DF	688	JMP RDERR3	;BUFFER IS EMPTY - GO OUTPUT RESULT
01C7	D6CB	689	BUF1: JNIBF BUF2	;TEST FOR INPUT
01C9	24D5	690	JMP RDIBF	;IF INPUT, GO TEST IT
01CB	56C7	691	BUF2: JDBF BUF1	;TEST OBF
01CD	F0	692	MOV A, @LBOU	;OBF FREE, GET DATA FROM BUFFER
01CE	02	693	OUT DDB, A	;OUTPUT IT
01CF	FB	694	MOV A, LBOU	;GET OUT POINTER
01D0	54CC	695	CALL BUMPIT	;BUMP IT TO POINT AT NEXT DATA
01D2	AB	696	MOV LBOU, A	;RETURN IT
01D3	24C0	697	JMP BUFFER	;GO TEST IT DONE
		698		
		699	;IBF FOUND DURING READ OPERATION - TEST IF ABORT, IGNORE IF NOT	
		700		



# APPLICATIONS

ISIS-II MCS-48/UPI-41 MACRO ASSEMBLER, V3.0  
DIGITAL CASSETTE CONTROLLER REV 1.0 - 26 MARCH 80

LOC	OBJ	LINE	SOURCE	STATEMENT
01D5	22	701	RDIBF:	IN A,DBB ;READ DBBIN
01D6	76DA	702	JF1	ABTST ;TEST FOR COMMAND
01D8	2430	703	JMP	RD3 ;MUST BE DATA, IGNORE IT
01DA	D305	704	ABTST:	XRL A,#ABORT ;COMPARE TO ABORT COMMAND
01DC	9630	705	JNZ	RD3 ;NOT EQUAL, IGNORE IT
01DE	BA01	706	ABTST1:	MOV RESULT,#ABTCMP ;IT IS AN ABORT, ABORT COMPLETE RESULT CODE
01E0	04D4	707	JMP	RDERR ;EXIT LIKE IT WAS AN ERROR, ADVANCE TO IRG
		708		
		709	#EJECT	
0200		710	ORG	200H
		711		
		712		*****
		713		
		714		SKIPER SUBROUTINE - ADVANCES TO NEXT IRG BASED ON DIRECTION AND
		715		SPEED PASSED IN BLKTIM
		716		CARRY=0, NO EOT ENCOUNTERED. CARRY=1, EOT ENCOUNTERED
		717		ENTER AND EXIT IN RBO
		718		
		719		*****
		720		
0200	97	721	SKIPER:	CLR C ;CLEAR EOT INTERNAL FLAG
0201	09	722	IN	A,P1 ;READ DRIVE STATUS
0202	9212	723	JB4	SKIPR3 ;TEST FOR CLEAR LEADER
0204	4600	724	JNT1	SKIPER ;NO CLEAR LEADER, WAIT UNTIL INPUT IS HIGH
0206	560C	725	SKIPR1:	JT1 SKIPR2 ;WHILE INPUT IS HIGH, DEC BLKTIM COUNTER
0208	FE	726	MOV	A,BLKSAV ;INPUT WENT LOW, RESET BLKTIM COUNTER
0209	AD	727	MOV	BLKTIM,A
020A	4400	728	JMP	SKIPER ;GO WAIT UNTIL INPUT IS HIGH AGAIN
020C	09	729	SKIPR2:	IN A,P1 ;READ DRIVE STATUS
020D	9212	730	JB4	SKIPR3 ;TEST CLEAR LEADER - ERROR IF TRUE
020F	ED06	731	DJNZ	BLKTIM,SKIPR1 ;INPUT STILL HIGH, DEC BLKTIM COUNTER
0211	83	732	RET	
0212	A7	733	SKIPR3:	CPL C ;RETURN WHEN AT IRG
0213	83	734	RET	
		735		
		736		*****
		737		
		738		SKIP COMMAND ROUTINE - NEXT DATA BYTE IS NUMBER OF IRG'S TO SKIP
		739		
		740		*****
		741		
0214	D614	742	SKIP:	JNIBF SKIP ;WAIT FOR SKIP COUNT INPUT
0216	7692	743	JF1	CMDINJ ;TEST IF COMMAND INSTEAD - EXIT IF YES
0218	85	744	CLR	FO ;CLEAR DIRECTION FLAG - DEFAULT FORWARD
0219	54BC	745	CALL	DRACTS ;GO SET DRIVE ACTIVE
021B	8904	746	ORL	P1,#SLOW ;START OUT SLOW
021D	22	747	IN	A,DBB ;READ SKIP COUNT INPUT
021E	AC	748	MOV	BLKCNT,A ;SAVE IT IN BLOCK COUNTER
021F	F262	749	JB7	RSKIP ;IF BIT 7 SET, IT'S A REVERSE SKIP
		750		
		751		FORWARD SKIP
		752		
0221	D5	753	SEL	RB1
0222	FB	754	MOV	A,RSTAT ;GET READ STATUS
0223	B278	755	JB5	SKIPB ;STATUS SAYS WE'RE AT EOT - EXIT WITH ERROR
0225	99F4	756	ANL	P1,#FORWD AND SRT AND RD ;IT'S GO - FORWARD
0227	C5	757	SEL	RBO
0228	37	758	CPL	A
0229	D22D	759	JB6	SKIP2 ;COMP A FOR 0 TEST
022B	54DC	760	SKIP1:	CALL PASHOL ;WE'RE NOT AT BOT SO JUST DO SKIP
022D	14C1	761	SKIP2:	CALL DEL150 ;AT BOT SO GET PAST CLEAR LEADER AND HOLE
022F	B644	762	SKIP3:	JFO SKIP6 ;WAIT OUT JUNK AT BEGINNING OF EACH BLOCK
0231	FC	763	MOV	A,BLKCNT ;DON'T WORRY ABOUT FAST OR SLOW WHEN REVERSE
0232	03FB	764	ADD	A,#-BH ;GET BLOCK COUNT
0234	F63E	765	JC	SKIP4 ;SEE IF COUNT IS >8
0236	BD33	766	MOV	BLKTIM,#SLWIRG ;YES, USE FAST IRG TIMING
0238	BE33	767	MOV	BLKSAV,#SLWIRG ;COUNT IS <8, USE SLOW IRG TIMING
023A	8904	768	ORL	P1,#SLOW
023C	4444	769	JMP	SKIP6 ;SELECT SLOW
023E	BD20	770	SKIP4:	MOV BLKTIM,#FASIRG ;GO DO SKIP
0240	BE20	771	MOV	BLKSAV,#FASIRG ;COUNT IS >8, USE FAST IRG TIMING
0242	99FB	772	ANL	P1,#FAST
0244	464B	773	SKIP6:	JNT1 SKIP7 ;SELECT FAST
0246	09	774	IN	A,P1 ;WAIT FOR SPACE TO START IRG FIND
0247	9278	775	JB4	SKIPB ;READ DRIVE STATUS
0249	4444	776	JMP	SKIP6 ;TEST CLEAR LEADER - EXIT IF FOUND
024B	D64F	777	SKIP7:	JNIBF SKIP12 ;CONTINUE TO WAIT FOR SPACE
024D	4487	778	JMP	SKIP11 ;TEST IBF WHILE SKIPPING
024F	5400	779	SKIP12:	CALL SKIPER ;IBF SET, GO TEST IT
0251	F678	780	JC	SKIPB ;DO SKIP TO IRG
0253	EC2D	781	DJNZ	BLKCNT,SKIP2 ;TEST IF EOT OR BOT FOUND
0255	B65E	782	JFO	SKIP13 ;DO IT FOR ALL BLOCK COUNT
0257	D5	783	SKIP14:	SEL RB1 ;DELAY A LITTLE IF REVERSE
0258	BA00	784	MOV	RESULT,#GOOD
025A	B800	785	MOV	RSTAT,#00H ;GOOD RESULT
025C	04DF	786	JMP	RDERR3 ;CLEAR READ STATUS
		787		USE READ EXIT TO COMPLETE
		788		REVERSE SKIP DELAY WHEN BLOCK COUNT EXPIRED
		789		
025E	14BB	790	SKIP13:	CALL DEL50
0260	4457	791	JMP	SKIP14
		792		

# APPLICATIONS

ISIS-II MCS-48/UPI-41 MACRO ASSEMBLER, V3.0  
DIGITAL CASSETTE CONTROLLER REV 1.0 - 26 MARCH 80

LOC	OBJ	LINE	SOURCE STATEMENT
		793	; REVERSE SKIP IS DESIRED - SETUP DRIVE AND DIRECTION FLAG
		794	;
0262	95	795	RSKIP: CPL FO ; SET DIRECTION FLAG
0263	537F	796	ANL A, #7FH ; MASK OFF DIRECTION
0265	AC	797	BLKCNT, A ; MASK OFF BLKCNT
0266	B220	798	MOV BLKTIM, #RWDIRG ; SET REWIND BLOCK TIMER
0268	BE20	799	MOV BLKSAV, #RWDIRG
026A	D5	800	SEL RB1
026B	FB	801	MOV A, RSTAT ; GET READ STATUS
026C	D27B	802	JB4 SKIPB ; AT BOT SO EXIT WITH ERROR
026E	8901	803	ORL P1, #REWIND ; SELECT REVERSE
0270	99F5	804	ANL P1, #SRT AND RD ; START DRIVE
0272	C5	805	SEL RBO
0273	37	806	CPL A ; COMP A FOR 0 TEST
0274	B22D	807	JB5 SKIP2 ; NOT AT EDT SO JUST DO SKIP
0276	442B	808	JMP SKIP1 ; AT EDT SO WAIT PAST CLEAR LEADER AND HOLE
		809	;
		810	; CLEAR LEADER FOUND DURING SKIP OR TAPE ALREADY AT EDT OR BOT
		811	;
0278	D5	812	SKIP8: SEL RB1
0279	B681	813	JF0 SKIP9 ; TEST DIRECTION
027B	BA47	814	MOV RESULT, #SKPEOT ; IT'S FORWARD SO IT'S EDT
027D	BB20	815	MOV RSTAT, #EDTFLG ; SET EDT FLAG
027F	04DF	816	JMP RDERR3 ; GO EXIT
0281	BA48	817	SKIP9: MOV RESULT, #SKPBOT ; IT'S REVERSE SO IT'S BOT
0283	BB40	818	MOV RSTAT, #BOTFLG ; SET BOT FLAG
0285	04DF	819	JMP RDERR3 ; GO EXIT
		820	;
		821	; IBF FOUND SET DURING SKIP - TEST INPUT
		822	;
0287	22	823	SKIP11: IN A, DBB ; READ INPUT
0288	D305	824	XRL A, #ABORT ; TEST IF ABORT
028A	964F	825	JNZ SKIP12 ; IGNORE IT IF NOT
028C	8902	826	ORL P1, #STP ; STOP DRIVE
028E	8A02	827	ORL P2, #DAOFF ; TURN OFF DRIVE ACTIVE LED
0290	24DE	828	JMP ABTST1 ; YES - EXIT WITH RESULT
		829	;
		830	; OUT OF PAGE JUMP FOR CMDIN
		831	;
0292	0433	832	CMDINJ: JMP CMDING
		833	;
		834	*****
		835	;
		836	; REWIND COMMAND - STOP WHEN CLEAR LEADER IS FOUND FOR >50MS.
		837	;
		838	*****
		839	;
0294	D5	840	REWIND: SEL RB1
0295	FB	841	MOV A, RSTAT ; GET READ STATUS
0296	D28B	842	JB4 REWND4 ; TEST IF ALREADY AT BOT - EXIT IF YES
0298	C5	843	SEL RBO
0299	54BC	844	CALL DRACTS ; SET DRIVE ACTIVE
029B	99F3	845	ANL P1, #RD AND FAST ; SELECT RD AND FAST
029D	8901	846	ORL P1, #REWIND ; SELECT REWIND
029F	99FD	847	ANL P1, #SRT ; START DRIVE
02A1	14BB	848	REWND1: CALL DEL50 ; WAIT 50MS
02A3	09	849	IN A, P1 ; READ DRIVE STATUS
02A4	92AB	850	JB4 REWND2 ; TEST CLEAR LEADER
02A6	44A1	851	JMP REWND1 ; NO CLEAR LEADER - WAIT
02AB	14BB	852	REWND2: CALL DEL50 ; WAIT 50MS AGAIN
02AA	09	853	IN A, P1 ; READ DRIVE STATUS AGAIN
02AB	92AF	854	JB4 REWND3 ; AT END IF CLEAR LEADER STILL SET
02AD	44A1	855	JMP REWND1 ; OTHERWISE IT WAS JUST HOLE
02AF	8904	856	REWND3: ORL P1, #STP OR SLOW ; STOP DRIVE, SELECT SLOW
02B1	99FE	857	ANL P1, #FORWD ; SELECT FORWARD
02B3	14BB	858	CALL DEL50 ; WAIT 50MS FOR DRIVE RESET
02B5	D5	859	SEL RB1
02B6	8840	860	MOV RSTAT, #BOTFLG AND (NOT EDTFLG) ; SET UP READ STATUS
02B8	BA00	861	REWND4: MOV RESULT, #GOOD ; GOOD RESULT
02BA	04DF	862	JMP RDERR3 ; GO OUTPUT RESULT
		863	;
		864	*EJECT

# APPLICATIONS

ISIS-II MCS-48/UPI-41 MACRO ASSEMBLER, V3.0  
 DIGITAL CASSETTE CONTROLLER REV 1.0 - 26 MARCH 80

```

LOC  OBJ      LINE      SOURCE STATEMENT
-----
      865 ; *****
      866 ;
      867 ; DRIVE ACTIVE STATUS SUBROUTINE - ENTER/EXIT IN RBO
      868 ; DRIVE ACTIVE BIT IN STATUS IS SET AND DRIVE ACTIVE LED IS TURNED ON
      869 ;
      870 ; *****
      871 ;
02BC  FF      872 DRACTS: MOV      A, STAT      ; GET STS IMAGE
02BD  4310    873         ORL      A, #DRACT    ; SET DRIVE ACTIVE BIT
02BF  AF      874         MOV      STAT, A      ; RESTORE IMAGE
02C0  90      875         MOV      STS, A      ; UPDATE STS
02C1  9AFD    876         ANL      P2, #DAON     ; TURN ON DRIVE ACTIVE LED
02C3  83      877         RET
      878 ;
      879 ; *****
      880 ;
      881 ; DRIVE INACTIVE STATUS SUBROUTINE - ENTER/EXIT IN RBO
      882 ; BOTH DRIVE ACTIVE AND BUSY BITS IN STATUS ARE RESET, DRIVE ACTIVE LED IS OFF
      883 ;
      884 ; *****
      885 ;
02C4  FF      886 NDRACT: MOV      A, STAT      ; GET STS IMAGE
02C5  536F    887         ANL      A, #NOT (BUSY OR DRACT) ; RESET DRACT AND BUSY
02C7  AF      888         MOV      STAT, A      ; RESTORE IMAGE
02C8  90      889         MOV      STS, A      ; UPDATE STS
02C9  8A02    890         ORL      P2, #DAOFF    ; TURN OFF DRIVE ACTIVE LED
02CB  83      891         RET
      892 ;
      893 ; *****
      894 ;
      895 ; BUMPIT - POINTER MANAGEMENT - VALUE IN A IS INCREMENTED AND TESTED
      896 ; FOR OVERFLOW. IF OVERFLOW OCCURS, SET A TO BOTTOM OF BUFFER.
      897 ;
      898 ; *****
      899 ;
02CC  17      900 BUMPIT: INC      A      ; INC A
02CD  D2D0    901         JB6      OVFLOW      ; TEST FOR OVERFLOW
02CF  83      902         RET      ; NO OVERFLOW, RET
02D0  2320    903 OVFLOW: MOV      A, #20H    ; OVERFLOW SO RESET A
02D2  83      904         RET      ; RETURN
      905 ;
      906 $EJECT
      907 ; *****
      908 ;
      909 ; DUMPIT - POINTER MANAGEMENT - VALUE IN A IS DECREMENTED AND TESTED
      910 ; FOR UNDERFLOW. IF UNDERFLOW OCCURS, SET A TO TOP OF BUFFER.
      911 ;
      912 ; *****
      913 ;
02D3  07      914 DUMPIT: DEC      A      ; DEC A
02D4  37      915         CPL      A      ;
02D5  B2D9    916         JB5      UNFLOW     ; TEST IF UNDERFLOW
02D7  37      917         CPL      A      ; NO, COMP BACK
02D8  83      918         RET      ; RETURN
02D9  233F    919 UNFLOW: MOV      A, #3FH    ; UNDERFLOW SO RESET A
02DB  83      920         RET      ; RETURN
      921 ;
      922 ; *****
      923 ;
      924 ; SUBROUTINE TO GET PAST HOLE IN TAPE
      925 ;
      926 ; *****
      927 ;
02DC  09      928 PASHOL: IN       A, P1      ; READ DRIVE STATUS
02DD  92DC    929         JB4      PASHOL     ; WAIT UNTIL OFF CLEAR LEADER
02DF  09      930 PAS1:  IN       A, P1      ; READ DRIVE STATUS AGAIN
02E0  37      931         CPL      A      ; COMP A FOR 0 TEST
02E1  92DF    932         JB4      PAS1      ; WAIT UNTIL HOLE
02E3  09      933 PAS2:  IN       A, P1      ; READ DRIVE STATUS
02E4  92E3    934         JB4      PAS2      ; WAIT UNTIL PAST HOLE
02E6  83      935         RET      ; RETURN
      936 ;
      937 $EJECT
    
```

# APPLICATIONS

ISIS-II MCS-48/UPI-41 MACRO ASSEMBLER, V3.0  
 DIGITAL CASSETTE CONTROLLER REV 1.0 - 26 MARCH 80

```

LOC  OBJ      LINE      SOURCE STATEMENT
0300          938          DRG      300H
939 ;
940 ;*****
941 ;
942 ;TIMER INTERRUPT ROUTINES - FIRST DECIDE IF IT'S READ OR WRITE
943 ;
944 ;*****
945 ;
0300 D5      946 INT:   SEL      RB1
0301 AF      947         MOV      ASAVE,A      ;SAVE ACCUMULATOR
0302 FB      948         MOV      A,RSTAT    ;GET CURRENT STATUS REGISTER
0303 F25E    949         JB7      WRINT      ;IF RD/WR FLAG SET, IT'S A WRITE
950         ;OTHERWISE, IT'S A READ
951 ;
952 ;*****
953 ;
954 ;READ INTERRUPT ROUTINE
955 ;
956 ;*****
957 ;
0305 97      958 RDINT:  CLR      C          ;CLEAR SHIFTER
0306 560C    959         JTI      RDI1       ;TEST INPUT
0308 B612    960         JFO      RD12       ;INPUT=0, TEST LAST
030A 6416    961         JMP      SHIFIN      ;INPUT=0, LAST=0, SHIFT IN 0
030C B616    962 RD11:   JFO      RD13       ;INPUT=1, TEST LAST
030E A7      963         CPL      C          ;INPUT=1, LAST=0, SHIFT IN 1
030F 95      964         CPL      FO         ;SET FO TO CURRENT VALUE OF DATA IN
0310 6416    965         JMP      SHIFIN      ;INPUT=0, LAST=1, SHIFT IN 1
0312 A7      966 RD12:  CPL      C          ;INPUT=0, LAST=1, SHIFT IN 1
0313 95      967         CPL      FO         ;SET FO TO CURRENT VALUE OF DATA IN
0314 6416    968         JMP      SHIFIN      ;INPUT=1, LAST=1, SHIFT IN 0
969 RD13:   970 SHIFIN: MOV      A,DESERL    ;GET CURRENT VALUE OF DATA BYTE
0316 FD      971         RRC      A          ;SHIFT IN NEW BIT
0317 67      972         MOV      DESERL,A    ;RESTORE DESERIALIZER
0318 AD      973         DJNZ    BITCNT,RD14 ;TEST IF BYTE DONE
0319 EC22    974         MOV      RDATA,A     ;IT'S DONE, BUFFER IT IN RDATA
031C BC08    975         MOV      BITCNT,#RDCNT ;RESET BIT COUNTER
031E FB      976         MOV      A,RSTAT     ;GET READ STATUS
031F 4301    977         ORL      A,#RDYFLG   ;SET DATA READY FLAG
0321 AB      978         MOV      RSTAT,A     ;RESTORE STATUS
0322 65      979 RD14:   STOP    TCNT        ;STOP COUNTER
0323 23FA    980         MOV      A,#RDTIM    ;GET TIMER CONSTANT (3/4 CELL TIME)
0325 62      981         MOV      T,A        ;LOAD TIMER
0326 B933    982         MOV      IRCNT,#SLWIRG ;LOAD IRG COUNT (READ USES SLOW SPEED)
0328 09      983 RD17:   IN      A,P1        ;READ DRIVE STATUS
0329 9254    984         RD19      ;TEST IF CLEAR LEADER FOUND - ERROR
032B 564E    985         JTI      RD15       ;TEST INPUT LOOKING FOR EDGE
032D B650    986         JFO      RD16       ;INPUT=0, TEST LAST
032F E928    987 RD18:  DJNZ    IRCNT,RD17  ;INPUT/LAST SAME, DEC IRG COUNT
0331 8902    988         ORL      P1,#STP     ;COUNT EXPIRED, AT IRG, STOP DRIVE
0333 8A02    989         ORL      P2,#DAOFF   ;TURN OFF DRIVE ACTIVE LED
0335 FB      990         MOV      A,RSTAT     ;GET READ STATUS
0336 4308    991         ORL      A,#IRGFLG   ;SET IRG FOUND FLAG
0338 AB      992         MOV      RSTAT,A     ;RESTORE STATUS
993 ;
994 ;INTERRUPT EXIT ROUTINE - UPDATES FO IN STACK TO PRESERVE IT OVER RETR
995 ;
0339 C7      996 INTXT:  MOV      A,PSW        ;GET CURRENT PSW FOR STACK POINTER
033A 5307    997         ANL      A,#07H     ;LOOK AT STACK POINTER ONLY
033C 07      998         DEC      A          ;TRYING TO GET PSW ON TOP OF STACK
033D E7      999         RL      A          ;2 BYTES PER STACK ENTRY
033E 17      1000        INC      A          ;POINT AT PSW ENTRY
033F 0308    1001        ADD      A,#08H     ;ADD OFFSET FOR POINTER
0341 A9      1002        MOV      IRCNT,A    ;LOAD POINTER - USE IRCNT REGISTER
0342 F1      1003        MOV      A,@IRCNT   ;GET PSW
0343 8649    1004        JFO      EXIT1      ;TEST FO TO SEE WHAT TO SET FO TO
0345 530F    1005        ANL      A,#0DFH   ;FO=0 THEREFORE RESET IT
0347 6448    1006        JMP      EXIT2      ;EXIT
0349 4320    1007 EXIT1: ORL      A,#20H     ;FO=1 THEREFORE SET IT
034B A1      1008 EXIT2: MOV      @IRCNT,A ;RESTORE STACK
034C FF      1009        MOV      A,ASAVE    ;RECOVER A
034D 93      1010        RETR      ;RETURN WITH RESTORE
1011 ;
034E B62F    1012 RD15:  JFO      RD18       ;INPUT=1, TEST LAST, SAME
0350 95      1013 RD16:  CPL      FO         ;FINALLY DIFFERENT, SET FO TO CURRENT INPUT
0351 55      1014        STRT    T          ;START TIMER
0352 6439    1015        JMP      INTXT      ;EXIT
1016 ;
0354 FB      1017 RD19:  MOV      A,RSTAT     ;GET READ STATUS
0355 4320    1018        ORL      A,#EOTFLG  ;SET EOT FLAG
0357 AB      1019        MOV      RSTAT,A    ;RESTORE STATUS
0358 8902    1020        ORL      P1,#STP     ;EOT SO STOP DRIVE
035A 8A02    1021        ORL      P2,#DAOFF   ;TURN OFF DRIVE ACTIVE LED
035C 6439    1022        JMP      INTXT      ;EXIT
1023 ;
1024 *EJECT
    
```

# APPLICATIONS

ISIS-II MCS-4B/UPI-41 MACRO ASSEMBLER, V3.0  
DIGITAL CASSETTE CONTROLLER REV 1.0 - 26 MARCH 80

LOC	OBJ	LINE	SOURCE STATEMENT
		1025	*****
		1026	;
		1027	WRITE INTERRUPT ROUTINE
		1028	;
		1029	*****
		1030	;
035E	23FC	1031	WRINT: MOV A,#WRTIM ;GET WRITE TIME CONSTANT (1/2 CELL TIME)
0360	62	1032	MOV T,A ;LOAD TIMER (IT'S STILL RUNNING)
0361	B672	1033	JFO WRINT1 ;TEST IF SECOND INT - DO NEXT BIT IF IT IS
0363	0A	1034	IN A,P2 ;FIRST INT - COMPLEMENT DATA OUT
0364	126A	1035	JBO WR11
0366	8A01	1036	ORL P2,#DDHI
0368	646C	1037	JMP WR12
036A	9AFE	1038	WR11: ANL P2,#DOLW
036C	95	1039	WR12: CPL FO ;SET SECOND INT FLAG
036D	09	1040	IN A,P1 ;TEST FOR CLEAR LEADER
036E	92DC	1041	JB4 CLRLED ;HANDLE IT IF IT'S FOUND
0370	6439	1042	JMP INTEXT ;GO EXIT
		1043	;
		1044	SECOND INTERRUPT FOR THIS BIT - GO GET NEXT BIT
		1045	;
0372	FB	1046	WRINT1: MOV A,WSTAT ;GET WRITE STATUS
0373	52BA	1047	JB2 WRD4 ;TEST WRITE DONE FLAG - DONE IF YES
0375	FD	1048	MOV A,SERIAL ;GET DATA REMAINDER
0376	67	1049	RRC A ;ROTATE NEXT BIT INTO CARRY
0377	AD	1050	MOV SERIAL,A ;RESTORE DATA
0378	E6B3	1051	JNC WR13 ;TEST NEXT BIT
037A	0A	1052	IN A,P2 ;IF 0 - NO CHANGE IN OUTPUT
037B	12B1	1053	JBO WR14 ;IF 1 - COMPLEMENT OUTPUT
037D	8A01	1054	ORL P2,#DDHI
037F	64B3	1055	JMP WR13
0381	9AFE	1056	WR14: ANL P2,#DOLW
0383	85	1057	WR13: CLR FO ;RESET SECOND INT FLAG
0384	EC39	1058	DJNZ BITCNT,INTEXT ;EXIT IF CHR NOT DONE
0386	BC0B	1059	MOV BITCNT,#WRCNT ;CHR IS DONE SO RESET BIT COUNTER
0388	FB	1060	MOV A,WSTAT ;GET WRITE STATUS
0389	12AA	1061	JBO WRD2 ;TEST CHECKSUM FLAG
038B	32B3	1062	JB1 WRD3 ;TEST SYNC FLAG
038D	C5	1063	SEL RBO
		1064	;
		1065	NO SPECIAL CHR BEING DONE - DEC BYTE COUNTER AND TEST IF DONE
		1066	;
038E	E894	1067	DJNZ CNTLSB,WR15 ;DEC LSB - IF NON-ZERO GET NEXT BYTE
0390	F9	1068	MOV A,CNTMSB ;IF ZERO, GET MSB AND TEST IT
0391	C6A1	1069	JZ WRD1 ;IF MSB IS ZERO - DONE, GO WRITE CHECKSUM
0393	C9	1070	DEC CNTMSB ;MSB IS NON-ZERO SO DEC IT
0394	D6CA	1071	WR15: JN15B WRURUN ;NOT DONE WITH ALL BYTES,
		1072	;
0396	22	1073	IN A,DBB ;IS THE NEW DATA IN DBBIN YET? NO - UNDERRUN
0397	AC	1074	MOV TEMP1,A ;YES, READ IT
0398	76CF	1075	JF1 WCOMD ;BE SURE IT WASN'T A COMMAND
039A	68	1076	ADD A,CHKSUM ;IT'S DATA, ADD TO CHECKSUM
039B	AB	1077	MOV CHKSUM,A ;RESTORE CHECKSUM
039C	FC	1078	MOV A,TEMP1 ;GET DATA AGAIN
039D	D5	1079	SEL RB1
039E	AD	1080	MOV SERIAL,A ;PUT DATA IN SERIALIZER
039F	6439	1081	JMP INTEXT ;GO EXIT
		1082	;
		1083	BYTE COUNT DONE - WRITE CHECKSUM
		1084	;
03A1	FB	1085	WRD1: MOV A,CHKSUM ;GET CHECKSUM
03A2	D5	1086	SEL RB1
03A3	AD	1087	MOV SERIAL,A ;PUT IT IN SERIALIZER
03A4	FB	1088	MOV A,WSTAT ;GET WRITE STATUS
03A5	4301	1089	ORL A,#CKSFLG ;SET CHECKSUM FLAG
03A7	AB	1090	MOV WSTAT,A ;RESTORE STATUS
03A8	6439	1091	JMP INTEXT ;GO EXIT
		1092	;
		1093	CHECKSUM BYTE DONE - DO SYNC
		1094	;
03AA	53FE	1095	WRD2: ANL A,#NOT CKSFLG ;RESET CHECKSUM FLAG
03AC	4302	1096	ORL A,#SYNFLG ;SET SYNC FLAG
03AE	AB	1097	MOV WSTAT,A ;RESTORE STATUS
03AF	BDAA	1098	MOV SERIAL,#SYNC ;PUT SYNC IN SERIALIZER
03B1	6439	1099	JMP INTEXT ;GO EXIT
		1100	;
		1101	SYNC DONE - SET WRITE DONE FLAG
		1102	;
03B3	53FD	1103	WRD3: ANL A,#NOT SYNFLG ;RESET SYNC FLAG
03B5	4304	1104	ORL A,#WRDFLG ;SET WRITE DONE FLAG
03B7	AB	1105	MOV WSTAT,A ;RESTORE WRITE STATUS
03B8	6439	1106	JMP INTEXT ;GO EXIT
		1107	;
		1108	WRITE DONE FLAG FOUND SET - COMPLETELY DONE SO STOP AND RESET BUSY
		1109	;
03BA	8A00	1110	WRD4: MOV RESULT,#GOOD ;GOOD RESULT
03BC	8A01	1111	WRD0N: ORL P2,#DDHI ;SET OUTPUT TO 1
03BE	65	1112	STOP TCNT ;STOP TIMER
03BF	8902	1113	ORL P1,#STP ;STOP DRIVE
03C1	8A02	1114	ORL P2,#DAOFF ;TURN OFF DRIVE ACTIVE LED
03C3	D5	1115	SEL RB1
03C4	FB	1116	MOV A,WSTAT ;GET WRITE STATUS
03C5	537D	1117	ANL A,#NOT (WRFLG OR SYNFLG) ;RESET WR/RD FLAG
03C7	AB	1118	MOV WSTAT,A ;RESTORE STATUS

# APPLICATIONS

ISIS-II MCS-48/UPI-41 MACRO ASSEMBLER, V3.0  
 DIGITAL CASSETTE CONTROLLER REV 1.0 - 26 MARCH 80

LDC	OBJ	LINE	SOURCE STATEMENT
03CB	6439	1119	JMP INTTEXT ;GO EXIT
		1120	;
		1121	;UNDERRUN OCCURRED
		1122	;
03CA	D5	1123	WRURUN: SEL RB1
03CB	BAB1	1124	MOV RESULT, #UNDERW ;UNDERRUN ERROR CODE
03CD	648C	1125	JMP WRDON ;EXIT
		1126	;
		1127	;COMMAND FOUND WHEN DATA EXPECTED - CHECK IF ABORT
		1128	;
03CF	D5	1129	WCOMD: SEL RB1
03D0	D305	1130	XRL A, #ABORT ;COMPARE TO ABORT
03D2	C608	1131	JZ WC1 ;YES, THEN ABORT
03D4	BAB2	1132	MOV RESULT, #WCMDCR ;NO, DATA ERROR RESULT CODE
03D6	648C	1133	JMP WRDON ;EXIT
03DB	BA01	1134	WC1: MOV RESULT, #ABTCMP ;ABORT COMPLETE RESULT COE
03DA	648C	1135	JMP WRDON
		1136	;
		1137	;CLEAR LEADER FOUNMD DURING WRITE
		1138	;
03DC	BAB3	1139	CLRLED: MOV RESULT, #EOTERR ;CLEAR LEADER ERROR CODE
03DE	FB	1140	MOV A, WSTAT ;GET WRITE STATUS
03DF	4320	1141	ORL A, #EOTFLG ;SET EOT FLAG
03E1	AB	1142	MOV WSTAT, A ;RESTORE STATUS
03E2	648C	1143	JMP WRDON ;EXIT
		1144	;
		1145	END

USER SYMBOLS

ABORT	0005	ABTCMP	0001	ABTST	01DA	ABTST1	01DE	ARTJMP	0044	ASAVE	0007	B1	0017	B2	001F
BADCHS	0044	BEGIN	0009	BITCNT	0004	BLKNT	0004	BLKSAV	0006	BLKTIM	0005	BDTFLG	0040	BUF1	01C7
BUF2	01CB	BUFFER	01C0	BUMFIT	02CC	BUSY	0080	CHKSAV	0003	CKSER	018E	CKSFLG	0001	CLRLED	03DC
CLRTF	0059	CMDERR	0002	CHDIN	0023	CHDIN1	002C	CHDIN2	003A	CHDIN3	0033	CMDINJ	0292	CMDJMP	003E
CMSAV	0002	CNTLSB	0000	CNTMSB	0001	DAOFF	0002	DAON	00FD	DEL1	008A	DEL150	00C1	DEL2	00BC
DEL50	008B	DESERL	0005	DOH1	0001	DOLQW	00FE	DR1	00CD	DRACT	0010	DRACTS	028C	DRIVER	00C5
DRIVJ	014F	DUMPTI	02D3	EOTERR	0083	EOTFLG	0020	EXIT1	0349	EXIT2	0348	FASIRG	0020	FAST	00F8
FILPRT	0020	FORWD	00FE	GETDAT	0151	GOOD	0000	INT	0300	INTEXT	0339	IRGCNT	0001	IRGF1	018A
IRGFLG	0008	IRGFND	0191	LBIN	0000	LBDUT	0000	LBRDY	0001	LBTST	0177	NDRACT	02C4	NINMSB	0060
NDFULL	0162	NT	00D0	NTAPE	0003	NWR	0004	OVERUN	0041	OVFLOW	02D0	PAS1	02DF	PAS2	02E3
PASHDL	02DC	RCLRTF	0103	RCMDER	0045	RD	00F7	RD1	0127	RD1A	012B	RD2	012D	RD3	0130
RD4	0135	RDATA	0006	RDCMD	0001	RDCNT	0008	RDERR	00D4	RDERR2	00F1	RDERR3	00DF	RDERR4	00E3
RDERR5	00E7	RDERR6	00FB	RD11	030C	RD12	0312	RD13	0316	RD14	0322	RD15	034E	RD16	0350
RD17	0328	RD18	032F	RD19	0354	RD1BF	01D5	RDINT	0305	RDTIM	FFFA	RDYFLG	0001	READ	0100
REDJMP	0046	REDT	0148	REDTER	0046	RESCMD	0000	RESCOM	004E	RESET	0000	RESJMP	0044	RESULT	0002
REWIND	0001	REWJMP	004C	REWND	0294	REWND1	02A1	REWND2	02A8	REWND3	02AF	REWND4	02B8	RSKIP	0262
RSNB	018B	RSTAT	0003	RWCMD	0004	RWDIRG	0020	SERIAL	0005	SHIFIN	0316	SKCMD	0003	SKIP	0214
SKIP1	022B	SKIP11	0287	SKIP12	024F	SKIP13	025E	SKIP14	0257	SKIP2	022D	SKIP3	022F	SKIP4	023E
SKIP6	0244	SKIP7	0248	SKIP8	027B	SKIP9	0281	SKIPR	0200	SKIPR1	0206	SKIPR2	020C	SKIPR3	0212
SKPBOT	0048	SKPEOT	0047	SKPJMP	004A	SLOW	0004	SLWIRG	0033	SNBFLG	0002	SNBST	0182	SRT	00FD
STAP	0007	STP	0002	STRFLG	0004	STSUP	00AD	SYNC	00AA	SYNC1	0042	SYNC2	0043	SYNFLG	0002
TAPIN	0040	TEMPO	0006	TEMP1	0004	TIMINT	0007	UNDERW	0081	UNFLOW	02D9	WC1	03DB	WCMDCR	0082
WCMDC	03CF	WEDTER	0049	WR	000B	WR1	005B	WR2	0061	WR3	008D	WR4	009B	WR5	009F
WRCMD	0002	WRCNT	0008	WRD1	03A1	WRD2	03AA	WRD3	03B3	WRD4	03BA	WRDFLG	0004	WRDON	038C
WRFLG	0080	WRI1	036A	WR12	036C	WR13	0383	WR14	0381	WR15	0394	WRINT	035E	WRINT1	0372
WRITE	0055	WRTIM	FFFC	WRTJMP	0048	WRURUN	03CA	WSTAT	0003						

ASSEMBLY COMPLETE. NO ERRORS

---

# Using the 8295 Dot Matrix Printer Controller

## Contents

INTRODUCTION	2-78
THE 8295	2-78
THE LRC 7040 PRINTER	2-78
8295/PRINTER INTERFACE	2-80
8295 COMMAND SOFTWARE	2-81
PARALLEL INTERFACE	2-82
SERIAL INTERFACE	2-88
8295 SOFTWARE	2-90
CONCLUSION	2-91
APPENDIX	2-92

## INTRODUCTION

Many microprocessor systems require the real-time control of a peripheral device such as a printer, keyboard, or alpha-numeric display, etc. These medium speed but still real-time tasks can be rather mundane, time-consuming, and require a fair amount of system software overhead. Of course, any time spent by the main processor in servicing these I/O devices is unavailable for other, possibly more important, tasks. This processor burden can largely be removed by isolating the real-time portion of the task to a dedicated peripheral-control processor.

Until recently, this approach was usually not cost effective due to the large number of components required by the dedicated processor: CPU, RAM, ROM, I/O, etc. To help make the approach more cost effective, Intel borrowed the I/O processing concepts found in many main-frame and minicomputers; put all the hardware in one package; and introduced a family of Universal Peripheral Interface controllers—the UPI-41A™ family. The basic family consists of the 8041A and the 8741A. These two devices are essentially single-chip microcomputers with a standard microprocessor bus interface. They have on-chip RAM, ROM (8041A) or EPROM (8741A), CPU, timer/counter, and I/O. Using one of the UPI family, the designer simply codes his custom or proprietary peripheral control algorithm into the UPI device itself rather than the main system software. The UPI device then takes over the peripheral control task while the host processor simply issues commands and transfers data. More information on the UPI family is available in the documents referenced opposite the table of contents.

Illustrating the UPI concept as both design examples and actual products, a number of pre-programmed 8041As are available. These devices are the 8278 Keyboard/Display Controller, the 8294 Data Encryption Unit, the 8292 GPIB Controller, and the 8295 Dot Matrix Printer Controller. Data sheets for these devices are found in the Peripheral Design Handbook and their source listings (except for the 8294) are available in Insite, Intel's User's library. This application note deals with the 8295.

## THE 8295

The 8295 Dot Matrix Printer Controller is a device specifically designed to interface microprocessors to the LRC 7040 Series of dot matrix impact printers. It offers complete solenoid and motor drive timing and contains an on-chip 7×7 character generator accommodating 64 ASCII characters. An on-chip FIFO buffers up to 40 ASCII characters before printing. Character density, width, and print intensity are all programmable. Three programmable tabulations and two general purpose outputs are also provided. Four data transfer methods are possible: polling, interrupt-driven, and Direct Memory

Access (DMA) are available when in parallel data transfer mode and asynchronous serial is available in serial mode. The data transfer mode is hardware selectable.

Let's first look at the LRC printer itself and its interface to the 8295.

## THE LRC 7040 PRINTER

The LRC Model 7040 printer is manufactured by LRC, Inc. of Riverton, Wyoming. Capable of printing 40 columns of characters at a speed of 1.25 lines/sec, the 7040 is mechanically simple and is ideal for point-of-sale or data logging terminals.

It is an impact printer whose print head consists of seven solenoids which each drives a stiff wire to impact the paper through an inked ribbon. While the wires are arranged in a circular fashion at the solenoid end, they form a vertical column at the ribbon impact point. Characters are formed by firing the solenoids to form a 5×7 or 7×7 matrix of "dots" (impacts of the wires). Figure 1 shows how the character A is formed using a 7×7 matrix. The columns are labeled C1 thru C7 and the rows R1 thru R7. The print head moves left to right across the paper so at time T1, the head is over column C1. If the correct solenoids are activated at each time Tx for each column Cx, the character is formed.

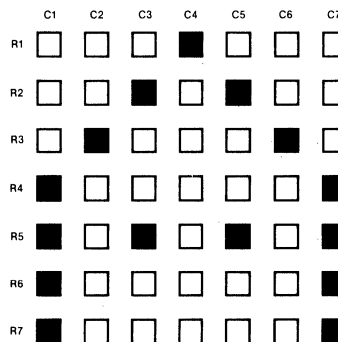


Figure 1. Character A in 7×7 Format

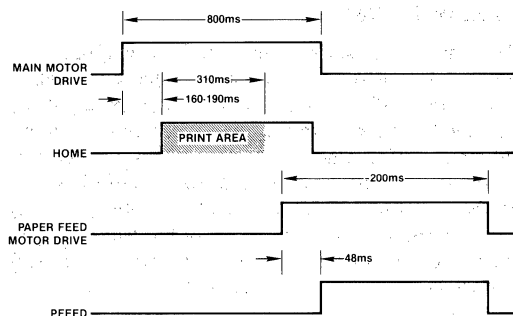
The print head is moved across the paper by the main motor drive. The main motor drive consists of a 24-pole synchronous motor which drives a rotating plastic drum. The drum has a spiral groove molded into it and a pin on the print head rests in the groove so that the print head traverses the paper as the drum rotates. Characters are printed by firing the solenoids during the left-to-right traverse. At the end of the print area, the spiral groove reverses the direction of the print head returning it to its home position.



A HOME microswitch riding on a cam attached to the plastic drum provides the only feedback as to the print head position. When the print head is in its home resting position the HOME switch is inactive. To start a print cycle, the main motor drive is activated which starts the print head motion. As the print head reaches the beginning of the print area, the cam activates the HOME switch as a signal to the printer controller to commence firing the solenoids. The controller then activates the solenoids as appropriate for each character in the line. The print area is defined as the 310ms immediately after HOME goes active. Solenoid timing is the responsibility of the controller; the printer mechanism supplies no character-position information.

After the line is printed and the print head has traversed right to left, the HOME switch is deactivated. This transition signals the controller to turn off the main motor drive since the home position has been reached. A new print cycle may start immediately if data is ready.

Paper feed is accomplished with a second synchronous motor and a PFEED (Paper Feed) microswitch. In the quiescent state, the PFEED switch is inactive. Activating the paper feed motor drive starts the line feed cycle. The switch becomes active at some point during the cycle (typically about 48ms later) and is deactivated when the cycle is complete. The controller uses the active-to-inactive transition to remove the paper feed motor drive. The paper feed operation is independent of the print cycle so the two could occur simultaneously. Figure 2 shows the timing required by the printer for a print cycle followed by a line feed.



**Figure 2. LRC 7040 Motor Drive Timing**

Solenoid timing determines the location of any given "dot" and its intensity. The LRC 7040 printer specification states a  $400\mu\text{s}$  maximum solenoid "ON" time and a 1.3ms typical period. Since the print area is 310ms "long," this timing allows a total of 240 dots ( $310\text{ms}/1.3\text{ms}$  per dot) in one row or 40 characters on a  $5 \times 7$  matrix with a one dot space between characters. While  $5 \times 7$  characters have acceptable readability, their distinctness and format can be improved with a  $7 \times 7$  matrix, however, 40  $7 \times 7$  characters translate to 320 dots per row or a 0.97ms solenoid duty cycle spec if the solenoids are fired for every column. The best way to get around this dilemma and still retain the improved readability of the  $7 \times 7$  format is to simply fire the solenoid every other column. The 8295 uses this technique and the "every-other" column spacing is reflected in Figure 1. The 8295 character set is included in Figure 3.

## CHARACTER SET

Hex Code	Print Char.	Hex Code	Print Char.	Hex Code	Print Char.	Hex Code	Print Char.
20	space	30	0	40	@	50	P
21	!	31	1	41	A	51	Q
22	"	32	2	42	B	52	R
23	#	33	3	43	C	53	S
24	\$	34	4	44	D	54	T
25	%	35	5	45	E	55	U
26	&	36	6	46	F	56	V
27	,	37	7	47	G	57	W
28	(	38	8	48	H	58	X
29	)	39	9	49	I	59	Y
2A	*	3A	:	4A	J	5A	Z
2B	+	3B	;	4B	K	5B	[
2C	.	3C	<	4C	L	5C	\
2D	-	3D	=	4D	M	5D	]
2E	.	3E	>	4E	N	5E	^
2F	/	4F	?	4F	O	5F	_

**Figure 3. 8295 Character Set**

## 8295/Printer Interface

It's the job of the 8295/Printer interface to convert the TTL-compatible outputs of the 8295 to the motor and solenoid drive levels. Since the printer side of the 8295 is independent of the system side, this same 8295/Printer interface is used for all examples discussed in the later sections.

For solenoid drive, the 8295 supplies seven solenoid outputs,  $\overline{S1}$  thru  $\overline{S7}$ , plus a solenoid strobe, STB. STB modulates the S1-S7 outputs externally to supply the actual solenoid "ON" time. This time is software programmable. Figure 4 shows the recommended S1-S7/STB gating.

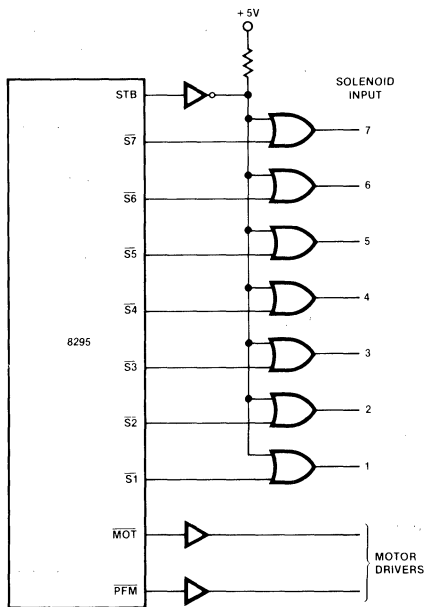


Figure 4. Solenoid and Motor Gating

The solenoids must be driven from a  $40 \pm 10\%$  volt source. The peak current is approximately 3.6A, the average current is approximately 0.5A. A circuit providing the required drive is shown in Figure 5. The output stage, consisting of the 2N6045 Darlington transistor, the 1N4002 catching diode, and the 100-ohm damping resistor, is the one suggested by the manufacturer. The input stage is a discrete implementation of a DTL gate. Note that the base-emitter junction of the 2N6045 protects the 2N2222A transistor from overvoltage on its collector. This circuit has several features which are important to the printer interface:

1. All solenoid power (including the power used to drive the base of the power transistor) is derived from the 40-volt supply.
2. Disconnecting the drivers from the 8295 or the loss of the 5-volt supply to the 8295 results in the solenoids turning off.

The first feature of the drivers minimizes the impact of the printer and its interface on the 5-volt supply. The second feature prevents the activation of the solenoids erroneously during power on/off cycles or during system checkout. This an important point since the solenoids will be damaged if left activated continuously. The fuses in series with the solenoids help protect them from mishap.

The two motors can each be driven as shown in Figure 6. The Monsanto MCS-6200 is an optically-coupled TRIAC which is ideal for driving the small synchronous motors in the printer. Coupled with a buffer this part provides a simple means of controlling the motors without sacrificing the isolation required for safe and reliable operation.

These driver circuits were borrowed from the Intel application note AP-27 "Printer Control With the UPI-41" (The 8295 development was inspired by the success of the AP-27 design.) Other solenoid and motor driver circuits are described in the LRC Interface Guide available from the manufacturer.

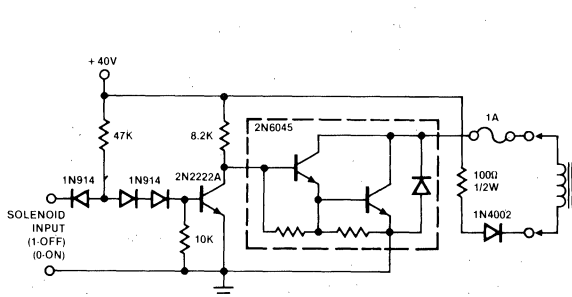


Figure 5. Solenoid Driver

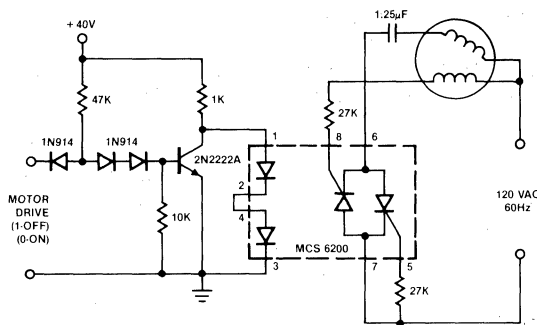


Figure 6. Motor Driver

COMMAND SET

Hex Code	Description	Hex Code	Description
00	Clear GP1. This command brings the GP1 pin to a logic low state. After power on it is automatically set high.	0D	Carriage Return. Signifies end of a line and enables the printer to start printing.
01	Clear GP2. Same as the above but for GP2.	0E*	Set Tab #1, followed by tab position byte.
02	Set GP1. Sets GP1 pin to a logic high state, inverse of command 00.	0F*	Set Tab #2, followed by tab position byte. Should be greater than Tab #1.
03	Set GP2. Same as above but for GP2. Inverse command 01.	10*	Set Tab #3, followed by tab position byte. Should be greater than Tab #1.
04	Software Reset. This is a pacify command. This command is not effective immediately after commands requiring a parameter, as the Reset command will be interpreted as a parameter.	11	Print Head Home on Right. On some printers the print head home position is on the right. This command would enable normal left to right printing with such printers.
05	Print 10 characters/in. density.	12*	Set Strobe Width; must be followed by strobe width selection byte. This command adjusts the duration of the strobe activation.
06	Print 12 characters/in. density.		
07	Print double width characters. This command prints characters at twice the normal width, that is, at either 17 or 20 characters per line.		
08*	Enable DMA mode; must be followed by two bytes specifying the number of data characters to be fetched. Least significant byte accepted first.		
09	Tab character.		
0A	Line feed.		
0B*	Multiple Line Feed; must be followed by a byte specifying the number of line feeds.		
0C	Top of Form. Enables the line feed output until the Top of Form input is activated.		

D7-D3	D2	D1	D0	Solenoid on (μs)
0	0	0	0	200
0	0	0	1	240
0	0	1	0	280
0	0	1	1	320
0	1	0	0	360
0	1	0	1	400
0	1	1	0	440
0	1	1	1	480

\*parameter(s) required

Figure 7. 8295 Command Set

8295 Command Software

The software control of the 8295 is very straightforward. The host processor simply issues ASCII characters to the 8295. The printable characters, 20H thru 5FH, are stored in the on-chip FIFO for printing while the non-printable codes, 00H thru 12H, serve as 8295 commands. (Codes 13H thru 1FH are treated as no-ops.) The 8295 command set is shown in Figure 7. Note that some of the commands require an extra byte or two of information (parameters). These additional parameters must follow the command otherwise data and parameters might be confused. Commands and data may be mixed at any time although while the data is stored in the FIFO, commands take effect immediately. Commands do not "pass-thru" the FIFO.

All printable characters are entered into the FIFO. The FIFO is printed when either a Carriage Return command is received or the FIFO becomes full. In either case, the FIFO is printed, however there is no automatic line feed

unless the printer happens to be so equipped mechanically. Thus, a Line Feed command should be issued after each Carriage Return or after the last character to fill the FIFO. The FIFO is printed as soon as the character that filled it is accepted. If the character immediately following this filling character is a Carriage Return, the 8295 ignores it to prevent a useless print cycle.

Some commands clear the FIFO. The Carriage Return command effectively clears the FIFO since it causes the FIFO contents to be printed. The character density and width commands also clear the FIFO however they do not print its contents; the FIFO size is adjusted by these commands. Obviously, a 10 chr/in density with double width printing would not allow 40 characters per line. The 8295 recognizes this fact and modifies internally the FIFO size limits. The FIFO size is modified according to the table below. For example, if the density is 10 char/in, single width printing, the 8295 accepts only 33 printable

## APPLICATIONS

characters before starting a print cycle. Since these commands take effect as soon as they are accepted, this prevents mixing different character densities or widths on a given line. Any such commands must precede the data for a line.

DENSITY	WIDTH	BUFFER SIZE
12	SINGLE	40
12	DOUBLE	20
10	SINGLE	33
10	DOUBLE	17

The Software Reset command clears the FIFO, resets the density to 12 chr/in and selects single width printing. It does not effect the solenoid strobe width, the tab positions, or the general purpose outputs. This command should be issued only when the 8295 is expecting a command or data. Issuing it when the 8295 is expecting a parameter causes it to be interpreted as the parameter and not the intended software reset.

A hardware reset causes the 8295 to default into the following states:

1. Clears the FIFO
2. GPI and GP2 set high
3. 12 chr/in density
4. single width printing
5. 320 $\mu$ s strobe width
6. tab positions indeterminate.

### Parallel Interfaces

The 8295 has the option of using serial or parallel communication with the main processor. The choice must be

made early in the design cycle since it is a hardware, not a software, selection. Let's look at the parallel options first.

In parallel mode, the 8295 has the traditional microprocessor bus interface: data, control, etc. The parallel mode is selected by not grounding the IRQ/SER pin. To the main processor, the 8295 in parallel mode appears as two registers: the Input Data register and the Output Status register. The main processor writes commands and data into the Input Data register while it reads the 8295 status from the Output Status register.

The Output Status register format is shown in Figure 8. The Input Buffer Full bit (IBF) indicates whether the 8295 has accepted the previous command or data byte. IBF is automatically set when the host processor writes to the 8295 and it is reset when the 8295 accepts the data or command. If IBF = 1, no writes to the Input Data register are allowed. Only when IBF = 0 may a Input Data register write be done. The DMA Enable bit (DE) is set whenever the 8295 is performing DMA data transfers. When the specified number of transfers has been made, the DE bit is cleared. Since DMA cycles are usually transparent to the main processor, the DE bit tells the processor when the DMA block transfer is complete.

The processor does not always have to read the Output Status register, checking IBF, before loading the Input Data register. An interrupt output (IRQ) pin is available to interrupt the processor whenever the 8295 is ready to receive new data or commands. The fact that IRQ is set implies that IBF = 0, so it's not necessary for the processor to read the 8295 status when interrupted; it can just write the next byte.

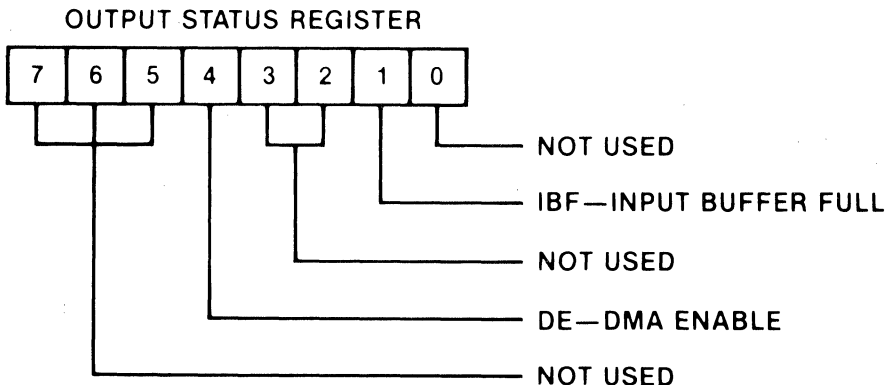


Figure 8. Output Status Register Format

# APPLICATIONS

Figure 9 shows the system schematic for using the 8295 in polled-parallel mode in an 8085A system; ie the IRQ line is not used. The 8085A/8295 interface is standard as for any Intel peripheral.  $\overline{CS}$  is decoded from the high-order address lines.  $\overline{RD}$  and  $\overline{WR}$  are the 8085A read and write control lines.  $\overline{RESET}$  is the system reset.

Example 8085A polling software is shown in Figure 10. This routine simply outputs the print buffer starting at the location pointed to in PRTSRT. The system software builds the buffer, terminates it with a 0FFH character, and loads PRTSRT before calling PRINT.

PRINT is not very efficient with respect to processing time. Since the 8295 does not accept data while in a print or line feed cycle, if the buffer contained more printable characters than the FIFO size, the processor would sit in the PRT2 loop during the 800ms print and 200ms line feed cycles. That is obviously not too efficient. The obvious way around this problem is to restrict the buffer size to less than that of the FIFO however this could complicate the system software since more buffer building is required. A better approach is to use interrupts.

By connecting the 8295's IRQ output to one of the 8085A RST interrupt inputs (dotted line in Figure 9), the pro-

cessor is interrupted only when the 8295 is able to take another character. Figure 11 shows such interrupt-driven software assuming the RST 6.5 interrupt input is used for IRQ.

To further enhance the bus efficiency and processor overhead at the expense of slightly more complex hardware, use the 8295 DMA interface. This DMA interface is compatible with the 8257 DMA Controller. With such an interface all that's necessary is for the processor to load the DMA Controller with the print buffer starting address and write the Enable DMA command and length parameters into the 8295. The 8295 does the rest by requesting data directly from memory thru the DMA Controller. It keeps track of the number of characters to request. As long as there are characters remaining to be transferred, the DE bit in the Output Status register is set. After the last byte is transferred into the 8295, the DE bit is reset and the IRQ is made active. Either event is used to tell the processor that DMA is complete and the 8295 is ready for the next block. It is not necessary to restrict the DMA block size to 40 characters, the Enable DMA command parameters allow for up to 65k byte block sizes. The block size given the 8295 must reflect both data plus commands and parameters.

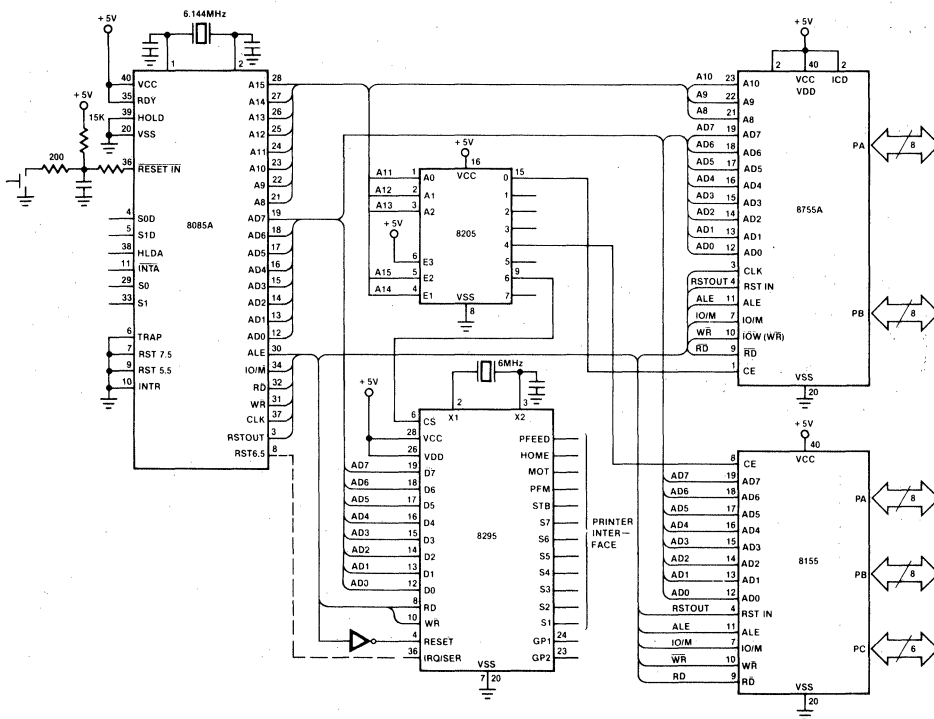


Figure 9. 8295 Parallel Interface

# APPLICATIONS

ASM80 :F1:95F10.SRC TITLE('8295 AP NOTE FIGURE 10')

IS15-II 8080/8085 MACRO ASSEMBLER, X108      MODULE  
8295 AP NOTE FIGURE 10

LOC	OBJ	SEQ	SOURCE STATEMENT
		1	#M0085
		2	;
		3	:SYSTEM EQUATES
2000		4	PR1SRT EQU 2000H      ; POINTER STORAGE
0002		5	IBF EQU 02H          ; IBF FLAG MASK
0031		6	STS95 EQU 31H        ; 8295 STATUS REGISTER PORT
0031		7	DATA95 EQU 31H       ; 8295 DATA REGISTER PORT
		8	;
2030		9	ORG 2030H
		10	;
		11	:PRINT BUFFER OUTPUT SUBROUTINE - THIS ROUTINE PRINTS THE BUFFER
		12	:STARTING AT THE POINTER STORED AT PR1SRT. THE ROUTINE RETURNS WHEN
		13	:A 0FFH IS FETCHED FROM THE BUFFER.
		14	;
2030	E5	15	PRINT: PUSH H            ; SAVE HL
2031	C5	16	PUSH B             ; SAVE BC
2032	2FD020	17	LHLD PR1SRT        ; GET BUFFER POINTER
2035	7E	18	PRT1: MOV A,H       ; GET CHARACTER FROM BUFFER
2036	47	19	MOV B,A            ; SAVE IT IN B
2037	FEFF	20	CPI 0FFH          ; IS IT THE BUFFER END?
2039	CA4A20	21	JZ PEXIT          ; YES, GO EXIT
203C	DB31	22	PRT2: IN STS95     ; NO, READ 8295 STATUS
203E	E602	23	ANI IBF            ; LOOK AT IBF FLAG
2040	C23C20	24	JNZ PRT2          ; WAIT UNTIL IBF=0
2043	78	25	MOV A,B            ; RECOVER CHARACTER
2044	D331	26	OUT DATA95       ; OUTPUT TO 8295
2046	23	27	INX H             ; BUMP BUFFER POINTER
2047	C33520	28	JMP PRT1          ; GET NEXT CHARACTER
		29	;
204A	C1	30	PEXIT: POP B            ; RESTORE BC
204B	E1	31	POP H             ; RESTORE HL
204C	C9	32	RET               ; RETURN
		33	;
		34	END

PUBLIC SYMBOLS

EXTERNAL SYMBOLS

USER SYMBOLS

DATA95 A 0031    IBF    A 0002    PEXIT A 204A    PRINT A 2030    PRT1    H 2035    PRT2    A 203C    PR1SRT A 2000  
STS95    A 0031

ASSEMBLY COMPLETE. NO ERRORS

Figure 10. 8085A/8295 Polling Subroutine

# APPLICATIONS

ASM80 :F1:95F11.SRC TITLE('8295 AP NOTE FIGURE 11')

IS15-11 8080/8085 MACRO ASSEMBLER, X108           MODULE  
8295 AP NOTE FIGURE 11

```

LOC OBJ      SEQ      SOURCE STATEMENT
          1 #MOD85
          2 ;
          3 ;SYSTEM EQUATES
0000      4 PRTSRT EQU 2000H      ; POINTER STORAGE
0002      5 IBF EQU 02H         ; IBF FLAG MASK
0031      6 STS95 EQU 31H       ; 8295 STATUS REGISTER PORT
0031      7 DATA95 EQU 31H     ; 8295 DATA REGISTER PORT
          8 ;
          9 ;
         10 ;RST6.5 INTERRUPT VECTOR LOCATION - JUMP TO PRINTER SUBROUTINE
         11 ;
0034      12      ORG 34H
         13 ;
0034 C33020  14 RST65: JMP PRINT      ; GO TO PRINT ROUTINE
         15 ;
         16 ;
0230      17      ORG 2030H
         18 ;
         19 ;PRINTER OUTPUT SUBROUTINE FOR INTERRUPT-DRIVEN SYSTEM - OUTPUTS
         20 ;CHR POINTED AT BY PRTSRT. IF CHR IS 0FFH, THE BUFFER IS COMPLETE
         21 ;AND THE RST6.5 INTERRUPT IS MASKED. THE MAIN PROGRAM MUST UNMASK
         22 ;RST6.5 AFTER IT BUILDS A NEW BUFFER. PRINT BUFFER STATUS IS REFLECTED
         23 ;TO THE MAIN PROGRAM BY THE RST6.5 MASK BIT IN RIM INSTRUCTION.
         24 ;
0230 E5     25 PRINT: PUSH H          ; SAVE HL
0231 F5     26      PUSH PSW        ; SAVE PSW
0232 2A0020  27      LHLD PRTSRT      ; GET BUFFER POINTER
0235 7E     28      MOV A,M        ; GET NEXT CHR
0236 FEFF  29      CPI 0FFH       ; TEST IF BUFFER COMPLETE
0238 CA4520  30      JZ EXIT         ; YES, GO EXIT WITH RST MASKED
023B D331   31      OUT DATA95     ; NO, OUTPUT CHR TO 8295
023D 23     32      INX H         ; BUMP POINTER
023E 22D020  33      SHLD PRTSRT      ; RESTORE POINTER
0241 F1     34 PRT1: POP PSW       ; RESTORE PSW
0242 E1     35      POP H         ; RESTORE HL
0243 FB     36      EI          ; RE-ENABLE INTERRUPTS
0244 C9     37      RET          ; RETURN
         38 ;
0245 3E0A   39 EXIT: MVI A,0AH        ; MASK RST6.5
0247 30     40      SIM          ; SET INTERRUPT MASK
0248 C34120  41      JMP PRT1        ; GO EXIT WITH MASK IN PLACE
         42 ;
         43      END

```

PUBLIC SYMBOLS

EXTERNAL SYMBOLS

USER SYMBOLS

DATA95 A 0031    EX11    A 2045    IBF    A 0002    PRINT    A 2030    PRT1    A 2041    PRTSRT    A 2000    RST65    A 0034

Figure 11. 8085A/8295 Interrupt-Driven Software

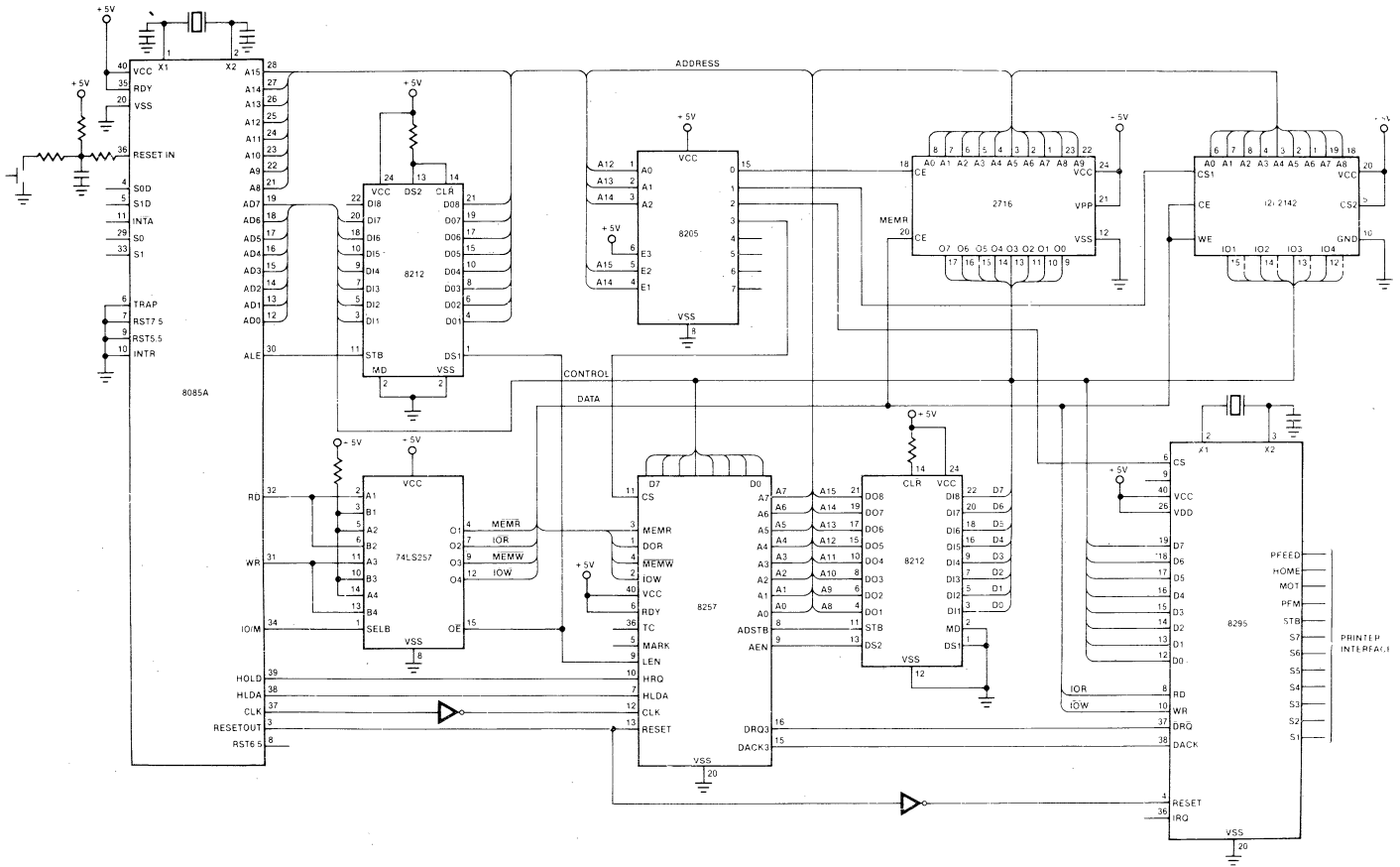


Figure 12. 8295/DMA Interface



# APPLICATIONS

ASM80 :F1:95F13 SRC TITLE('8295 AP NOTE FIGURE 13')

IS15-II 8880/8085 MACRO ASSEMBLER, X108      MODULE    PAGE    1  
8295 AP NOTE FIGURE 13

LOC	OBJ	SEQ	SOURCE STATEMENT
		1	\$MOD85
		2	;
		3	SYSTEM EQUATES:
0038		4	MODE57 EQU 38H ;8257 CONTROL PORT
0036		5	CH3ADR EQU 36H ;8257 CH3 ADR PORT
0037		6	CH3TC EQU 37H ;8257 CH3 TC PORT
0002		7	IBF EQU 02H ;IBF MASK
0020		8	STS95 EQU 20H ;8295 STATUS PORT
0020		9	DAT95 EQU 20H ;8295 DATA PORT
		10	;
2030		11	ORG 2030H
		12	;
		13	DMA-DRIVEN PRINT ROUTINE - THE MAIN PROGRAM CALLS THIS SUBROUTINE
		14	AFTER BUILDING A PRINT BUFFER AND TESTING THE 8295 DE BIT FOR
		15	COMPLETION OF THE LAST DMA BLOCK TRANSFER. THE STARTING ADDRESS
		16	OF THE PRINT BUFFER IS PASSED IN THE DE REGISTER PAIR, THE COUNT IN BC.
		17	;
2030	3E07	18	PRINT: MVI A,07H ;DISABLE DMA CH3
2032	D338	19	OUT MODE57 ;8257 CONTROL PORT
2034	7B	20	MOV A,E ;GET ADR LSB
2035	D336	21	OUT CH3ADR ;8257 CH3 ADR PORT
2037	7A	22	MOV A,D ;GET ADR MSB
2038	D336	23	OUT CH3ADR ;8257 CH3 ADR PORT
203A	3EFF	24	MVI A,0FFH ;MAKE CH3 TC FFFFH
203C	D337	25	OUT CH3TC ;8257 CH3 TC PORT
203E	3EBF	26	MVI A,0BFH ;DMA DIRECTION IS MEMORY READ
2040	D337	27	OUT CH3TC ;8257 CH3 TC PORT
2042	1608	28	MVI D,08H ;ENABLE DMA COMMAND TO 8295
2044	CD5420	29	CALL OUT95 ;OUTPUT TO 8295
2047	51	30	MOV D,C ;GET LSB OF COUNT
2048	CD5420	31	CALL OUT95 ;OUTPUT TO 8295
204B	50	32	MOV D,B ;GET MSB OF COUNT
204C	CD5420	33	CALL OUT95 ;OUTPUT TO 8295
204F	3E0F	34	MVI A,0FH ;ENABLE CH3 DMA
2051	D338	35	OUT MODE57 ;8257 CONTROL PORT
2053	C9	36	RET ;RETURN
		37	;
2054	D820	38	OUT95 IN STS95 ;READ 8295 STATUS
2056	E602	39	ANI IBF ;LOOK AT IBF FLAG
2058	C25420	40	JNZ OUT95 ;WAIT UNTIL IBF=0
205B	7A	41	MOV A,D ;GET DATA
205C	D320	42	OUT DAT95 ;OUTPUT TO 8295 PORT
205E	C9	43	RET ;RETURN
		44	;
		45	END

PUBLIC SYMBOLS  
EXTERNAL SYMBOLS  
USER SYMBOLS

Figure 13. 8295 DMA Subroutine

# APPLICATIONS

Figure 12 illustrates an 8257/8295 interface and Figure 13 shows example software for handling the system. This software assumes that the 8295 is doing the counting of the transfers hence the Terminal Count of the 8257 DMA channel is loaded with the maximum value while the 8295 receives the actual block size. The 8295 simply stops making requests once the requested number of transfers have been made.

## Serial Interface

In addition to the parallel interface options, the 8295 supports a "stand-alone" serial interface. In this mode, the only communication with the main processor is via a serial link. This configuration is perfect for remote printer applications; only three wires are required compared to 12 or 13 for the parallel interfaces.

The serial mode is invoked by simply grounding the IRQ/SER pin. See Figure 14. The internal 8295 software interrogates this pin upon power-on and reconfigures the function of several pins if it's grounded. The DACK/SIN pin becomes the serial data input (SIN) and the DRQ/CTS pin becomes the hardware data holdoff, Clear-to-Send. The lower three Data Bus pins become the Baud Rate Select inputs. Note that it is necessary to ground CS and WR, and pull RD high. This enables the "input" direction of the Data Bus pins so that the 8295 may read the baud rate. All standard baud rates from 110 to 4800 baud are accommodated.

After power-on the 8295 looks at IRQ/SER and if it's grounded, the data bus pins are read to determine the baud rate. Data from the serial input is requested by lowering CTS. CTS stays low until during the eight bit of the serial data character at which point it goes high (inactive). After the character is assembled and interpreted, CTS again goes active to request the next character. The 8295 does not check for parity and characters with invalid start bits or framing errors (stop bit wrong polarity) are ignored. CTS is normally connected to the UART's CTS input. An inactive CTS holds off the UART transmitter from transmitting characters.

In serial mode, the command and data definitions still apply as in parallel mode. Commands and data may be mixed although commands take effect immediately when received.

Figure 15 shows example software to drive an 8251A Programmable Serial Interface when connected to an 8295. This software is similar to Figure 10 except it assumes that the 8251A has the same I/O port addresses as the 8295 had in Figure 9. Note that the TXE (Transmitter Empty) flag is used to load data into the 8251A transmitting both characters in the transmitter (the transmitter is double buffered) if CTS goes inactive. The TXE flag allows only one character at a time in the transmitter so CTS going inactive simply finishes off the current character. The 8295 accepts only one character at a time.

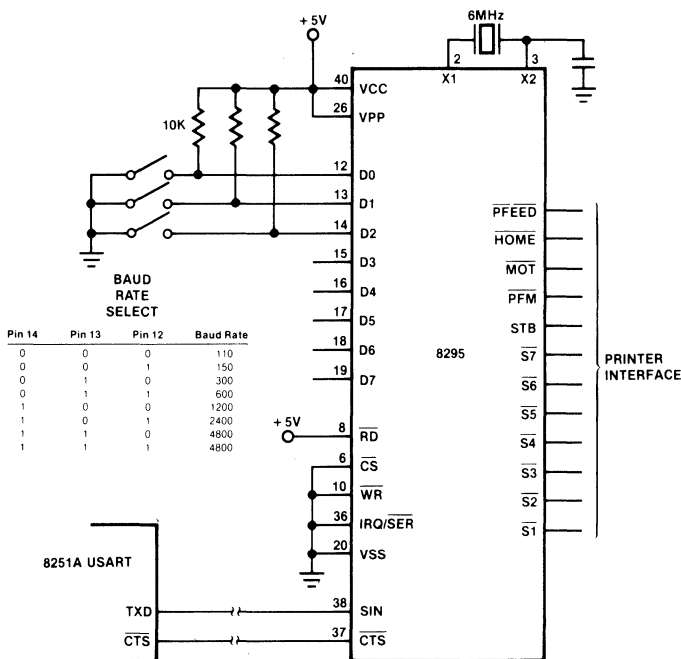


Figure 14. 8295 Serial Interface

# APPLICATIONS

ASM80 :F1:95F15 SRC TITLE('8295 AP NOTE FIGURE 15')

IS15-II 8080/8085 MACRO ASSEMBLER, X108      MODULE    PAGE    1  
8295 AP NOTE FIGURE 15

```

LOC  OBJ      SEQ      SOURCE STATEMENT
      1 #MOD85
      2 ;
      3 ;SYSTEM EQUATES
2000  4 PRTSRT EQU  2000H      ; POINTER STORAGE
0004  5 TXE   EQU  04H       ; TXE FLAG MASK
0031  6 STS51 EQU  31H       ; 8251 STATUS REGISTER PORT
0031  7 DATA51 EQU  31H     ; 8251 DATA REGISTER PORT
      8 ;
2030  9      ORG   2030H
      10 ;
      11 ;PRINT BUFFER OUTPUT SUBROUTINE - THIS ROUTINE PRINTS THE BUFFER
      12 ;STARTING AT THE POINTER STORED AT PRTSRT.  THE ROUTINE RETURNS WHEN
      13 ;A 0FFH IS FETCHED FROM THE BUFFER.
      14 ;
2030  E5    15 PRINT: PUSH  H           ;SAVE HL
2031  05    16      PUSH  B           ;SAVE BC
2032  2AD020 17      LHLD  PRTSRT        ;GET BUFFER POINTER
2035  7E    18 PRT1:  MOV   A,M         ;GET CHARACTER FROM BUFFER
2036  47    19      MOV   B,A         ;SAVE IT IN B
2037  FEFF  20      CPI   0FFH        ;IS IT THE BUFFER END?
2039  CA4A20 21      JZ   PEXIT         ;YES, GO EXIT
203C  0B31  22 PRT2:  IN   STS51         ;NO, READ 8251 STATUS
203E  E604  23      ANI   TXE          ;LOOK AT TXE FLAG
2040  CA3C20 24      JZ   PRT2         ;WAIT UNTIL TXE=1
2043  78    25      MOV   A,B         ;RECOVER CHARACTER
2044  0331  26      OUT  DATA51        ;OUTPUT TO 8251
2046  23    27      INX   H           ;BUMP BUFFER POINTER
2047  C33520 28      JMP  PRT1         ;GET NEXT CHARACTER
      29 ;
204A  01    30 PEXIT: POP   B           ;RESTORE BC
204B  E1    31      POP   H           ;RESTORE HL
204C  09    32      RET            ;RETURN
      33 ;
      34      END

```

PUBLIC SYMBOLS

EXTERNAL SYMBOLS

USER SYMBOLS

DATA51 A 0031    PEXIT A 204A    PRINT A 2030    PRT1 A 2035    PRT2 A 203C    PRTSRT H 2000    STS51 A 0031  
TXE A 0004

ASSEMBLY COMPLETE, NO ERRORS

Figure 15. 8251A Subroutine

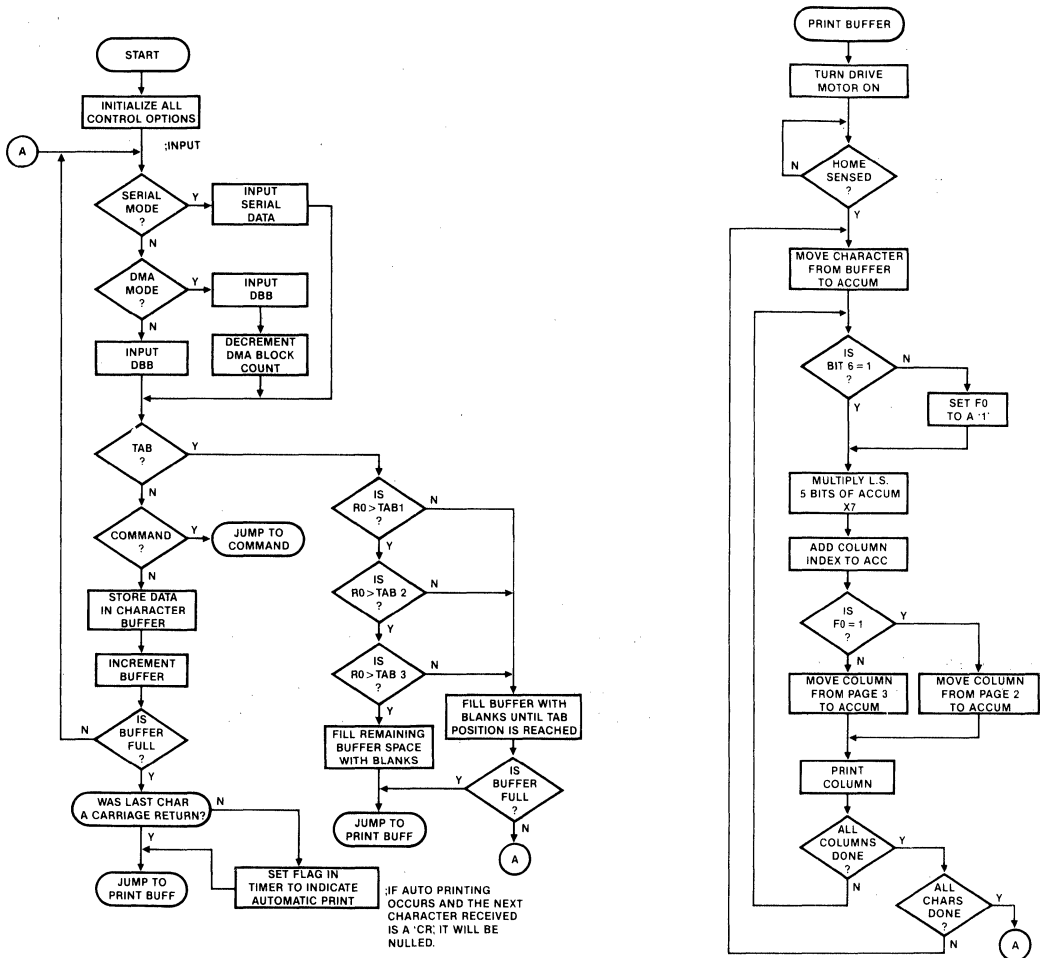


Figure 16. 8295 Flow Chart

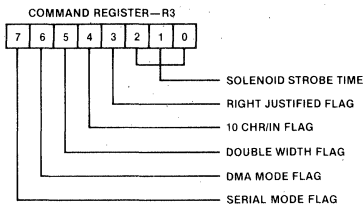
**8295 SOFTWARE**

For those readers using the 8295 as a design example for UPI software, the flow charts for the program are shown in Figure 16 and the 8295 source listing is included as Appendix A. (Machine readable source listings are available through Insite, the Intel User's Library.) As an aid to understanding this software, the following observations can be made:

1. The 8295 uses only Register Bank 0. The function of registers R6 and R7 is determined by the mode. In parallel mode they are concatenated to form the 16 bit DMA count register. In serial mode, R6 is a counter during character reception.
2. Characters and commands are input from the Input Data register via the INPUT subroutine. The routine defines the input mode, fetches the data, and stores it in R2. If the DMA mode is enabled, the block count in R6 and R7 is decremented by the DECR routine each time a data transfer occurs until the count is exhausted.
3. Characters are decoded by routine P6A which also detects any illegal characters by the INPUT routine. R0 is assigned as the character buffer pointer and R4 is designated as the buffer size limit. The commands which affect the buffer size will affect R0 and R4.

# APPLICATIONS

4. Command characters are decoded by the routine CMD. All command routines are referenced via an indirect jump table. The command routines are easy to understand from the listing hence they are not included in Figure 16 but simply referenced.
5. Register R3 is the bit-oriented command register. Each bit of R3 represents an operating mode. This definition is shown below.



6. After the character buffer has reached its limit ( $R0 = R4$ ) or a CR character is received, the contents of the buffer are printed. Subroutine PRINT loads R0 with the address of the character to be printed and R2 serves as an index to keep track of the current column within the character. Subroutine CHAR determines which ASCII table is accessed by setting or clearing flag F0.

7. Subroutine XS2 multiplies the least significant 5 bits of the ASCII character by 7. The result addresses one of the 32 characters on Page 1 or 2 of the Program Memory ASCII table. The column index, R2, is then added to the result to address the current column. Each character is represented by 7 bytes. R2 indexes thru each byte to select the appropriate solenoid information.
8. Subroutine COL8 fetches the solenoid on-time and off-time constants from a table starting at location 0F8H. The time is represented by a hex number which is used as a loop counter in a software timing loop. No character input is allowed while printing is in progress.

## CONCLUSION

The 8295 is an excellent example of what can be done with the UPI-41A family. As a printer controller, it completely relieves the main processor of all the real-time tasks associated with the control of the printer plus valuable system ROM space is not required to store the ASCII-to-dot matrix conversion table or the timing software since it's all done in the 8295 itself. As a UPI design example, the 8295 illustrates the variety of data transfer interfaces available. If the 8295 itself does not fit your printer controller requirements, feel free to modify the 8295 software contained in this application note or that in AP-27 and program your own 8741A.

**Appendix**

APPENDIX A

ASM48 :F1:8295.SRC

ISIS-II MCS-48/UPI-41 MACRO ASSEMBLER, V2.0

PAGE 1

LRC 7040 SERIES PRINTER CONTROLLER SOURCE CODE

```

LOC  OBJ      SEQ      SOURCE STATEMENT
1  #MOD42 TITLE('LRC 7040 SERIES PRINTER CONTROLLER SOURCE CODE')
2
3 ;*****
4 ;** 8295 - LRC 7040 SERIES PRINTER CONTROLLER      **
5 ;** REV. 0 FOR 7X7 CHARACTER MATRIX                **
6 ;*****
7
8
9
10 ;COPYRIGHT (C) 1978
11 ;INTEL CORPORATION
12 ;3065 BOWERS AVE.
13 ;SANTA CLARA, CA. 95051
14
15
16
17 ;*****
18 ;** PAGE0 CONTAINS THE INITIALIZATION SEQUENCE, THE OUTPUTING **
19 ;** OF DATA TO THE SOLENIODS, THE SERIAL INPUT ROUTINE, THE **
20 ;** PAPER FEED ROUTINE, AND THE SOLENIOD FIRETIME ROUTINE    **
21 ;*****
22
23 $EJECT
    
```

ISIS-II MCS-48/UPI-41 MACRO ASSEMBLER, V2.0

PAGE 2

LRC 7040 SERIES PRINTER CONTROLLER SOURCE CODE

```

LOC  OBJ      SEQ      SOURCE STATEMENT
24
25
26
27 ;*****
28 ;**
29 ;**          ; REGISTER ASSIGNMENT TABLE          **
30 ;**
31 ;*****
32 ;**
33 ;**          R0      INPUT BUFFER POINTER          **
34 ;**          R1      TEMPORARY STORAGE              **
35 ;**          R2      TEMPORARY STORAGE              **
36 ;**          R3      COMMAND REGISTER              **
37 ;**          R4      BUFFER SIZE                   **
38 ;**          R5      TEMPORARY STORAGE FOR DELAY ROUTINE **
39 ;**          R6      LOW ORDER DMA COUNTER          **
40 ;**          R7      HIGH ORDER DMA COUNTER         **
41 ;**          TIMER   TEMPORARY STORAGE              **
42 ;**
43 ;*****
44
45
46 $EJECT
    
```

# APPLICATIONS

ISIS-II MCS-48/UPI-41 MACRO ASSEMBLER, V2.0  
LRC 7040 SERIES PRINTER CONTROLLER SOURCE CODE

PAGE 3

LOC	OBJ	SEQ	SOURCE STATEMENT
		47	;*****
		48	;** **
		49	;**           RAM ASSIGNMENT TABLE           **
		50	;** **
		51	;*****
		52	;** **
		53	;**           RAM ADDRESS                   FUNCTION           **
		54	;** **
		55	;**           00-07H                   REGISTER BANK 1   **
		56	;**           08-14H                   PROGRAM STACK     **
		57	;**           15-17H                   TAB POSITION STORAGE **
		58	;**           18-40H                   CHARACTER BUFFER  **
		59	;** **
		60	;*****
		61	
		62	\$EJECT

ISIS-II MCS-48/UPI-41 MACRO ASSEMBLER, V2.0  
LRC 7040 SERIES PRINTER CONTROLLER SOURCE CODE

PAGE 4

LOC	OBJ	SEQ	SOURCE STATEMENT
		63	
		64	;*****
		65	;** **
		66	;**           COMMAND REGISTER DEFINITION           **
		67	;** **
		68	;*****
		69	;** **
		70	;**           BIT 7   SERIAL MODE FLAG           **
		71	;**           BIT 6   DMA MODE FLAG           **
		72	;**           BIT 5   DOUBLE WIDE FLAG       **
		73	;**           BIT 4   32 COLUMNS/LINE       **
		74	;**           BIT 3   RIGHT JUSTIFIED PRINT   **
		75	;**           BITS 2,1,0 INDICATE SOLENOID ON TIME   **
		76	;** **
		77	;*****
		78	\$EJECT



# APPLICATIONS

ISIS-II MCS-48/UPI-41 MACRO ASSEMBLER, V2.0  
 LRC 7040 SERIES PRINTER CONTROLLER SOURCE CODE

PAGE 5

LOC	OBJ	SEQ	SOURCE STATEMENT
0000		79	ORG 000H
		80	
		81	
0000	02	82	INIT OUT DBB:A ;SET OBF
0001	0A	83	IN R,P2 ;CHECK SERIAL STRAP
0002	B208	84	JB5 PARA
0004	B883	85	MOV R3,#83H ;SET SERIAL BIT IN CMD
0006	040E	86	JMP CLR1
0008	9ABF	87	PARA ANL P2,#0BFH
000A	F5	88	EN FLAGS
000B	E5	89	EN DMA
000C	B803	90	MOV R3,#83H
000E	27	91	CLR1 CLR A ;CLEAR DMA BUSY FLAG
000F	90	92	MOV STS:A
0010	8C40	93	CLEAR MOV R4,#40H ;INITIALIZE BUFFER
0012	B818	94	AGAIN MOV R0,#18H ;INITIALIZE POINTER
0014	27	95	CLR A ;RESET STACK TO SAVE TABS
0015	07	96	MOV PSW:A ;STACK = 0, ALL FLAGS = 0
0016	3414	97	DECO CALL INPUT
0018	3428	98	CALL R0A ;DECODE DATA
001A	FC	99	MOV R,R4
001B	D8	100	XPL R,R0
001C	9616	101	JNZ DECO
		102	
001E	FD	103	PRINT MOV R,R3
001F	C8	104	DEC R0 ;LOCATE LAST CHARACTER INPUT IF R.J.
0020	7224	105	JED ON ;CHECK FOR RIGHT JUST.
0022	B818	106	MOV R0,#18H ;PRINT FROM THE ORIGIN
0024	3AFC	107	ON ANL P2,#0BFH ;TURN DRIVE MOTOR ON
0026	4626	108	NHOME JNT1 NHOME ;WAIT FOR HOME SWITCH
0028	2340	109	MOV R,#40H ;STALL
002A	54F8	110	CALL WAIT
002C	B806	111	MOV R2,#06H ;R.J. COL. INDEX
002E	FB	112	MOV R,R3 ;CHECK FOR R.J.
002F	7233	113	JB2 CHAR ;R.J. TRUE
0031	B800	114	MOV R2,#00H ;INDEX FOR NORM. PRINTING
0033	F0	115	CHAR MOV A,@R0 ;FETCH CHARACTER
0034	85	116	CLR F0 ;F0 DETERMINES WHICH CHARACTER TABLE
0035	B238	117	JB5 PAGE
0037	95	118	CPL F0
0038	54E0	119	PAGE CALL X52 ;FETCH COL. FROM TABLE
003A	A9	120	MOV R1,A
003B	FB	121	MOV R,R3 ;CHECK FOR D.W.
003C	B23F	122	JB5 NOT5
003E	95	123	CPL F0 ;F0 INDICATES D.W. MODE
003F	F9	124	NOT5 MOV R,R1
0040	147B	125	CALL FIRE ;PRINT COL.
0042	FB	126	MOV R,R3 ;CHECK R.J.
0043	724D	127	JB3 RJP
0045	2306	128	MOV R,#06H
0047	DA	129	XPL R,R2
0048	1A	130	INC R2
0049	9633	131	JNZ CHAR ;PRINT NEXT COL.
004B	0452	132	JMP L5TCOL
004D	27	133	RJP CLR A ;CHECK R.J. FIRE COLS. IN REVERSE ORDER

# APPLICATIONS

IS15-II MCS-48/UPI-41 MACRO ASSEMBLER, V2.0  
 LRC 7040 SERIES PRINTER CONTROLLER SOURCE CODE

PAGE 6

LOC	OBJ	SEQ	SOURCE STATEMENT
004E	DA	134	XRL A, R2
004F	CA	135	DEC R2
0050	9633	136	JNZ CHAR
0052	B656	137	LSTCOL: JF0 A4
0054	1480	138	CALL COL8
0056	237F	139	MOV A, #7FH ; CLEAR STB & DATA PINS
0058	39	140	OUTL P1, A
0059	2319	141	MOV A, #19H
005B	54F8	142	CALL WAIT
005D	FB	143	MOV A, R3
005E	7264	144	JB3 RJ2
0060	FC	145	MOV A, R4
0061	18	146	INC R0 ; INCR POINTER
0062	0467	147	JMP CK
0064	2317	148	MOV A, #17H
0066	C8	149	DEC R0 ; DECR POINTER
0067	D8	150	CK: XRL A, R0
0068	962C	151	JNZ XFER ; RETURN FOR NEXT CHAR.
006A	566A	152	HOME: JT1 HOME ; SENSE HOME LOW?
006C	2320	153	MOV A, #20H ; STALL
006E	54F8	154	CALL WAIT
0070	8A10	155	ORL P2, #10H ; STOP DRIVE MOTOR
0072	0412	156	JMP AGAIN ; NEXT LINE
		157	
0074	FB	158	DMAIN: MOV A, R3 ; EXIT IF SERIAL MODE
0075	F27A	159	JB7 SERR0R ; SERIAL CMD EKROR
0077	D677	160	INBUF: JNIBF INBUF ; WAIT FOR DMA PARAMS
0079	22	161	IN A, DBB
007A	93	162	SERR0R: RETR
		163	
		164	
		165	
007B	B67F	166	FIRE: JF0 SGLE
007D	09	167	IN A, P1 ; D.W. AND PREVIOUS COL.
007E	59	168	ANL A, R1
007F	39	169	SGLE: OUTL P1, A ; OUTPUT TO SOL.
0080	FB	170	COL8: MOV A, R3 ; A GETS ON TIME
0081	43F8	171	ORL A, #0F8H
0083	A3	172	MOV P, A
0084	530F	173	ANL A, #0FH
0086	8980	174	ORL P1, #80H ; STROBE SOLENOIDS
0088	54F8	175	CALL WAIT
008A	997F	176	ANL P1, #7FH ; DISABLE SOL. STROBE
008C	FB	177	MOV A, R3 ; A GET OFF TIME
008D	43F8	178	ORL A, #0F8H
008F	A3	179	MOV P, A
0090	47	180	SWAP A
0091	530F	181	ANL A, #0FH
0093	2B	182	XCH A, R3
0094	9299	183	JB4 C10
0096	2B	184	XCH A, R3
0097	049C	185	JMP CON
0099	2B	186	C10: XCH A, R3
009A	0306	187	ADD A, #06H ; INCREASE BIAS FOR 10C/I
009C	B6A3	188	CON: JF0 SING ; SKIP IF SINGLE

# APPLICATIONS

IS15-II MCS-48/UPI-41 MACRO ASSEMBLER, V2.0  
 LRC 7040 SERIES PRINTER CONTROLLER SOURCE CODE

PAGE 7

LOC	OBJ	SEQ	SOURCE STATEMENT
009E	0314	189	ADD A, #14H ; ADD 7 TO OFFTIME IF D. W.
00A0	29	190	XCH A, R1 ; SAVE PREVIOUS COL.
00A1	39	191	OUTL P1, A ; SAVE PREVIOUS COL.
00A2	29	192	XCH A, R1
00A3	44F8	193	SING JMP WAIT
		194	
		195	
		196	*****
		197	; SERIAL ROUTINE, ASSEMBLES THE DESIRED DATA FROM THE
		198	; SERIAL INPUT AND PLACE THE DATA IN THE ACCUMULATOR.
		199	*****
		200	
00A5	9ABF	201	CTS: ANL P2, #0BFH ; REQUEST /CTS
00A7	0A	202	ONE: IN A, P2 ; LOOP UNTIL START BIT FOUND
00A8	F2A7	203	JB7 ONE
00AA	B900	204	MOV R1, #0 ; RESET TEMP REG
00AC	BA09	205	MOV R2, #09H ; SET INDEX
00AE	09	206	IN A, P1 ; BIAS
00AF	74E0	207	CALL HBIT ; WAIT 1/2 CYCLE
00B1	0A	208	IN A, P2 ; CHECK FOR START BIT
00B2	F2A7	209	JB7 ONE ; WRONG START BIT
00B4	BE03	210	MOV R6, #03H
00B6	EEB6	211	LZ: DJNZ R6, LZ
00B8	EACE	212	CONT: DJNZ R2, LOAD ; LOAD THE EIGHT BITS
00BA	8A40	213	ORL P2, #40H ; DISABLE /CTS
00BC	BE06	214	MOV R6, #06H ; BIAS
00BE	EEBE	215	W14: DJNZ R6, W14 ; WAIT
00C0	74E0	216	CALL HBIT
00C2	74E0	217	CALL HBIT
00C4	0A	218	IN A, P2
00C5	37	219	CPL A ; CHECK STOP BIT
00C6	F2A7	220	JB7 ONE ; WRONG STOP BIT
00C8	F9	221	MOV A, R1
00C9	F7	222	RLC A
00CA	537F	223	ANL A, #7FH
00CC	AA	224	MOV R2, A
00CD	93	225	RETR
		226	
		227	
00CE	74E0	228	LOAD: CALL HBIT ; DELAY 1 CYCLE
00D0	74E0	229	CALL HBIT
00D2	BE03	230	MOV R6, #03H
00D4	EED4	231	L1: DJNZ R6, L1
00D6	00	232	NOP
00D7	0A	233	IN A, P2 ; INPUT SERIAL BIT
00D8	5380	234	ANL A, #80H ; MASK BIT
00DA	49	235	ORL A, R1 ; ADD PREVIOUS BITS
00DB	67	236	RRC A
00DC	A9	237	MOV R1, A
00DD	04B8	238	JMP CONT ; FINISH JOB
		239	
00DF	9AFE	240	PF: ANL P2, #0FEH ; PF MOTOR ON
00E1	B90A	241	MOV R1, #09AH
00E3	2388	242	P3C: MOV A, #088H
00E5	54F8	243	CALL WAIT

# APPLICATIONS

ISIS-II MCS-48/UPI-41 MACRO ASSEMBLER, V2.0  
LRC 7040 SERIES PRINTER CONTROLLER SOURCE CODE

PAGE 8

LOC	OBJ	SEQ	SOURCE STATEMENT
00E7	E9E3	244	DJNZ R1, P3C
00E9	F8	245	IT0: MOV R, R0 ; DELAY CONTANT =BUFF POINTER (10H TO 40H)
00EA	26EA	246	IT1: JNT0 IT1
00EC	54F8	247	CALL WAIT ; DELAY =1MS TO 2.5MS
00EE	36E9	248	JT0 IT0
00F0	23F3	249	MOV R, #0F3H ; STALL
00F2	54F8	250	CALL WAIT
00F4	8A01	251	ORL P2, #01H ; PF MOTOR OFF
00F6	93	252	P3F: RETR
		253	
00F8		254	ORG 0F8H ; SOL ON TIME CONSTANTS
00F8	D4	255	DB 0D4H ; 200US ON TIME
00F9	C5	256	DB 0C5H ; 240
00FA	B6	257	DB 0B6H ; 200
00FB	A7	258	DB 0A7H ; 320 ; DEFAULT
00FC	98	259	DB 98H ; 360
00FD	89	260	DB 89H ; 400
00FE	7A	261	DB 7AH ; 440
00FF	6B	262	DB 6BH ; 480
		263	
		264	
		265	*****
		266	; PAGE 1 INPUTS, DECODES, AND EXECUTES COMMANDS AND DATA.
		267	*****
		268	
0100		269	ORG 100H
0100	00	270	NOP
0101	B5	271	DB (501 AND 0FFH) ; ADDRESS FOR SET OUTPUT 1
0102	B2	272	DB (502 AND 0FFH) ; S02
0103	BB	273	DB (R01 AND 0FFH) ; R01
0104	B8	274	DB (R02 AND 0FFH) ; R02
0105	BE	275	DB (RESET AND 0FFH) ; RESET
0106	A8	276	DB (B32 AND 0FFH) ; B32
0107	E4	277	DB (B40 AND 0FFH) ; B40
0108	EA	278	DB (DWE AND 0FFH) ; DWE
0109	C9	279	DB (SDMA AND 0FFH) ; SDMA
010A	A0	280	DB (SSOL AND 0FFH) ; SSOL
010B	88	281	DB (SLF AND 0FFH) ; SLF
010C	81	282	DB (MLF AND 0FFH) ; MLF
010D	84	283	DB (TOF AND 0FFH) ; TOF
010E	DE	284	DB (CR AND 0FFH) ; CR
010F	72	285	DB (T1 AND 0FFH) ; T1
0110	72	286	DB (T2 AND 0FFH) ; T2
0111	72	287	DB (T3 AND 0FFH) ; T3
0112	F9	288	DB (RJ AND 0FFH) ; RJ
0113	A0	289	DB (SSOL AND 0FFH) ; SSOL
		290	
		291	
		292	
0114	FB	293	INPUT: MOV R, R3
0115	F226	294	JB7 YME
0117	37	295	CPL A
0118	D21C	296	JB6 NODECR
011A	8A40	297	ORL P2, #40H ; SET DRQ FOR DMA
011C	D61C	298	NODECR: JNIBF NODECR ; SHARED BY PARALLEL & DMA

LOC	OBJ	SEQ	SOURCE STATEMENT
011E	22	299	IN A,DEB
011F	537F	300	ANL A,#7FH
0121	AA	301	MOV R2,A
0122	3462	302	CALL DECR ;DEC DMA COUNT FOR DMA & PARALLEL
0124	FA	303	MOV A,R2 ;DATA STORED IN A & R2
0125	93	304	RETR ;RET & RESTORE FLAGS
0126	04A5	305 YME:	JMP CTE ;SERIAL, USE SERIAL INPUT ROUTINE
		306	
0128	74ED	307 P6A:	CALL SPGR ;CHECK FOR SPECIAL CASE CR
012A	D24E	308	JB6 CHECK5
012C	B250	309	DATA ;CHECK FOR VALID CHAR.
012E	D309	310	XRL A,#09H ;TAB ?
0130	9656	311	JNZ CMD ;COMMAND
0132	B915	312 TAB:	MOV R1,#15H ;R1 GETS TAB(I)
0134	BA03	313	MOV R2,#03H
0136	F1	314 P6BB:	MOV A,@R1 ;CHECK TAB
0137	F24D	315	JB7 TERROR ;LIMIT TAB TO 0AH
0139	D24D	316	JB6 TERROR
013B	37	317	DPL A
013C	17	318	INC A
013D	68	319	ADD A,R0
013E	F1	320	MOV A,@R1 ;R GET TAB LOC.
013F	E645	321	JNC P6AA ;FIND WHICH TAB
0141	19	322	INC R1
0142	EA36	323	DJNZ R2,P6BB
0144	FC	324 SPRL:	MOV A,R4 ;EXCEED ALL TAB, FILL IN BLANKS
0145	AA	325 P6AH:	MOV R2,A
0146	B020	326 RTAB:	MOV @R0,#20H
0148	18	327	INC R0
0149	FA	328	MOV A,R2
014A	D8	329	XRL A,R0 ;FILL IN BLANKS
014B	9646	330	JNZ RTAB
014D	93	331 TERROR:	RETR
		332	
014E	B255	333 CHECK5:	JB5 SEND
0150	FA	334 DATA:	MOV A,R2
0151	A0	335	MOV @R0,A
0152	18	336	INC R0
0153	54ED	337	CALL PEON ;SET SPECIAL FLAG FOR LAST DATA CHARACTER
0155	93	338 SEND:	RETR
		339	
0156	B914	340 CMD:	MOV R1,#14H ;R1 EQ INDEX
0158	FA	341 P7C:	MOV A,R2 ;A GETS CMD
0159	17	342	INC A
015A	D9	343	XRL A,R1
015B	C660	344	JZ FOUND ;MATCH ?
015D	E958	345	DJNZ R1,P7C
015F	93	346	RETR
		347	
0160	F9	348 FOUND:	MOV A,R1
0161	B3	349 JMPF	@A ;JUMP INDIRECT TO CMD ROUTINE
		350	
0162	FE	351 DECR:	MOV A,R6
0163	9670	352	JNZ LARS ;DEC R6,R7 AS REG. PAIR,RET ON 0
0165	4F	353	ORL A,R7

# APPLICATIONS

ISIS-II MCS-48/UPI-41 MACRO ASSEMBLER, V2.0  
 LRD (NO TEST) SCHEDULER SOURCE CODE

PAGE 10

LOC	OBJ	SEQ	SOURCE STATEMENT
0166	966F	354	JNZ NRST
0168	2B	355	XCH A,R3
0169	53BF	356	ANL A,#0BFH
016B	2B	357	XCH A,R3
016C	90	358	MOV STS,A
016D	8A20	359	ORL P2,#20H ;ENABLE INTERRUPT PIN
016F	0F	360	NRST: DEC R7
0170	0E	361	LAPS: DEC R6
0171	93	362	RETR
		363	
		364	
		365	;*****
		366	; COMMAND LOOK UP TABLE
		367	;*****
		368	
		369	
		370	T1: ;A = ADDR OF CMD JUMP IN CMD TABLE
		371	T2: ;A = F, 1 OR 0
0172	17	372	T3: INC A ;A=0, 1, OR 2H
0173	5303	373	ANL A,#03H ;MASK SIGNIFICANT BITS
0175	0315	374	ADD A,#15H ;ACCUM = 15, 16, OR 17H -(RAM LOCATIONS FOR TABS)
0177	62	375	STAB: MOV T,A ;TEMP STORAGE FOR TAB
0178	3414	376	CALL INPUT
017A	0318	377	ADD A,#15H
017C	A9	378	MOV R1,A
017D	42	379	MOV A,T
017E	29	380	XCH A,R1
017F	A1	381	MOV @R1,A
0180	93	382	RETR
		383	
0181	85	384	MLF: CLR F0 ;MULTIPLE LINE FEED
0182	248A	385	JMP LF
		386	
0184	97	387	TOF: CLR C ;TOP OF FORM
0185	A7	388	CPL C
0186	248A	389	JMP LF ;LFUTF
		390	
0188	85	391	SLF: CLR F0 ;SINGLE LINE FEED
0189	95	392	CPL F0
018A	F69C	393	LF: JC P12B ;LFUTOF
018C	B693	394	JF0 P12A ;SINGLE LF
018E	3414	395	CALL INPUT
0190	AA	396	MOV R2,A
0191	C69B	397	JZ P12C
0193	140F	398	P12A: CALL PF
0195	F69C	399	JC P12B
0197	B69B	400	JF0 P12C
0199	EA93	401	DJNZ R2,P12A ;DECR. # OF LINES
019B	93	402	P12C: RETR
019C	0A	403	P12B: IN A,P2
019D	3293	404	JB1 P12A
019F	93	405	RETR
		406	
01A0	3414	407	SSOL: CALL INPUT ;FETCH SOL. ON TIME
01A2	2B	408	XCH A,R3

# APPLICATIONS

ISIS-II MCS-48/UPI-41 MACRO ASSEMBLER, V2.0  
 LRC 7040 SERIES PRINTER CONTROLLER SOURCE CODE

PAGE 11

LOC	OBJ	SEQ	SOURCE STATEMENT
01A3	53F8	409	ANL A, #0F8H ; CLEAR PREV. SOL. TIME
01A5	68	410	ADD A, R3
01A6	2B	411	XCH A, R3
01A7	93	412	RETR
		413	
01A8	FB	414 B32:	MOV A, R3 ; 32 CHARACTER BUFFER
01A9	4310	415	ORL A, #10H
01AB	53DF	416	ANL A, #0DFH
01AD	AB	417	MOV R3, A
01AE	BC39	418	MOV R4, #39H ; 33 CHAR. /LINE
01B0	0412	419	JMP AGAIN
		420	
01B2	8A04	421 S02:	ORL P2, #04H ; SET G02
01B4	93	422	RETR
		423	
01B5	8A08	424 S01:	ORL P2, #08H ; SET G01
01B7	93	425	RETR
		426	
01B8	9AFB	427 R02:	ANL P2, #0FBH ; RESET G02
01BA	93	428	RETR
		429	
01BB	9AF7	430 R01:	ANL P2, #0F7H ; RESET G01
01BD	93	431	RETR
		432	
01BE	89FF	433 RESET:	ORL P1, #0FFH ; RESET PORT 1
01C0	23BF	434	MOV A, #0BFH
01C2	3A	435	OUTL P2, A ; RESET PORT 2
01C3	FB	436	MOV A, R3 ; RESET CMD EXCEPT FOR SERIAL & SOL
01C4	5387	437	ANL A, #87H
01C6	AB	438	MOV R3, A
01C7	040E	439	JMP CLR1 ; CLEAR STS & RESET STACK
01C9	1474	440 SDMA:	CALL DMAIN
01CB	AE	441	MOV R6, A ; LOAD DMA COUNTERS
01CC	1474	442	CALL DMAIN
01CE	9ADF	443	ANL P2, #0DFH ; CLEAR INT PIN
01D0	AF	444	MOV R7, A
01D1	4E	445	ORL A, R6
01D2	C662	446	JZ DECR
01D4	3462	447	CALL DECR
01D6	2B	448	XCH A, R3
01D7	4340	449	ORL A, #40H ; SET DMA FLAG
01D9	2B	450	XCH A, R3
01DA	2310	451	MOV A, #10H ; SET FLAG FOR TELL HOST DMA ON
01DC	90	452	MOV STS, A
01DD	93	453	RETR
		454	
01DE	42	455 CR:	MOV A, T ; CHECK BMAX+1 FLAG
01DF	D300	456	XRL A, #0DH ; IF BUFF. PRINTED AUTO, NO CR.
01E1	9644	457	JNZ SPRL
01E3	93	458	RETR
		459	
		460	
01E4	FB	461 B40:	MOV A, R3 ; 40 CHARACTER BUFFER
01E5	53CF	462	ANL A, #0CFH
01E7	AB	463	MOV R3, A

# APPLICATIONS

IS15-II MCS-48/UPI-41 MACRO ASSEMBLER, V2.0  
 LRC 7040 SERIES PRINTER CONTROLLER SOURCE CODE

PAGE 12

LOC	OBJ	SEQ	SOURCE STATEMENT
01E8	0410	464	JMP CLEAR
		465	
		466	
		467	
01EA	2320	468	DWDE: MOV A, #20H ; DOUBLE WIDE PRINT MODE
01EC	4B	469	ORL A, R3 ; SET DW BIT
01ED	0B	470	MOV R3, A
01EE	B818	471	MOV R0, #18H ; CLEAR BUFFER POINTER
01F0	FC	472	MOV A, R4
01F1	D2F6	473	JB6 X0
01F3	BC2A	474	MOV R4, #2AH ; 32 CHAR. BUFFER
01F5	93	475	RETR
01F6	BC2C	476	X0. MOV R4, #2CH ; 40 CHAR. BUFFER
01F8	93	477	RETR
		478	
01F9	FB	479	RJ: MOV A, R3 ; SET RJ BIT IN CMD
01FA	4308	480	ORL A, #08H
01FC	0B	481	MOV R3, A
01FD	93	482	RETR
		483	
		484	
		485	
		486	; *****
		487	; HBIT SUBR. AND THE DATA CONSTANTS ARE IN PAGE 3
		488	; *****
		489	
03E0		490	ORG 2E0H
		491	
03E0	22	492	HBIT: IN A, DBB ; CHECK DBB FOR BUAD RATE
03E1	43F8	493	ORL A, #0F8H
03E3	A3	494	MOVP A, 0A
03E4	0E	495	MOV R6, A
03E5	BF03	496	LOOP1: MOV R7, #03H ; 25US PER LOOP PAIR
03E7	EFE7	497	LOOP2: DJNZ R7, LOOP2
03E9	EEEE	498	DJNZ R6, LOOP1
03EB	0A	499	IN A, P2
03EC	93	500	RETR
		501	
03ED	D300	502	SPCR: XRL A, #00H ; CHECK CR FLAG, EXIT IF TRUE
03EF	96F5	503	JNZ XCR
03F1	34DE	504	CALL CR
03F3	BAFF	505	MOV R2, #0FFH ; DO NOT EXECUTE CR TWICE
03F5	FA	506	XCR: MOV A, R2
03F6	62	507	MOV T, A
03F7	93	508	RETR
		509	
03F8		510	ORG 3F8H
		511	
03F8	B2	512	DB 0B2H ; 110 BAUD
03F9	84	513	DB 0B4H ; 150
03FA	40	514	DB 40H ; 300
03FB	1F	515	DB 1FH ; 600
03FC	0E	516	DB 0EH ; 1200
03FD	06	517	DB 06H ; 2400
03FE	02	518	DB 02H ; 4800



# APPLICATIONS

ISIS-II MCS-48/UPI-41 MACRO ASSEMBLER, V2 0  
 LRC 7040 SERIES PRINTER CONTROLLER SOURCE CODE

PAGE 13

LOC	OBJ	SEQ	SOURCE STATEMENT
03FF	02	519	DB 02H ;4800
		520	
		521	*****
		522	OTHER THAN CHAR TABLE, WAIT AND XS2 ROUTINES EXIST IN PAGE2
		523	*****
		524	
02E0		525	ORG 2E0H
02E0	531F	526 XS2	AML A,#1FH ;FIND & ADJUST CHARACTER INDEX
02E2	A9	527	MOV R1,A ;MULTIPLY INDEX BY 7
02E3	E7	528	RL A
02E4	E7	529	RL A
02E5	69	530	ADD A,R1
02E6	69	531	ADD A,R1
02E7	69	532	ADD A,R1
02E8	6A	533	ADD A,R2 ;ADD COLUMN INDEX TO CHARACTER INDEX
02E9	B6F5	534	JF0 PAGE3
02EB	E3	535	MOVPI A,0A
02EC	83	536	RET
02ED	FC	537 PEON	MOV A,R4 ;SET SPECIAL OR FLAG IF LAST CHAR IS DATA
02EE	D8	538	XRL A,R0
02EF	96F4	539	JNZ FSPA
02F1	230D	540	MOV A,#0DH
02F3	62	541	MOV T,A
02F4	93	542 FSPA	RETR
		543	
		544	
02F5	A3	545 PAGE3	MOVPI A,0A
02F6	85	546	CLR F0
02F7	83	547	RET
02F8	BD06	548 WAIT	MOV R5,#06H
02FA	EDFA	549 CONX	DJNZ R5,CONX ;48US PER COUNT OF ACC
02FC	07	550	DEC A
02FD	96F8	551	JNZ WAIT
02FF	93	552	RETR
		553	
		554	
		555	
		556	*****
		557	CHARACTER TABLE IN PAGE 2
		558	MSB IS IGNORED, DATA INVERTED
		559	SEE EXAMPLE (A)
		560	*****
		561	
0200		562	ORG 200H
		563	
0200	41	564	DB 41H ;0
0201	3F	565	DB 3FH
0202	62	566	DB 62H
0203	3F	567	DB 3FH
0204	62	568	DB 62H
0205	3F	569	DB 3FH
0206	43	570	DB 43H
		571	
0207	70	572	DB 70H ;A -----*
0208	6F	573	DB 6FH ; ---*---

# APPLICATIONS

ISIS-II MCS-48/UPI-41 MACRO ASSEMBLER, V2.0  
LRC 7040 SERIES PRINTER CONTROLLER SOURCE CODE

PAGE 14

ISIS-II MCS-48/UPI-41 MACRO ASSEMBLER, V2.0  
LRC 7040 SERIES PRINTER CONTROLLER SOURCE CODE

PAGE 15

LOC	OBJ	SEQ	SOURCE STATEMENT
0209	5B	574	DB 5BH ; ---*---
020A	3F	575	DB 3FH ; -*-----
020B	5B	576	DB 5BH ; ---*---
020C	6F	577	DB 6FH ; ---*---
020D	7D	578	DB 7DH ; ----****
		579	
020E	3E	580	DB 3EH ; B
020F	41	581	DB 41H
0210	3E	582	DB 3EH
0211	77	583	DB 77H
0212	3E	584	DB 3EH
0213	77	585	DB 77H
0214	49	586	DB 49H
		587	
0215	41	588	DB 41H ; C
0216	3E	589	DB 3EH
0217	7F	590	DB 7FH
0218	3E	591	DB 3EH
0219	7F	592	DB 7FH
021A	3E	593	DB 3EH
021B	5D	594	DB 5DH
		595	
021C	3E	596	DB 3EH ; D
021D	41	597	DB 41H
021E	3E	598	DB 3EH
021F	7F	599	DB 7FH
0220	3E	600	DB 3EH
0221	7F	601	DB 7FH
0222	41	602	DB 41H
		603	
0223	00	604	DB 00H ; E
0224	7F	605	DB 7FH
0225	36	606	DB 36H
0226	7F	607	DB 7FH
0227	36	608	DB 36H
0228	7F	609	DB 7FH
0229	3E	610	DB 3EH
		611	
022A	00	612	DB 00H ; F
022B	7F	613	DB 7FH
022C	37	614	DB 37H
022D	7F	615	DB 7FH
022E	37	616	DB 37H
022F	7F	617	DB 7FH
0230	3F	618	DB 3FH
		619	
0231	41	620	DB 41H ; G
0232	3E	621	DB 3EH
0233	7F	622	DB 7FH
0234	3E	623	DB 3EH
0235	7B	624	DB 7BH
0236	3E	625	DB 3EH
0237	59	626	DB 59H
		627	
0238	00	628	DB 00H

LOC	OBJ	SEQ	SOURCE STATEMENT
0239	7F	629	DB 7FH ; H
023A	77	630	DB 77H
023B	7F	631	DB 7FH
023C	77	632	DB 77H
023D	7F	633	DB 7FH
023E	00	634	DB 00H
		635	
023F	7F	636	DB 7FH ; I
0240	3E	637	DB 3EH
0241	7F	638	DB 7FH
0242	00	639	DB 00H
0243	7F	640	DB 7FH
0244	3E	641	DB 3EH
0245	7F	642	DB 7FH
		643	
0246	7D	644	DB 7DH ; J
0247	7E	645	DB 7EH
0248	7F	646	DB 7FH
0249	7E	647	DB 7EH
024A	7F	648	DB 7FH
024B	7E	649	DB 7EH
024C	01	650	DB 01H
		651	
024D	00	652	DB 00H ; K
024E	7F	653	DB 7FH
024F	6F	654	DB 6FH
0250	77	655	DB 77H
0251	5B	656	DB 5BH
0252	7D	657	DB 7DH
0253	3E	658	DB 3EH
		659	
0254	00	660	DB 00H
0255	7F	661	DB 7FH
0256	7E	662	DB 7EH ; L
0257	7F	663	DB 7FH
0258	7E	664	DB 7EH
0259	7F	665	DB 7FH
025A	7E	666	DB 7EH
		667	
025B	40	668	DB 40H ; M
025C	3F	669	DB 3FH
025D	5F	670	DB 5FH
025E	67	671	DB 67H
025F	5F	672	DB 5FH
0260	3F	673	DB 3FH
0261	40	674	DB 40H
		675	
0262	20	676	DB 20H
0263	5F	677	DB 5FH ; N
0264	6F	678	DB 6FH
0265	77	679	DB 77H
0266	7B	680	DB 7BH
0267	7D	681	DB 7DH
0268	02	682	DB 02H
		683	

# APPLICATIONS

ISIS-II MCS-48/UPI-41 MACRO ASSEMBLER, V2.0      PAGE 16  
 LRC 7040 SERIES PRINTER CONTROLLER SOURCE CODE

ISIS-II MCS-48/UPI-41 MACRO ASSEMBLER, V2.0      PAGE 17  
 LRC 7040 SERIES PRINTER CONTROLLER SOURCE CODE

LOC	OBJ	SEQ	SOURCE STATEMENT
0269	41	684	DB 41H ;D
026A	3E	685	DB 3EH
026B	7F	686	DB 7FH
026C	3E	687	DB 3EH
026D	7F	688	DB 7FH
026E	3E	689	DB 3EH
026F	41	690	DB 41H
		691	
0270	00	692	DB 00H ;P
0271	7F	693	DB 7FH
0272	37	694	DB 37H
0273	7F	695	DB 7FH
0274	37	696	DB 37H
0275	7F	697	DB 7FH
0276	4F	698	DB 4FH
		699	
0277	41	700	DB 41H ;Q
0278	3E	701	DB 3EH
0279	7F	702	DB 7FH
027A	3F	703	DB 3FH
027B	7A	704	DB 7AH
027C	3D	705	DB 3DH
027D	42	706	DB 42H
		707	
027E	00	708	DB 00H ;R
027F	7F	709	DB 7FH
0280	37	710	DB 37H
0281	7F	711	DB 7FH
0282	33	712	DB 33H
0283	7D	713	DB 7DH
0284	4E	714	DB 4EH
		715	
0285	4D	716	DB 4DH ;S
0286	36	717	DB 36H
0287	7F	718	DB 7FH
0288	36	719	DB 36H
0289	7F	720	DB 7FH
028A	36	721	DB 36H
028B	59	722	DB 59H
		723	
028C	3F	724	DB 3FH
028D	7F	725	DB 7FH
028E	3F	726	DB 3FH
028F	40	727	DB 40H ;T
0290	3F	728	DB 3FH
0291	7F	729	DB 7FH
0292	3F	730	DB 3FH
		731	
0293	01	732	DB 01H ;U
0294	7E	733	DB 7EH
0295	7F	734	DB 7FH
0296	7E	735	DB 7EH
0297	7F	736	DB 7FH
0298	7E	737	DB 7EH
0299	01	738	DB 01H

LOC	OBJ	SEQ	SOURCE STATEMENT
		739	
029A	07	740	DB 07H ;V
029B	7B	741	DB 7BH
029C	7D	742	DB 7DH
029D	7E	743	DB 7EH
029E	7D	744	DB 7DH
029F	7B	745	DB 7BH
02A0	07	746	DB 07H
		747	
02A1	01	748	DB 01H
02A2	7E	749	DB 7EH
02A3	7D	750	DB 7DH
02A4	73	751	DB 73H
02A5	7D	752	DB 7DH
02A6	7E	753	DB 7EH ;W
02A7	01	754	DB 01H
		755	
02A8	3E	756	DB 3EH ;X
02A9	5D	757	DB 5DH
02AA	6B	758	DB 6BH
02AB	77	759	DB 77H
02AC	6B	760	DB 6BH
02AD	5D	761	DB 5DH
02AE	3E	762	DB 3EH
		763	
02AF	3F	764	DB 3FH ;Y
02B0	5F	765	DB 5FH
02B1	6F	766	DB 6FH
02B2	70	767	DB 70H
02B3	6F	768	DB 6FH
02B4	5F	769	DB 5FH
02B5	3F	770	DB 3FH
		771	
02B6	3E	772	DB 3EH
02B7	7D	773	DB 7DH ;Z
02B8	3A	774	DB 3AH
02B9	77	775	DB 77H
02BA	2E	776	DB 2EH
02BB	5F	777	DB 5FH
02BC	3E	778	DB 3EH
		779	
02BD	00	780	DB 00H ;[
02BE	7F	781	DB 7FH
02BF	3E	782	DB 3EH
02C0	7F	783	DB 7FH
02C1	3E	784	DB 3EH
02C2	7F	785	DB 7FH
02C3	7F	786	DB 7FH
		787	
02C4	3F	788	DB 3FH ;\
02C5	5F	789	DB 5FH
02C6	6F	790	DB 6FH
02C7	77	791	DB 77H
02C8	7B	792	DB 7BH
02C9	7D	793	DB 7DH

# APPLICATIONS

ISIS-II MCS-48/UPI-41 MACRO ASSEMBLER, V2.0  
LRC 7040 SERIES PRINTER CONTROLLER SOURCE CODE

PAGE 18

LOC	OBJ	SEQ	SOURCE STATEMENT
020A	7E	794	DB 7EH
		795	
020B	7F	796	DB 7FH ;J
020C	7F	797	DB 7FH
020D	3E	798	DB 3EH
020E	7F	799	DB 7FH
020F	3E	800	DB 3EH
02D0	7F	801	DB 7FH
02D1	00	802	DB 00H
		803	
02D2	77	804	DB 77H ;C
02D3	6F	805	DB 6FH
02D4	5F	806	DB 5FH
02D5	20	807	DB 20H
02D6	5F	808	DB 5FH
02D7	6F	809	DB 6FH
02D8	77	810	DB 77H
		811	
02D9	7E	812	DB 7EH ;L
02DA	7F	813	DB 7FH
02DB	7E	814	DB 7EH
02DC	7F	815	DB 7FH
02DD	7C	816	DB 7EH
02DE	7F	817	DB 7FH
02DF	7E	818	DB 7EH
		819	
		820	
		821	
822			*****
823			CHAR. TABLE ON PAGE 3
824			MSB IS IGNORED, DATA INVERTED
825			SEE EXAMPLE (A) IN PAGE 2 OF ROM
826			*****
0300		828	ORG 300H
		829	
0300	7F	830	DB 7FH ;BLANK
0301	7F	831	DB 7FH
0302	7F	832	DB 7FH
0303	7F	833	DB 7FH
0304	7F	834	DB 7FH
0305	7F	835	DB 7FH
0306	7F	836	DB 7FH
		837	
0307	7F	838	DB 7FH ;I
0308	7F	839	DB 7FH
0309	7F	840	DB 7FH
030A	02	841	DB 02H
030B	7F	842	DB 7FH
030C	7F	843	DB 7FH
030D	7F	844	DB 7FH
		845	
030E	7F	846	DB 7FH ;"
030F	7F	847	DB 7FH
0310	0F	848	DB 0FH

ISIS-II MCS-48/UPI-41 MACRO ASSEMBLER, V2.0 PAGE 19  
LRC 7040 SERIES PRINTER CONTROLLER SOURCE CODE

LOC	OBJ	SEQ	SOURCE STATEMENT
0311	7F	849	DB 7FH
0312	0F	850	DB 0FH
0313	7F	851	DB 7FH
0314	7F	852	DB 7FH
		853	
0315	6B	854	DB 6BH ;#
0316	7F	855	DB 7FH
0317	00	856	DB 00H
0318	7F	857	DB 7FH
0319	00	858	DB 00H
031A	7F	859	DB 7FH
031B	6B	860	DB 6BH
		861	
031C	4D	862	DB 4DH ;\$
031D	36	863	DB 36H
031E	7F	864	DB 7FH
031F	00	865	DB 00H
0320	7F	866	DB 7FH
0321	36	867	DB 36H
0322	59	868	DB 59H
		869	
0323	0E	870	DB 0EH ;%
0324	7D	871	DB 7DH
0325	0B	872	DB 0BH
0326	77	873	DB 77H
0327	68	874	DB 68H
0328	5F	875	DB 5FH
0329	38	876	DB 38H
		877	
032A	49	878	DB 49H ;&
032B	36	879	DB 36H
032C	7F	880	DB 7FH
032D	37	881	DB 37H
032E	5A	882	DB 5AH
032F	7D	883	DB 7DH
0330	72	884	DB 72H
		885	
0331	7F	886	DB 7FH ;'
0332	7F	887	DB 7FH
0333	7F	888	DB 7FH
0334	0F	889	DB 0FH
0335	7F	890	DB 7FH
0336	7F	891	DB 7FH
0337	7F	892	DB 7FH
		893	
0338	7F	894	DB 7FH
0339	63	895	DB 63H ;(
033A	5D	896	DB 5DH
033B	3E	897	DB 3EH
033C	7F	898	DB 7FH
033D	7F	899	DB 7FH
033E	7F	900	DB 7FH
		901	
033F	7F	902	DB 7FH ;)
0340	7F	903	DB 7FH

# APPLICATIONS

ISIS-II MCS-48/UPI-41 MACRO ASSEMBLER, V2.0 PAGE 20  
LRC 7040 SERIES PRINTER CONTROLLER SOURCE CODE

ISIS-II MCS-48/UPI-41 MACRO ASSEMBLER, V2.0 PAGE 21  
LRC 7040 SERIES PRINTER CONTROLLER SOURCE CODE

LOC	OBJ	SEQ	SOURCE STATEMENT
0341	7F	904	DB 7FH
0342	3E	905	DB 3EH
0343	5D	906	DB 5DH
0344	63	907	DB 63H
0345	7F	908	DB 7FH
		909	
0346	77	910	DB 77H ;*
0347	5D	911	DB 5DH
0348	68	912	DB 68H
0349	14	913	DB 14H
034A	68	914	DB 68H
034B	5D	915	DB 5DH
034C	77	916	DB 77H
		917	
034D	77	918	DB 77H ;+
034E	7F	919	DB 7FH
034F	77	920	DB 77H
0350	49	921	DB 49H
0351	77	922	DB 77H
0352	7F	923	DB 7FH
0353	77	924	DB 77H
		925	
0354	7F	926	DB 7FH ;,
0355	7F	927	DB 7FH
0356	7F	928	DB 7FH
0357	7E	929	DB 7EH
0358	79	930	DB 79H
0359	7F	931	DB 7FH
035A	7F	932	DB 7FH
		933	
035B	7B	934	DB 7BH ;-
035C	7F	935	DB 7FH
035D	7B	936	DB 7BH
035E	7F	937	DB 7FH
035F	7B	938	DB 7BH
0360	7F	939	DB 7FH
0361	7B	940	DB 7BH
		941	
0362	7F	942	DB 7FH ;.
0363	7F	943	DB 7FH
0364	7F	944	DB 7FH
0365	7E	945	DB 7EH
0366	7F	946	DB 7FH
0367	7F	947	DB 7FH
0368	7F	948	DB 7FH
		949	
0369	7E	950	DB 7EH ;/
036A	7D	951	DB 7DH
036B	7B	952	DB 7BH
036C	77	953	DB 77H
036D	6F	954	DB 6FH
036E	5F	955	DB 5FH
036F	3F	956	DB 3FH
		957	
0370	41	958	DB 41H ;0

LOC	OBJ	SEQ	SOURCE STATEMENT
0371	7F	959	DB 7FH
0372	3A	960	DB 3AH
0373	77	961	DB 77H
0374	2E	962	DB 2EH
0375	7F	963	DB 7FH
0376	41	964	DB 41H
		965	
0377	7F	966	DB 7FH ;1
0378	5E	967	DB 5EH
0379	7F	968	DB 7FH
037A	00	969	DB 00H
037B	7F	970	DB 7FH
037C	7E	971	DB 7EH
037D	7F	972	DB 7FH
		973	
037E	5C	974	DB 5CH ;2
037F	3B	975	DB 3BH
0380	7E	976	DB 7EH
0381	37	977	DB 37H
0382	7E	978	DB 7EH
0383	37	979	DB 37H
0384	4E	980	DB 4EH
		981	
0385	3D	982	DB 3DH ;3
0386	7E	983	DB 7EH
0387	7F	984	DB 7FH
0388	7E	985	DB 7EH
0389	2F	986	DB 2FH
038A	56	987	DB 56H
038B	39	988	DB 39H
		989	
038C	7B	990	DB 7BH ;4
038D	77	991	DB 77H
038E	6B	992	DB 6BH
038F	5F	993	DB 5FH
0390	2D	994	DB 2DH
0391	7F	995	DB 7FH
0392	7B	996	DB 7BH
		997	
0393	00	998	DB 00H ;5
0394	7E	999	DB 7EH
0395	2F	1000	DB 2FH
0396	7E	1001	DB 7EH
0397	3F	1002	DB 3FH
0398	6E	1003	DB 6EH
0399	31	1004	DB 31H
		1005	
039A	79	1006	DB 79H ;6
039B	76	1007	DB 76H
039C	6F	1008	DB 6FH
039D	56	1009	DB 56H
039E	3F	1010	DB 3FH
039F	76	1011	DB 76H
03A0	79	1012	DB 79H
		1013	

# APPLICATIONS

IS15-II MCS-48/UPI-41 MACRO ASSEMBLER, V2.0      PAGE 22  
LRC 7040 SERIES PRINTER CONTROLLER SOURCE CODE

IS15-II MCS-48/UPI-41 MACRO ASSEMBLER, V2.0      PAGE 23  
LRC 7040 SERIES PRINTER CONTROLLER SOURCE CODE

LOC	OBJ	SEQ	SOURCE STATEMENT
03A1	3F	1014	DB 3FH ;7
03A2	7F	1015	DB 7FH
03A3	3B	1016	DB 3BH
03A4	77	1017	DB 77H
03A5	2F	1018	DB 2FH
03A6	5F	1019	DB 5FH
03A7	3F	1020	DB 3FH
		1021	
03A8	49	1022	DB 49H ;8
03A9	36	1023	DB 36H
03AA	7F	1024	DB 7FH
03AB	36	1025	DB 36H
03AC	7F	1026	DB 7FH
03AD	36	1027	DB 36H
03AE	49	1028	DB 49H
		1029	
03AF	4F	1030	DB 4FH ;9
03B0	37	1031	DB 37H
03B1	7F	1032	DB 7FH
03B2	36	1033	DB 36H
03B3	7D	1034	DB 7DH
03B4	3B	1035	DB 3BH
03B5	47	1036	DB 47H
		1037	
03B6	7F	1038	DB 7FH
03B7	7F	1039	DB 7FH ;:
03B8	7F	1040	DB 7FH
03B9	6B	1041	DB 6BH
03BA	7F	1042	DB 7FH
03BB	7F	1043	DB 7FH
03BC	7F	1044	DB 7FH
		1045	
03BD	7F	1046	DB 7FH ;;
03BE	7F	1047	DB 7FH
03BF	7E	1048	DB 7EH
03C0	69	1049	DB 69H
03C1	7F	1050	DB 7FH
03C2	7F	1051	DB 7FH
03C3	7F	1052	DB 7FH
		1053	
03C4	7F	1054	DB 7FH ;<
03C5	77	1055	DB 77H
03C6	6B	1056	DB 6BH
03C7	5D	1057	DB 5DH
03C8	3E	1058	DB 3EH
03C9	7F	1059	DB 7FH
03CA	7F	1060	DB 7FH
		1061	
03CB	6B	1062	DB 6BH ;=
03CC	7F	1063	DB 7FH
03CD	6B	1064	DB 6BH
03CE	7F	1065	DB 7FH
03CF	6B	1066	DB 6BH
03D0	7F	1067	DB 7FH
03D1	6B	1068	DB 6BH

LOC	OBJ	SEQ	SOURCE STATEMENT
		1069	
03D2	7F	1070	DB 7FH ;>
03D3	7F	1071	DB 7FH
03D4	3E	1072	DB 3EH
03D5	5D	1073	DB 5DH
03D6	6B	1074	DB 6BH
03D7	77	1075	DB 77H
03D8	7F	1076	DB 7FH
		1077	
03D9	7F	1078	DB 7FH ;?
03DA	5F	1079	DB 5FH
03DB	3F	1080	DB 3FH
03DC	7A	1081	DB 7AH
03DD	37	1082	DB 37H
03DE	4F	1083	DB 4FH
03DF	7F	1084	DB 7FH
		1085	
		1086	END

# APPLICATIONS

USER SYMBOLS

AA	0056	AGAIN	0012	B32	01A8	B40	01E4	C10	0099	CHAR	0033	CHECK5	014E	CK	0067
CLEAR	0010	CLR1	000E	CMD	0156	COL8	0080	CON	009C	CONT	0088	CONX	02FA	CR	01DE
CTS	00A5	DATA	0150	DECO	0016	DECR	0162	DMAIN	0074	DWDE	01EA	FIRE	007B	FOUND	0160
FSPA	02F4	HBIT	03E0	HOME	006A	INBUF	0077	INIT	0000	INPUT	0114	I10	00E9	IT1	00EA
L1	0004	LARS	0170	LF	018A	LOAD	00CE	LOOP1	03E5	LOOP2	03E7	LSTCOL	0052	LZ	0086
MLF	0181	NHOME	0026	NODECR	011C	NOTS	003F	NRST	016F	ON	0024	ONE	00A7	P12A	0193
P12B	019C	P12C	0198	P3C	00E3	P3F	00F6	P6A	0128	P6AA	0145	P6BB	0136	P7C	0158
PAGE	0038	PAGE3	02F5	PARA	0008	PEON	02ED	PF	00DF	PRINT	001E	RESET	018E	RJ	01F9
RJ2	0064	RJP	004D	R01	0188	R02	0188	RTAB	0146	SDMA	01C9	SEND	0155	SERROR	007A
SGLE	007F	SING	00A3	SLF	0188	S01	01B5	S02	01B2	SPCR	03ED	SPRL	0144	SSQL	01A0
STAB	0177	T1	0172	T2	0172	T3	0172	TAB	0132	TERROR	014D	TOF	0184	W14	00BE
WAIT	02F8	X0	01F6	XCR	03F5	XFER	002C	X52	02E0	YME	0126				

ASSEMBLY COMPLETE. NO ERRORS

# ***Memory Controllers***

---



# 5-Volt Only Dynamic RAM Interface for 8086 Systems

## Contents

<b>WHY DYNAMIC MEMORIES?</b>	2-111
<b>TYPES OF DYNAMIC MEMORY SYSTEMS</b>	2-111
<b>8086 SYSTEM CONFIGURATION REVIEW</b>	2-111
Quick Review of 8086 Family Bus Timing	2-111
8202 Read Cycle Timing	2-111
Read Access Time (Min Mode)	2-112
Read Access Time (Max Mode)	2-113
Read Access Time (Alternate Configuration)	2-113
8202 Write Cycle Timing	2-114
Write Cycle Timing (Min Mode)	2-116
Write Cycle Timing (Max Mode)	2-116
Write Cycle Timing (Alternate Configuration)	2-116
Handling 8-Bit Write Cycles in 16-Bit Systems	2-117
Multibus Byte Swap	2-117
PCS Generation	2-117
Ready Handshake Signals (SACK and XACK)	2-117
Refresh Considerations	2-118
References	2-119
<b>APPENDIX 1</b>	2-120
<b>APPENDIX 2</b>	2-122
<b>APPENDIX 3</b>	2-128

**NOTE:**

Refer to the updated application note AP-97A "5-Volt Only Dynamic RAM Interface for 8086 Systems", January 1982, for the latest product information.

## WHY DYNAMIC MEMORIES?

Dynamic RAM offers a four-to-one size advantage over their static RAM counterparts in medium to large size memory systems. In a typical 8086 system with 128K bytes of memory, you would need over 256 IC's and 300 square inches of PC board for a static memory array. The same memory array could be implemented with 68 IC's and 80 square inches of PC board if dynamic RAMs are used.

Besides this obvious size advantage, dynamic RAM designs offer a substantial power dissipation advantage. For example, the 2142 static RAM requires 0.1 mW/bit operating power, while the 2118 dynamic RAM requires only 0.01 mW/bit.

In the following sections we will show how to construct a complete dynamic memory interface for your 8086, 8088, and 8089 systems, using the 8202 dynamic RAM Controller and the 2118 5-volt only dynamic RAMs.

## TYPES OF DYNAMIC MEMORY SYSTEMS

Dynamic memory systems can be divided into two categories: 1) those that use distributed (or asynchronous) refresh and 2) those that use hidden (or synchronous) refresh. Each type has advantages over the other type; your choice will depend on your system requirements.

In a distributed refresh system the memory controller *periodically* requests a refresh cycle, typically every 10–16 microseconds. Since the refresh request is asynchronous to the CPU's request for the memory, the memory controller must have logic to arbitrate the requests. Once a cycle starts, the arbiter must let that cycle complete before starting a pending cycle. The memory controller should also have circuitry which can force the CPU to add WAIT state if a memory cycle is requested while a refresh cycle is in progress.

Hidden refresh designs use circuitry to monitor the CPU status lines, and request a refresh cycle when the CPU is not performing a bus cycle with the dynamic RAM. For example, a hidden refresh cycle can overlap an instruction fetch from ROM. If the hidden refresh cycles are performed frequently enough, the dynamic memory is always ready when the CPU requests a memory cycle, and no WAIT states are required due to arbitration.

Some memory systems use a combination of asynchronous and hidden refresh. For example, many real time systems allow a processor to enter a HALT state (which stops program execution) while waiting for an interrupt. During this time, the hidden refresh circuitry is inactive, and the asynchronous refresh logic inserts the necessary refresh requests. The Intel 8202 Dynamic Memory Controller provides a complete memory interface for your dynamic RAM. It can be used in synchronous and asynchronous refresh systems, and provides automatic switching between these two modes.

## 8086 SYSTEM CONFIGURATION REVIEW

### Quick Review of 8086 Family Bus Timing

There are three basic 8086 family system configurations:

- 1) Minimum Mode
- 2) Maximum Mode
- 3) Alternate Configuration

The 8086 has a  $\overline{MN}/\overline{MX}$  input which can be strapped high to select Min Mode, and grounded to select Max Mode. The  $\overline{MN}/\overline{MX}$  input changes the function of several other 8086 pins based on how it is strapped.

In the Min Mode, the CPU generates the  $\overline{RD}$  and  $\overline{WR}$  outputs directly. The Max Mode uses an 8288 to generate the 8202  $\overline{RD}$  and  $\overline{WR}$  signals from the CPU status lines. Refer to the "8086 Family Users Manual" for more details.

The Alternate Configuration uses several TTL gates and flip-flops and the CPU status outputs to generate the 8202  $\overline{RD}$  and  $\overline{WR}$  signals. The Alternate Configuration can be used when the CPU is strapped in either the Min Mode or the Max Mode. Each of the three basic system configurations can be used with data buffers for additional drive capability.

Regardless of the system configuration, each 8086 family bus cycle consists of four clock cycles, if no WAIT states are required. WAIT states can be used to extend the basic bus cycle, and thus allow the use of slower memories.

We will see in the following sections that the Min Mode will offer the lowest chip count, the Max Mode will offer better performance, while the Alternate Configuration will offer higher performance at the expense of several TTL packages.

### 8202 Read Cycle Timing

The 8202 uses its clock to sample the (asynchronous) READ requests generated by the CPU. When the  $\overline{RD}$  input is sensed active, the 8202 will generate a  $\overline{RAS}$  strobe and  $\overline{CAS}$  strobe as shown in Figure 1. The RAM uses RAS and CAS to latch the CPU address and read the desired location onto the RAM data output pin. The 8202 also generates an  $\overline{XACK}$  strobe which is used to latch the RAM data for the CPU, as shown in Figure 2. (See "READY HANDSHAKE SIGNALS" for further uses of  $\overline{XACK}$ .)

We can determine the amount of time it takes to generate valid READ data by calculating the delay from  $\overline{RD}\downarrow$  to  $\overline{CAS}\downarrow$ , and add this with the RAM's CAS access time ( $t_{CAC}$ ) and the latch's propagation delay. In other words,

$$t_{RLDV} = t_{CC, MAX. (8202)} + t_{CAC} (2118) + t_{PHL} (74LS373)$$

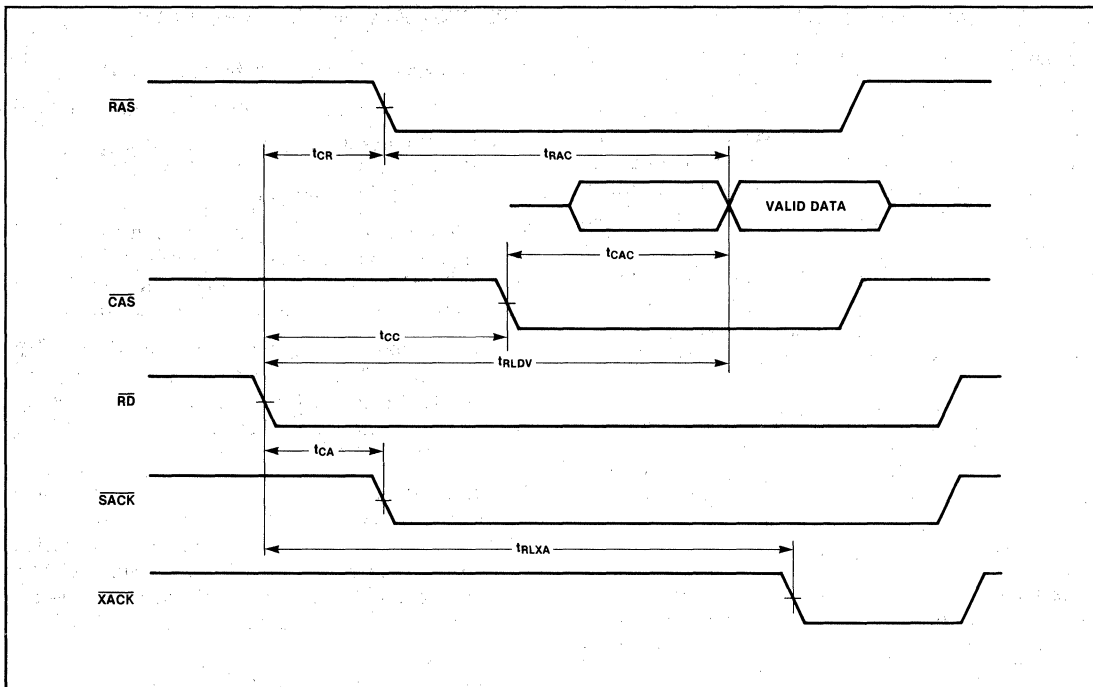


Figure 1. Read Cycle Timing

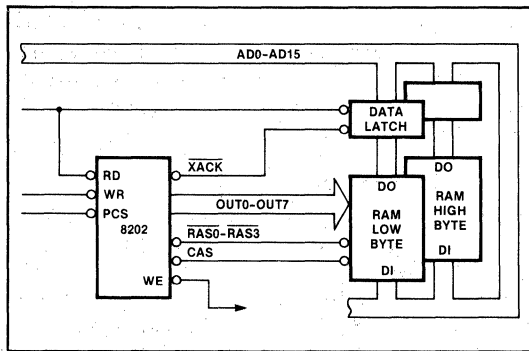


Figure 2. Basic Memory Architecture

Since the  $t_{CC}$  parameter is a function of the 8202's operating frequency, we can minimize  $t_{RLDV}$  by running the 8202 at the highest possible frequency. We can also reduce  $t_{RLDV}$  by choosing a RAM with a faster CAS access time. Table 1 shows the minimum  $t_{RLDV}$  for the various RAMs, based on the maximum 8202 frequency for that RAM. If system constraints force you to choose a slower 8202 operating frequency, then your  $t_{RLDV}$  will increase, which may slow down your CPU.

Table 1. 8202 System Timing

RAM TYPE	$t_{RAC}$	$t_{CAC}$	$t_{RLDV}^1$	$t_{CA}^1$	$t_{RLXA}^1$	$t_{CC, MIN}^{1,2}$	$f_{MAX}$
2118-3	100	55	280	80	445	125	25.00
2118-4	120	65	290	80	445	125	25.00
2118-7	150	80	310	81	456	126	24.24
2117-2	150	100	325	80	445	125	25.00
2117-3	200	135	361	80	449	126	24.69
2117-4	250	165	407	85	488	136	22.22
UNITS	nsec	nsec	nsec	nsec	nsec	nsec	MHz

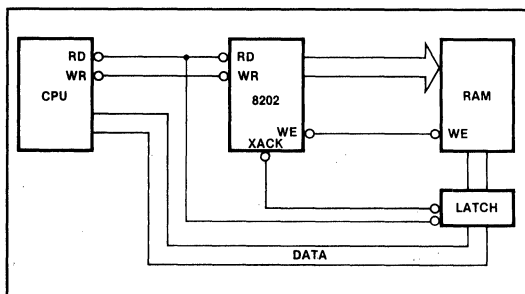
1. ASSUMES 8202 OPERATED AT  $f_{MAX}$ .  
 2. ADD  $t_p + 60$  nsec TO DERIVE  $t_{CC, MAX}$ .

Read Access Time (Min Mode)

In order to operate with no WAIT states, our memory system shown in Figure 3 must guarantee valid READ data within:

$$t_{RLDV} \leq 2 t_{CLCL} (\text{CPU}) - t_{CLRL, MIN} (\text{CPU}) - t_{DVCL, MIN} (\text{CPU})$$

Table 2 lists these times for various CPUs, operating at 5 MHz and 8 MHz. If your system has any additional buffers for the  $\overline{RD}$ ,  $\overline{WR}$ , or data lines, you must subtract their propagation delay from the  $t_{RLDV}$  values in Table 1 to derive your system access requirements.



**Figure 3. Unbuffered Min Mode Configuration**

Now we merely have to compare our system requirements from Table 2 with the memory access times in Table 1 to see if our RAM speed selection and 8202 operating frequency will work in our system with no WAIT states. If our system requirement is too fast for our first choice memory configuration, we will have to choose either a faster RAM, or add WAIT states.

If you choose to run with WAIT states, you can multiply the CPU clock period by the number of WAIT states, and add this number to the  $t_{RLDV}$  values in Table 2 to derive your new CPU access requirements. For example, a 5 MHz 8086 Min Mode configuration with no WAIT states requires a read access time of 205 nanoseconds, while the same configuration would need an access time of only 405 nanoseconds if we operate with one WAIT state. If we examine Table 1, we see that there is no RAM configuration which can run without WAIT states in a 5 MHz 8086 configuration; but we can use any 2118 speed selection if we operate with one WAIT state.

**Table 2. Read Cycle Access Requirements**

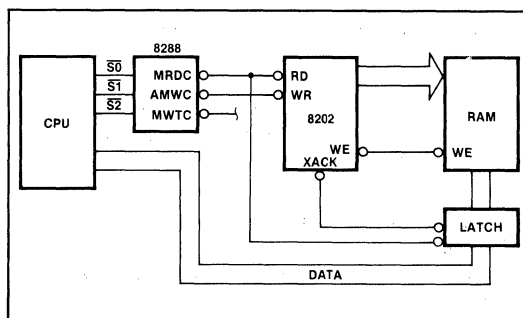
	MIN MODE			MAX MODE			
	8086-2	8086	8088	8086-2	8086	8088	8089
$t_{CPU}$	8 MHz	5 MHz	5 MHz	8 MHz	5 MHz	5 MHz	5 MHz
$t_{RLDV}$	130	205	205	195	335	335	335
$t_{RLDV_1}$	209	345	345	209	345	345	415

### Read Access Time (Max Mode)

Figure 4 illustrates the standard unbuffered Max Mode configuration. In order to run with no WAIT states, we need to guarantee a READ access time of

$$t_{RLDV} = 2 t_{CLCL} (\text{CPU}) - t_{CLML, MAX} (8288) - t_{DVCL, MIN} (\text{CPU})$$

These times are listed in Table 2 for the various CPUs, operating at 5 MHz and 8 MHz. If your system has buffers on the  $\overline{RD}$ ,  $\overline{WR}$ , or data lines, you must *subtract* their propagation delay from the  $t_{RLDV}$  values in Table 2.

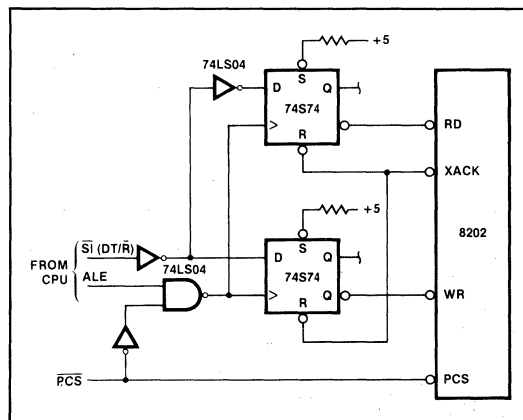


**Figure 4. Unbuffered Max Mode Configuration**

If we compare the Max Mode  $t_{RLDV}$  requirements in Table 2 with the Min Mode requirements, we notice that the Max Mode can run with a memory configuration that is 130 nsec slower than the memory required for an equivalent Min Mode configuration! In fact, all 5 MHz Max Mode CPU configurations can operate with no WAIT states in any of the 2118 memory configurations. Even the 8 MHz 8086-2 configuration can operate with all but one RAM configuration (2117-4) with only one WAIT state. This illustrates the advantage of using the Max Mode and the 8288 to generate the  $\overline{RD}$  signal for the 8202 when compared with the lower chip count Min Mode.

### READ Access Time (Alternate Configuration)

We can reduce our memory speed requirement even further by using the Alternate Configuration shown in Figure 5. This circuit uses the CPU status information to generate 8202  $\overline{RD}$  and  $\overline{WR}$  commands which start earlier in the memory cycle than either the Min Mode or the Max Mode. As previously stated, the Alternate Configuration can be used when the CPU is strapped in either the Min Mode or the Max Mode.



**Figure 5. Alternate Configuration**

# APPLICATIONS

The access requirement for the Alternate Configuration is

$$t_{RLDVI} = 2 t_{CLCL} (\text{CPU}) + t_{CHCL, \text{MIN}} (\text{CPU}) \\ - t_{CHLL, \text{MAX}} (\text{CPU}) - t_{CVCL, \text{MIN}} (\text{CPU}) \\ - t_{PHL} (74S74)$$

Table 2 lists these times for various CPUs operating at 5 MHz and 8 MHz. Note that in every configuration except one (8086-2 Max Mode Alternate Configuration), the memory READ access requirement is reduced. Hence, we can use slower memories when we use the Alternate Configuration in all but this one configuration. In general, the Alternate Configuration offers a significant advantage over the standard Min Mode, but little advantage over the standard Max Mode for READ cycles.

## 8202 Write Cycle Timing

8202 WRITE cycles have timing similar to READ cycles, except the  $\overline{WE}$  line is pulsed for the WRITE cycle; all other signal times are the same as a READ cycle.

There are three types of dynamic RAM Write cycles:

- Early Write
- Late Write
- Read-Modify-Write (RMW)

If  $\overline{WE}\downarrow$  precedes  $\overline{CAS}\downarrow$  by at least  $t_{WCS}$ , then the cycle is an early-write cycle, and the RAM data output will remain in its high-impedance state for the duration of the cycle. If  $\overline{WE}\downarrow$  occurs later, then the data output can leave its high-impedance state, and must be isolated from the CPU data bus via the RAM data latch used for READ cycles.

The dynamic RAM uses the  $\overline{WE}$  and  $\overline{CAS}$  signals to strobe the WRITE data into the RAM. If  $\overline{WE}\downarrow$  precedes  $\overline{CAS}\downarrow$ , as shown in Figure 6, then the data setup and hold times are measured relative to  $\overline{CAS}\downarrow$ , and the cycle is called an "early-WRITE" cycle. If the CPU starts a WRITE cycle by driving  $\overline{WR}$  active, and does not let  $\overline{WR}$  go inactive until  $\overline{CAS}\uparrow$ , then the 8202 will always perform an early-WRITE cycle.

The following three sections will illustrate 8202 WRITE cycles for Min Mode, Max Mode, and Alternate Con-

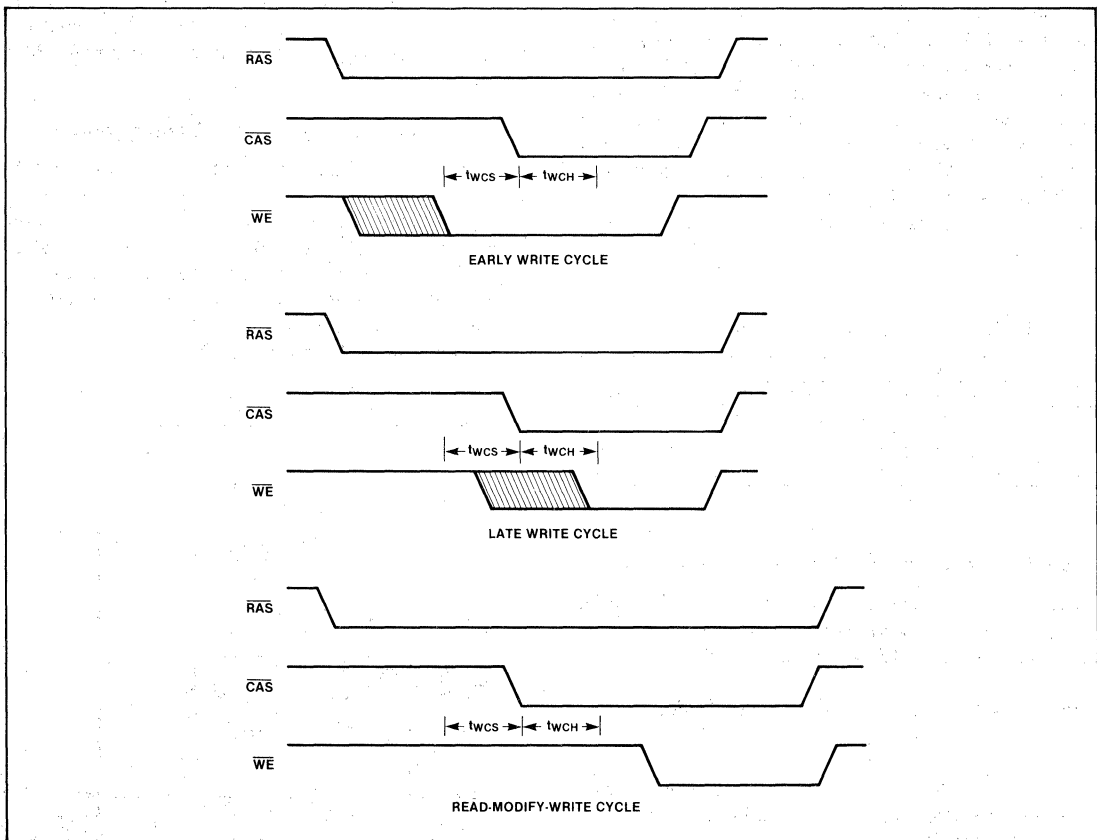


Figure 6. Write Cycle Timing

# APPLICATIONS

figurations. In most cases, we will use the 8202 early-WRITE configuration. However, in some cases, we will use a late-WRITE or READ-modify-WRITE 8202 configuration to prevent the RAM from storing the wrong information from the CPU's multiplexed address/data bus. In each case, we will examine the early-WRITE configuration first to see if the WRITE cycle will have adequate data setup and hold time for the RAM.

The  $t_{WLDV}$  data setup times for the various CPU configurations are listed in Table 3. In order to use the early-WRITE configuration of the 8202, we must guarantee:

$$t_{WLDV, MIN} (CPU) + t_{CC, MIN} (8202) = t_{DS, MIN} (RAM) \quad \text{(Equation 1)}$$

If this condition is not met by our system, we will have to either

- 1) Delay the 8202  $\overline{WR}$  signal, which will cause  $\overline{CAS} \downarrow$  to occur later in the bus cycle; or
- 2) Disconnect the 8202  $\overline{WE}$  signal from the RAM, and generate a delayed  $\overline{WE}$ , that goes active after the WRITE data becomes valid, but not so late in the cycle that the RAM's  $t_{CWL}$  parameter is violated.

The first alternative can be accomplished using the circuits shown in Figure 7. Both of these circuits will increase the  $t_{WLDV, MIN}$  values listed in Table 3 by an amount of time based on the CPU clock.

Figure 8 illustrates a method of generating a delayed  $\overline{WE}$  for the RAM using the 8288  $\overline{MWTC}$  signal, while starting the WRITE cycle with the 8288  $\overline{AMWC}$  signal. This circuit has the advantage of allowing the 8202 to respond with  $\overline{SACK} \downarrow$  early enough to prevent any WAIT states in certain configurations. For this reason, the second alternative (Figure 8) is more favorable over the first configuration.

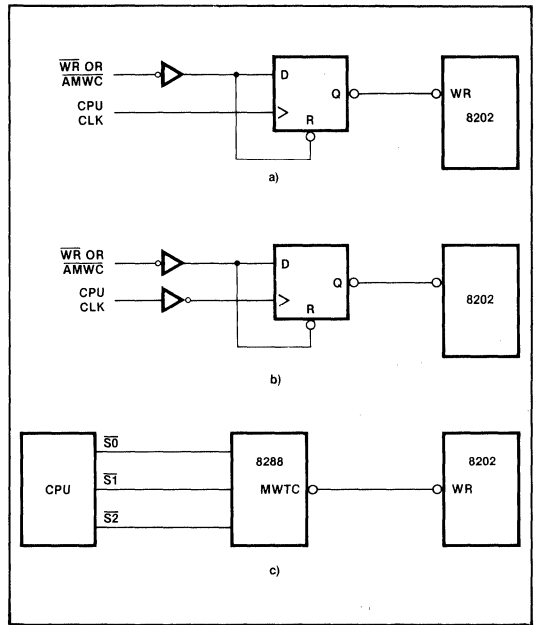


Figure 7. Generating a Delayed  $\overline{WR}$  for 8202

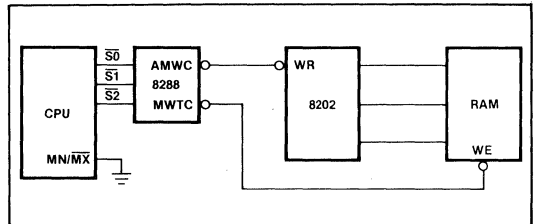


Figure 8. Using  $\overline{AMWC}$  and  $\overline{MWTC}$  for WRITE Cycles

**Table 3. Data Setup and Hold Times for Write Cycles**

	8086-2 MAX MODE <sup>1</sup> 8 MHz		8086-2 MIN MODE 8 MHz		8086-2 ALT. CONF. 8 MHz		8086 MAX MODE <sup>1</sup> 5 MHz		8086 MIN MODE 5 MHz		8086 ALT. CONF. 5 MHz	
	MIN	MAX	MIN	MAX	MIN	MAX	MIN	MAX	MIN	MAX	MIN	MAX
$t_{WLDV}$	-50 <sup>1</sup>	25 <sup>1</sup>	-50	60	-45	60	-100 <sup>1</sup>	25 <sup>1</sup>	-100	100	-70	85
$t_{DHADV}$	265		265		265		420		420		420	

	8088 MAX MODE <sup>1</sup> 5 MHz		8088 MIN MODE 5 MHz		8088 ALT. CONF. 5 MHz		8089 MAX MODE <sup>1</sup> 5 MHz		8089 ALT. CONF. 5 MHz	
	MIN	MAX	MIN	MAX	MIN	MAX	MIN	MAX	MIN	MAX
$t_{WLDV}$	-100 <sup>1</sup>	20 <sup>1</sup>	-100	100	-70	85	-100 <sup>1</sup>	25 <sup>1</sup>	-70	85
$t_{DHADV}$	420		420		420		420		420	

<sup>1</sup> ASSUMES  $\overline{AMWC}$  USED. ADD  $t_{CLCL}$  IF  $\overline{MWTC}$  USED.

# APPLICATIONS

We also need to check the system data hold times for each of the configurations to insure that the RAM's data hold time ( $t_{DH}$ ) is met. Table 3 lists the  $t_{DHADV}$  (data hold after data valid) times for each of the various CPU configurations. In order to meet the RAM's data hold time, we must guarantee:

$$t_{DHADV, MIN} \geq t_{WLDV, MAX} + t_{CC, MAX} + t_{DH, MAX} \quad (\text{Equation 2})$$

If this condition is not met, you will have to either insert WAIT states to extend the  $t_{DHADV, MAX}$  values, or generate your  $\overline{WE}$  transition earlier in the bus cycle. The latter can be performed by using one of the alternatives listed above, provided the RAM's data setup time is still met.

## WRITE Cycle Timing (Min Mode)

If we examine  $t_{CC, MIN}$  for various 8202 clock frequencies, we find that  $t_{CC} \geq 125$  nsec. Comparing this result with Table 3, we see that the condition required by Equation 1 is always met for the unbuffered Min Mode configuration, but just barely. For example, suppose we had a 5 MHz 8086 Min Mode system using 2118-3's ( $t_{DS} = 0$  nsec) with a 25 MHz 8202; Equation 1 becomes:

$$\begin{aligned} t_{WLDV, MIN} + [(t_{PH} + 2t_P + 25) - t_{DS}] \\ = -100 + [(20 + 80 + 25) - 0] \\ = 25 \text{ nsec} \end{aligned}$$

Therefore, we would have 25 nsec of data setup margin, so we can use the 8202 early-WRITE configuration.

Suppose our configuration had a data bus buffer as shown in Figure 9a. We would have to subtract the propagation delay of this buffer from Equation 1; if this delay is greater than 25 nanoseconds, then we cannot use the 8202 early-WRITE configuration.

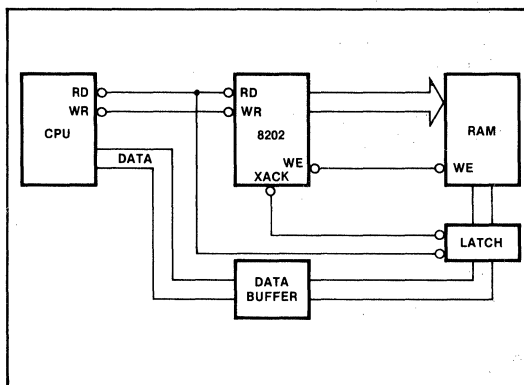


Figure 9a. Buffered Min Mode Configuration

Next, we need to examine our system to insure the RAM's data hold time is met. For the same system configuration, we can solve Equation 2:

$$\begin{aligned} t_{DHADV, MIN} &\geq t_{WLDV, MAX} + t_{CC, MAX} + t_{DH} \\ &= 100 + (t_{PH} + 3t_P + 85) + 25 \\ &= 350 \text{ nsec} \end{aligned}$$

Examining Table 3, we see that Equation 2 is satisfied with 70 nsec of margin. If we added any data buffers to our system, this margin would *increase*; if we buffer the 8202  $\overline{WR}$  input, this margin will decrease.

## Write Cycle Timing (Max Mode)

Let's see if we can get an 8 MHz 8086-2 Max Mode system to operate with 2118-4's in a 25 MHz 8202 early-WRITE configuration as shown in Figure 9b. Solving equation 1, we find:

$$\begin{aligned} t_{WLDV} + [(t_{PH} + 2t_P + 25) - t_{DS}] - t_{IVOV} (8286) \\ = -50 + [(20 + 80 + 25) - 0] - 35 \\ = 40 \text{ nsec} \end{aligned}$$

so we see we can use the early-WRITE configuration. Had we chosen a 5 MHz 8086 Max Mode configuration, we see from Table 3 that  $t_{WLDV}$  would decrease by 50 nanoseconds, which means we would not have adequate data set-up time.

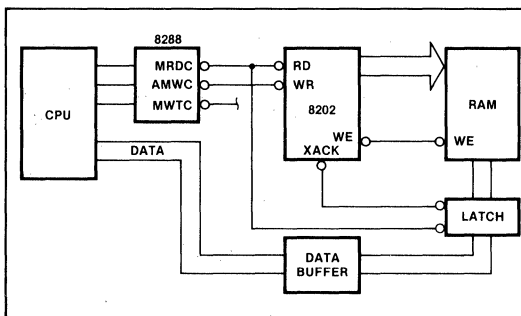


Figure 9b. Buffered Max Mode Configuration

## Write Cycle Timing (Alternate Configuration)

In general, the Alternate Configuration offers little advantage over the standard Min Mode or Max Mode, except in the earlier generation of the READY acknowledge signals,  $\overline{SACK}$  and  $\overline{XACK}$ . In some cases, using the Alternate Configuration will force you to generate a delayed  $\overline{WE}$  signal, since the 8202  $\overline{WR}$  signal goes active earlier in the bus cycle for the Alternate Configuration than it does for either the Min Mode or the Max Mode. So, unless you need to speed up your READY signal to reduce unnecessary WAIT states for WRITE cycles, the Alternate Configuration for WRITE cycles may not offer any performance advantage for your system.

# APPLICATIONS

## Handling 8-Bit Write Cycles in 16-Bit Systems

Systems which perform 8-bit WRITE cycles in a 16-bit memory array require a slight modification of the  $\overline{WE}$  control described previously. The memory must be broken into two 8-bit arrays with separate  $\overline{WE}$  control, as shown in Figure 10.

CPU signals  $A_0$  and  $\overline{BHEN}$  determine the type of bus cycle to be performed. If  $A_0=0$ , then the even byte is transferred on AD0-7; if  $\overline{BHEN}$  is active, then the odd byte is transferred on AD8-AD15. A word transfer is performed when  $A_0=0$  and  $\overline{BHEN}$  is active.

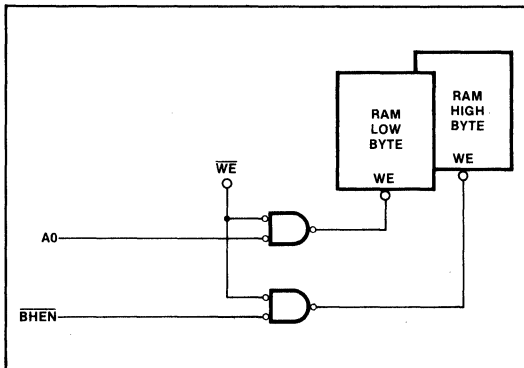


Figure 10. 8/16-Bit Write Cycle Control

## Multibus Byte Swap

To permit compatibility with existing 8-bit CPU boards, the Multibus specification requires all byte transfers to occur on the 8 least significant data lines ( $\overline{DAT0-DAT7}$ ). Figure 11 illustrates how to handle 8-bit and 16-bit bus cycles in a Multibus environment.

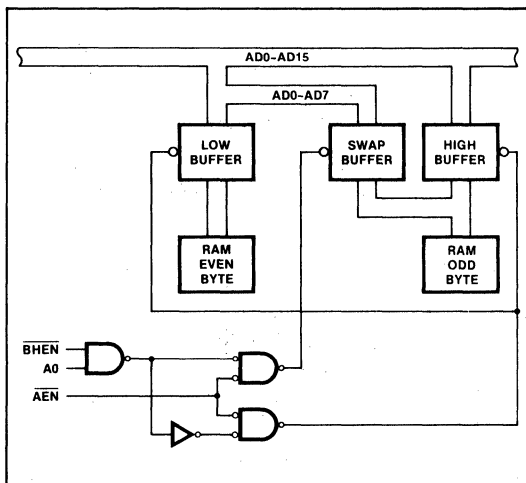


Figure 11. Multibus Byte Swap

Although Multibus uses the signals  $\overline{BHEN}$  and  $A_0$ , their meaning is slightly different than the CPU pin definitions. If  $\overline{BHEN}$  is inactive, then the bus cycle is always 8 bits, and always uses  $\overline{DAT0-DAT7}$ ; if  $\overline{BHEN}$  is active, then the cycle is always 16-bits, and  $A_0=0$ .

The High Byte Buffer and the Low Byte Buffer are enabled for all word transfers, and for all byte transfers to an even address. The Swap Byte Buffer is enabled only for byte transfers to an odd address.

This control logic will allow a 16-bit memory board to be compatible with both 8-bit CPU boards and 16-bit CPU boards.

## PCS Generation

In order to start a memory cycle, the 8202 requires its  $\overline{PCS}$  input, as well as  $\overline{RD}$  or  $\overline{WR}$ , to be active. Once a memory cycle is started, the 8202 will complete it, even if  $\overline{PCS}$  goes inactive. This feature can be used for battery-backup RAM designs. If you have a battery backed up design,  $\overline{RD}$ ,  $\overline{WR}$  and  $\overline{PCS}$  should be pulled up to the battery supply.

Normal decoding of the processor address bus can be used to generate  $\overline{PCS}$ . Since combinational logic (or even a bipolar PROM) is typically used to generate  $\overline{PCS}$ ; you must examine your system timing to make sure  $\overline{PCS}$  is stable before  $\overline{RD}$  or  $\overline{WR}$  goes active. If your decoding time is greater than the address set-up to command time, then two things can happen:

- 1) Your memory cycle will not start until  $\overline{PCS}$  goes active.
- 2) You may cause the 8202 to start an unwanted cycle due to a decoder glitch.

Remember, your decoding time is the amount of time it takes to ensure that only one device is selected, and that all other devices are deselected. Your decoder outputs may change after an address transition, and will only be stable after the decoding time has expired.

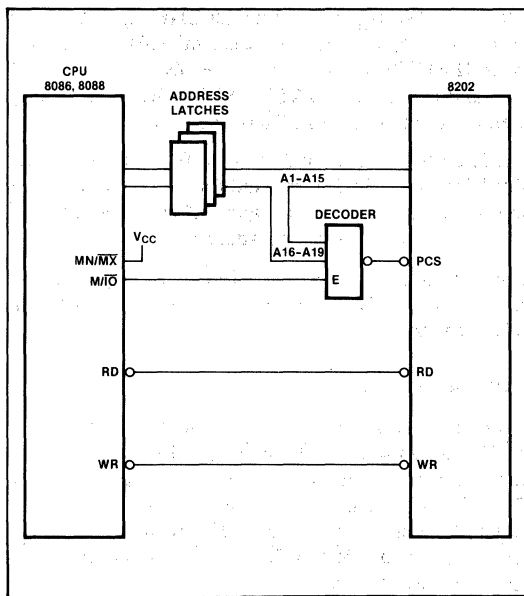
## Ready Handshake Signals ( $\overline{SACK}$ and $\overline{XACK}$ )

If our dynamic memory system was always available when the CPU requested a memory cycle, then we could generate our  $\overline{RAMRDY}$  signal as shown in Figure 13; if our RAM required no WAIT states, we could tie the  $\overline{READY}$  line high, assuming the I/O and the rest of the memory (e.g., PROM) did not need any WAIT states.

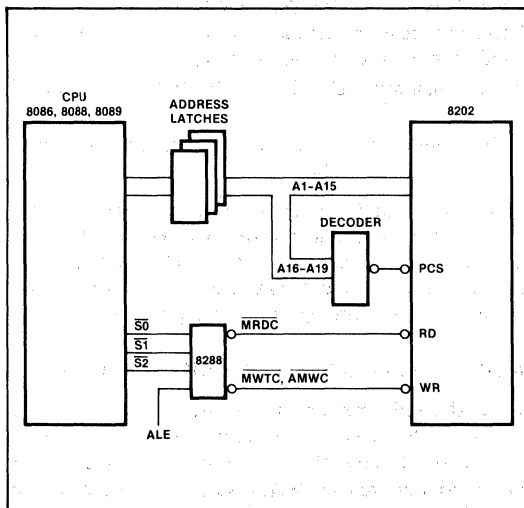
In systems which use asynchronous refresh, we cannot use these methods of  $\overline{READY}$  generation, since extra WAIT states are needed when a memory cycle is requested while refresh is in progress. This CPU holdoff can be performed using the 8202  $\overline{XACK}$  and  $\overline{SACK}$  signals.

$\overline{XACK}$  is a Multibus compatible acknowledge handshake signal, since it only goes active after READ data is valid, and after WRITE data has been latched.  $\overline{XACK}$





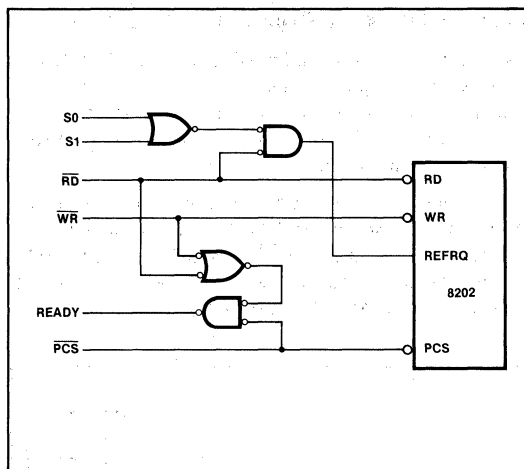
**Figure 12a. 8202 PCS Generation — Minimum Mode**



**Figure 12b. 8202 PCS Generation — Maximum Mode**

can be connected to the CPU's Ready input through an inverter.

Since most CPUs sample READY 1-2 clocks ahead of the time they sample data, XACK may cause more WAIT states than you really need; if your system has sufficient time between the READY sample and the data sample, SACK can be used.



**Figure 13. RAM RDY Generation (Transparent Refresh)**

If a memory cycle is requested while the 8202 is idle, SACK will occur 200-320 nanoseconds before data is valid. If the time between the CPU's first READY sample point and the first data sample point is greater than the difference in READ access time (either  $t_{RLDV}$  or  $t_{RLDV1}$ , depending on your configuration) and the  $t_{CA,MAX}$  time for your 8202 configuration, then SACK can be used to generate the CPU's READY signal. Otherwise, a delayed form of SACK (or XACK) should be used to generate the READY signal.

### Refresh Considerations

The 8202 has an internal timer which generates refresh requests every 12-16 microseconds, unless it is reset by an external refresh request. Thus, if the 8202 REFRQ is pulsed faster than 12 microseconds, we can reduce, or even eliminate, the amount of refresh interference with memory cycles. If, for any reason, our REFRQ signal fails to generate a pulse frequently enough, then the internal refresh timer will take over.

The standard hidden refresh circuit shown in Figure 13 can be used to eliminate all refresh interference in 8085A systems if the following condition is met:

$$4T (8085A) \geq t_{CR,MAX} (8202) + 2t_{RC,MAX} (8202)$$

where T is the CPU clock period. Remember, the 8202 internal refresh timer will automatically insert refresh requests if the CPU goes idle for extended periods of time (such as bursts of DMA cycles, or entering the HALT state).

Since the 8086 and 8088 pre-fetch instructions, the hidden refresh method shown in Figure 13 is not as useful as it is for 8085A systems. The 8086 Family CPUs will

# APPLICATIONS

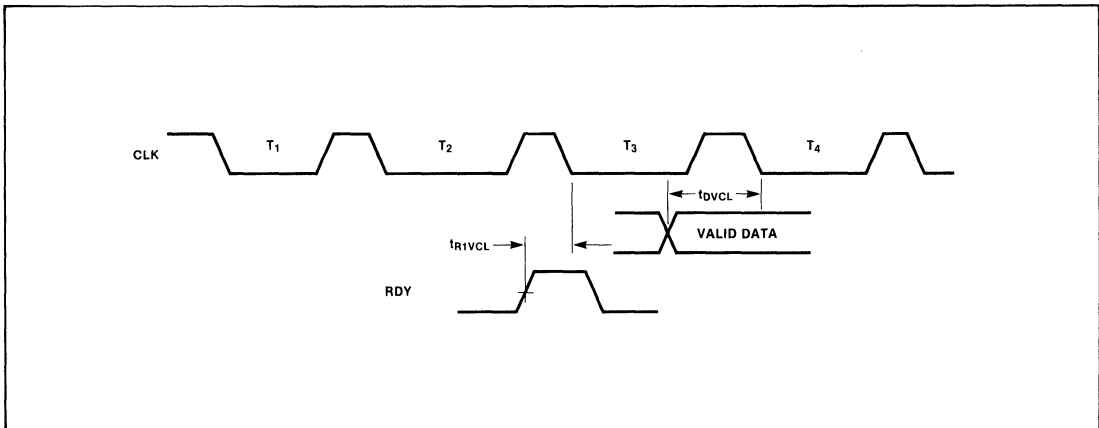


Figure 14. 8086 Family Data and READY Sample Points

have to use the 8202 internal refresh, and suffer the 3-7% performance degradation due to refresh interference. In reality, the performance degradation will be even less, since the instruction queue is normally full, and there is less chance of refresh interference than in an 8085A system.

## References

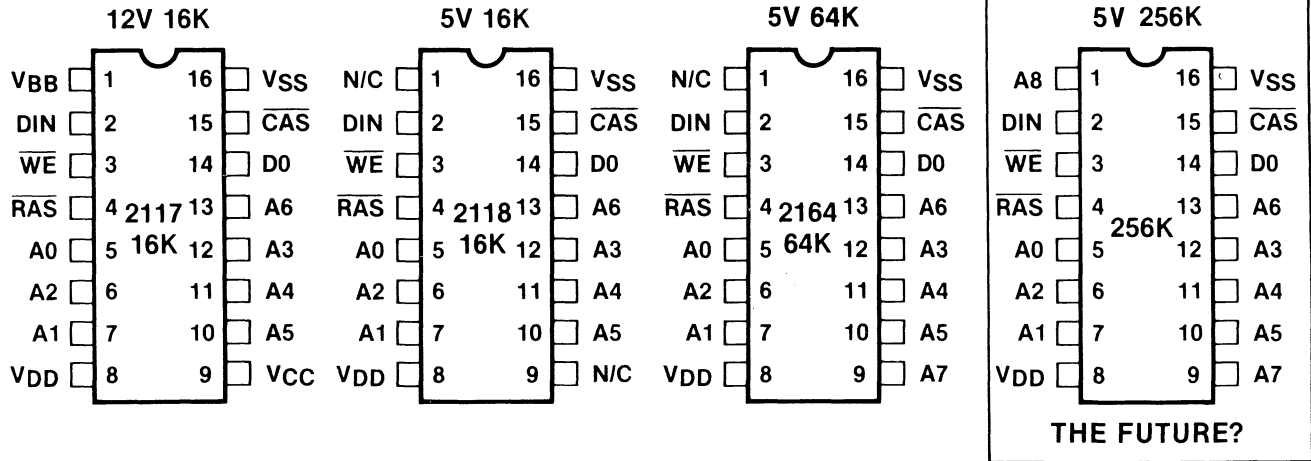
Intel Component Data Catalog  
Intel Peripheral Design Handbook  
Intel 8202 Data Sheet  
Intel Memory Design Handbook  
Intel 8086 Family User's Manual

# APPLICATIONS

---

## APPENDIX 1

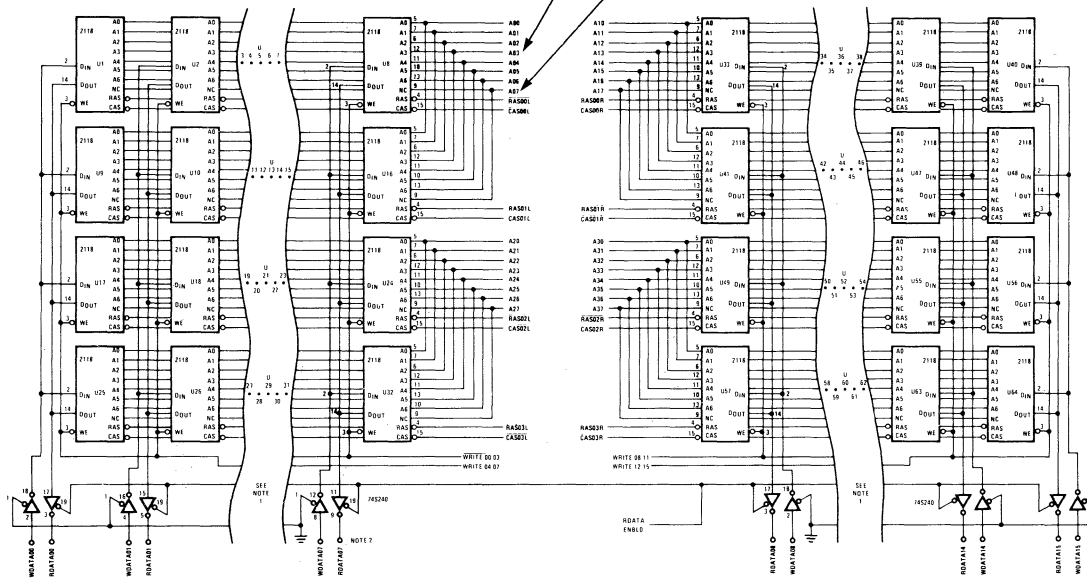
# PIN FUNCTION EVOLUTION



**APPENDIX 2**

# 2118/2164 MEMORY SYSTEM DESIGN

- ADDRESS BUFFERS DRIVE 16 MEMORY DEVICES
- A7 (FOR 64K RAM) CONNECTED TO MEMORY DEVICES



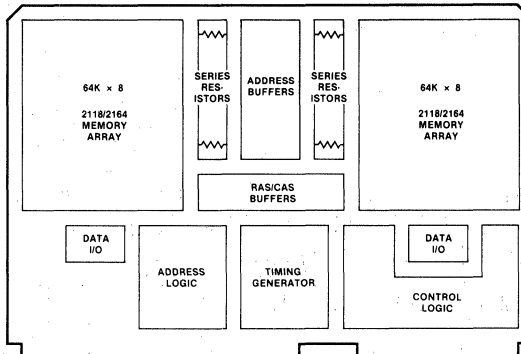
NOTES  
 1. FOR SIMPLICITY 2118 AND 74S20 DEVICES COMPRISING DATA BITS 02 BY 09 13 ARE NOT ILLUSTRATED.  
 2. I/O INTERCONNECTIONS FOR MDATAFX AND RDATAFX ARE LISTED IN TABLE 1.

APPLICATIONS

# APPLICATIONS

## 2118/2164 MEMORY SYSTEM DESIGN

### MEMORY SYSTEM COMPONENT PLACEMENT

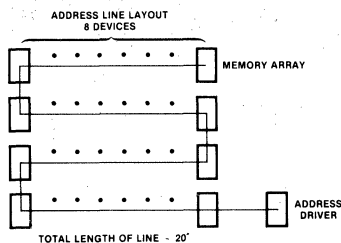


### P.C.B. LAYOUT — PITFALLS

- CONTROL LOGIC NOT CENTRALIZED
- LONG SIGNAL TRACES
  - RINGING
  - PROPAGATION DELAY ( $\approx 2\text{ns/ft}$ )
- NON-OPTIMAL ADDRESS/CONTROL LINE LAYOUT

### P.C.B. LAYOUT — PITFALLS

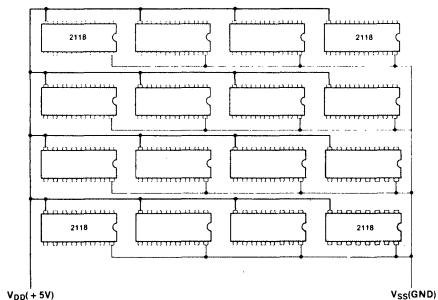
#### UNACCEPTABLE ADDRESS LINE ROUTING (SERPENTINE)



# APPLICATIONS

## POWER DISTRIBUTION

## UNACCEPTABLE GRIDDING



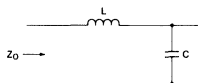
## FORMULAS!

$Z_0$  = TRANSMISSION LINE IMPEDANCE

$$= \sqrt{\frac{L}{C}}$$

L = INDUCTANCE/  
UNIT LENGTH  
C = CAPACITANCE/  
UNIT LENGTH

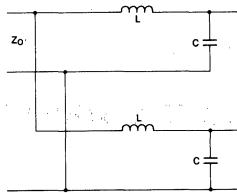
SO  
EQUIVALENT TRANSMISSION LINE CIRCUIT





# APPLICATIONS

## CONNECTING TWO LINES IN PARALLEL

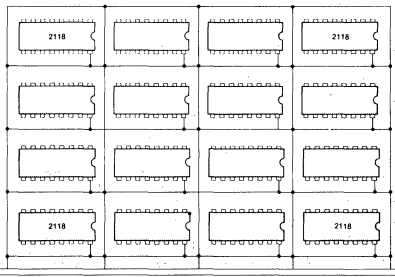


$$\text{AND } Z_0 = \sqrt{\frac{L}{2}} = \frac{1}{2} \sqrt{\frac{L}{C}} = \frac{1}{2} Z_0$$

THE MORE YOU CONNECT —  
THE LOWER THE EFFECTIVE IMPEDANCE

## STEP 1

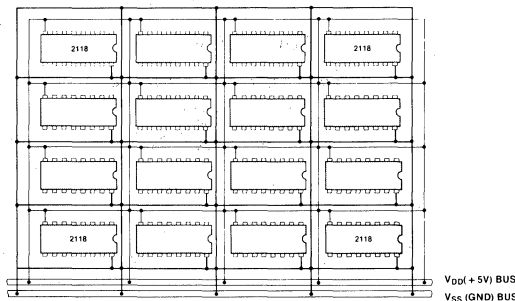
### GRIDDING $V_{SS}$ (GND)



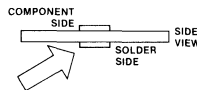
- TWO SIDED CARD
  - VERTICAL TRACES ON COMPONENT SIDE
  - HORIZONTAL TRACES ON SOLDER SIDE
- MAINGROUND BUS OR INTERCONNECTION TO TTL CONTROL, ADDRESS, DATA BUFFERS

## STEP 2

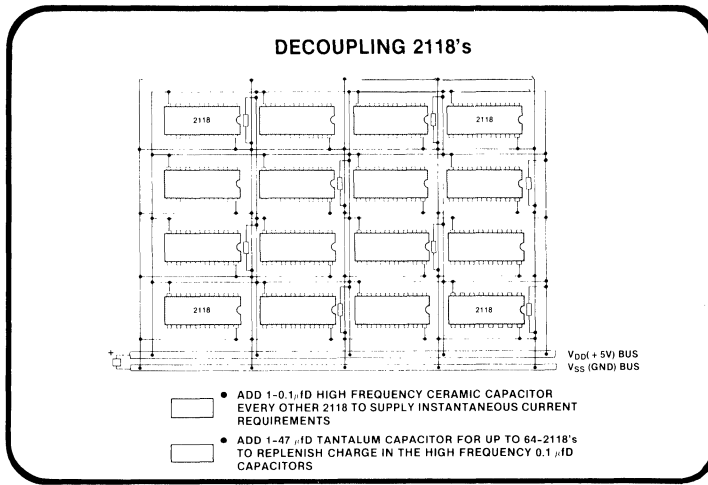
### GRIDDING $V_{DD}$ (+5V) AND $V_{SS}$ (GND)



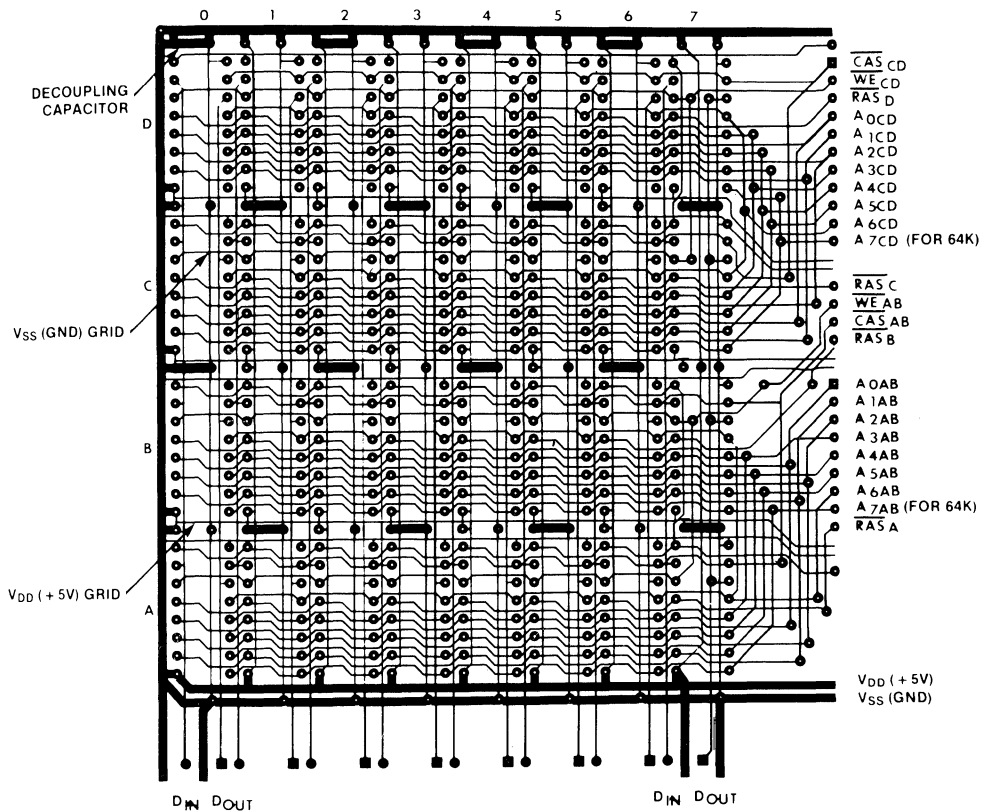
- STILL TWO SIDED CARD
- LOWER POWER AND GROUND DISTRIBUTION IMPEDANCE REDUCES CROSSTALK TO SIGNAL TRACES.
- IMPEDANCE BETWEEN POWER AND GROUND REDUCED BECAUSE TRACES ARE ABOVE EACH OTHER



# APPLICATIONS



## 2118/2164 MEMORY ARRAY P.C.B. LAYOUT POWER/GROUND GRID & DECOUPLING



**NOTE: MEMORY DEVICE SPACING IS 0.4" ONE TO ONE**

# APPLICATIONS

---

## APPENDIX 3

# APPLICATIONS

## POWER CONSIDERATIONS AND DECOUPLING

## DETERMINING 2118 POWER REQUIREMENTS

- NORMAL OPERATING CURRENT
- STANDBY CURRENT
- REFRESH CURRENT

- OPERATING CURRENT 2118 SYSTEM

$$I_{DDQ} = (I_{DD2} \times K) + I_{DDLO}$$

### WHERE

— K = NUMBER OF ACTIVE (RECEIVING BOTH  $\overline{RAS}$  &  $\overline{CAS}$ ) DEVICES IN THE SYSTEM

—  $I_{DD2}$  =  $V_{DD}$  SUPPLY CURRENT, OPERATING

2118-2 = 29 mA (MAX)

2118-3 = 25 mA (MAX)

2118-4 = 22 mA (MAX)

2118-7 = 22 mA (MAX)

—  $I_{DDLO}$  = 2118 OUTPUT LOAD CURRENT

Σ LEAKAGE CURRENTS + TTL LOAD INPUT CURRENT

2118 OUTPUT LEAKAGE  $I_{LOI}$  = 10  $\mu$ A

74S240 INPUT CURRENT  $I_{IL}$  = - 400  $\mu$ A

# APPLICATIONS

- STANDBY CURRENT 2118 SYSTEM

$$I_{DDs} = I_{DD1} \times M$$

WHERE

- M = NUMBER ON INACTIVE (RECEIVING  $\overline{\text{CAS}}$  ONLY) DEVICES IN THE SYSTEM
- $I_{DD1}$  =  $V_{DD}$  SUPPLY CURRENT, STANDBY.  
2118 = 3 mA (MAX)

- REFRESH CURRENT 2118 SYSTEM

$$I_{DDR} = (I_{DD3} \times N) \left( \frac{t_{RAS}}{t_{REF}} \right) (128)$$

WHERE

- N = TOTAL NUMBER OF DEVICES IN THE SYSTEM  
 $N = M + K$
- $I_{DD3}$  =  $V_{DD}$  SUPPLY CURRENT,  $\overline{\text{RAS}}$  ONLY CYCLE  
2118-2 = 24 mA  
2118-3 = 20 mA  
2118-4 = 18 mA  
2118-7 = 18 mA
- $t_{RAS}$  =  $\overline{\text{RAS}}$  PULSE WIDTH IN NANoseconds DURING THE REFRESH CYCLE
- $t_{REF}$  = TIME BETWEEN REFRESH IN MILLISECONDS

- TOTAL  $V_{DD}$  SUPPLY CURRENT 2118 SYSTEM

$$I_{DDT} = I_{DD0} + I_{DDs} + I_{DDR}$$

- TOTAL  $V_{DD}$  POWER

$$P_{DDT} = V_{DD} \times I_{DDT}$$

WHERE

$V_{DD} = 5.5V$  TO GET ABSOLUTE MAXIMUM SYSTEM POWER

# APPLICATIONS

● **ACTUAL POWER CALCULATION FOR A 64K × 16 BIT  
2118-4 MEMORY ARRAY:**

N = NUMBER OF 2118-4 IN THE SYSTEM = 64  
M = NUMBER OF STANDBY 2118-4 IN THE SYSTEM = 48  
K = NUMBER OF ACTIVE 2118-4 IN THE SYSTEM = 16  
t<sub>RAS</sub> = 140 ns  
t<sub>REF</sub> = 2 ms

$$I_{DD0} = (22 \text{ mA}) (16) + (4) (10 \text{ } \mu\text{A}) + 400 \text{ } \mu\text{A}$$

$$= 352 \text{ mA} + 0.440 \text{ mA} = 352.4 \text{ mA}$$

$$I_{DD5} = (3 \text{ mA}) (48) = 144 \text{ mA}$$

$$I_{DD6} = (18 \text{ mA}) (64) \left( \frac{140 \text{ ns}}{2 \text{ ms}} \right) (128)$$

$$= 10.3 \text{ mA}$$

$$I_{DD7} = 352.4 \text{ mA} + 144 \text{ mA} + 10.3 \text{ mA}$$

2118-4	2117-2
I <sub>DD1</sub> = 507 mA	I <sub>DD1</sub> = 632 mA
P <sub>DD1</sub> = 2.8 WATTS	P <sub>DD1</sub> = 8.4 WATTS

# APPLICATIONS

---

## An Intelligent Data Base System Using the 8272

### Contents

<b>INTRODUCTION</b>	<b>2-133</b>
The Floppy Disk	
The Floppy Disk Drive	
<b>SUBSYSTEM OVERVIEW</b>	<b>2-135</b>
Controller Electronics	
Drive Electronics	
Controller/Drive Interface	
Processor/Memory Interface	
<b>DISK FORMAT</b>	<b>2-136</b>
Data Recording Techniques	
Sectors	
Tracks	
Sector Interleaving	
<b>THE 8272 FLEXIBLE DISKETTE CONTROLLER</b>	<b>2-139</b>
Floppy Disk Commands	
Interface Registers	
Command/Result Phases	
Execution Phase	
Multi-sector and Multi-track Transfers	
Drive Status Polling	
Command Details	
<b>THE DATA SEPARATOR</b>	<b>2-154</b>
Single Density	
Double Density	
Phase-Locked Loop Design	
Initialization	
Floppy Disk Data	
Startup	
PLL Synchronization	
<b>AN INTELLIGENT DISKETTE DATA BASE SYSTEM</b>	<b>2-158</b>
Processor and Memory	
Serial I/O	
DMA	
Disk Drive Interface	
<b>SPECIAL CONSIDERATIONS</b>	<b>2-161</b>
<b>APPENDIX</b>	<b>2-163</b>
Schematics	
Power Distribution	

## 1. INTRODUCTION

Most microcomputer systems in use today require low-cost, high-density removable magnetic media for information storage. In the area of removable media, a designer's choice is limited to magnetic tapes and floppy disks (flexible diskettes), both of which offer non-volatile data storage. The choice between these two technologies is relatively straight-forward for a given application. Since disk drives are designed to permit random access to stored information, they are significantly faster than tape units. For example, locating information on a disk requires less than a second, while tape movement (even at the fastest rewind or fast-forward speed) often requires several minutes. This random access ability permits the use of floppy disks in on-line storage applications (where information must be located, read, and modified/updated in real-time under program or operator control). Tapes, on the other hand, are ideally suited to archival or back-up storage due to their large storage capacities (more than 10 million bytes of data can be archived on a cartridge tape).

A sophisticated controller is required to capitalize on the abilities of the disk storage unit. In the past, disk controller designs have required upwards of 150 ICs. Today, the single-chip 8272 Floppy Disk Controller (FDC) plus approximately 30 support devices can handle up to four million bytes of on-line data storage on four floppy disk drives.

### The Floppy Disk

A floppy disk is a circular piece of thin plastic material covered with a magnetic coating and enclosed in a protective jacket (Figure 1). The circular piece of plastic revolves at a fixed speed (approximately 360 rpm) within its jacket in much the same manner that a record revolves at a fixed speed on a stereo turntable. Disks are manufactured in a variety of configurations for various storage capacities. Two standard physical disk sizes are commonly used. The 8-inch disk (8 inches square) is the larger of the two sizes; the smaller size (5-1/4 inches square) is often referred to as a mini-floppy. Single-sided disks can record information on only one side of the disk, while double-sided disks increase the storage capacity by recording on both sides. In addition, disks are classified as single-density or double-density. Double-density disks use a modified recording method to store twice as much information in the same disk area as can be stored on a single-density disk. Table 1 lists storage capacities for standard floppy disk media.

A magnetic head assembly (in contact with the disk) writes information onto the disk surface and subsequently reads the data back. This head assembly can move from the outside edge of the disk toward the center in fixed increments. Once the head assembly is

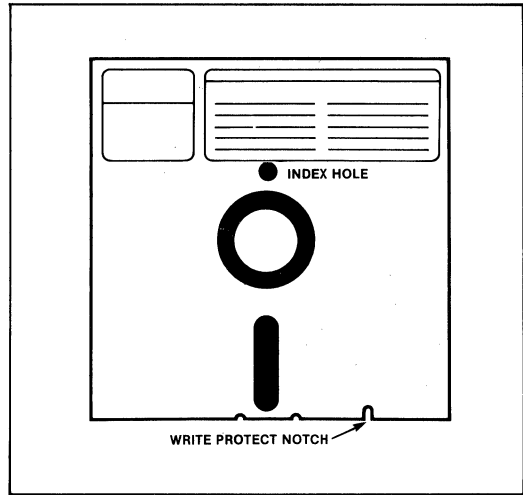


Figure 1. A Floppy Diskette

positioned at one of these fixed positions, the head can read or write information in a circular path as the disk revolves beneath the head assembly. This method divides the surface into a fixed number of cylinders (as shown in Figure 2). There are normally 77 cylinders on a standard disk. Once the head assembly is positioned at a given cylinder, data may be read or written on either side of the disk. The appropriate side of the disk is selected by the read/write head address (zero or one). Of course, a single-sided disk can only use head zero. The combination of cylinder address and head address uniquely specifies a single circular track on the disk. The physical beginning of a track is located by means of a small hole (physical index mark) punched through the plastic near the center of the disk. This hole is optically sensed by the drive on every revolution of the disk.

Table 1. Formatted Disk Capacities

Single-Density Format				
Byte/Sector	128	256	512	1024
Sectors/Track	26	15	8	4
Tracks/Disk	77	77	77	77
Bytes/Disk	256,256	295,680	315,392	315,392
Double-Density Format				
Bytes/Sector	128	256	512	1024
Sectors/Track	52	30	16	8
Tracks/Disk	77	77	77	77
Bytes/Disk	512,512	591,360	630,784	630,784



## APPLICATIONS

Each track is subdivided into a number of sectors (see detailed discussion in section 3). Sectors are generally 128, 256, 512, or 1024 data bytes in length. This track sectoring may be accomplished by one of two techniques: hard sectoring or soft sectoring. Hard sectored disks divide each track into a maximum of 32 sectors. The beginning of each sector is indicated by a sector hole punched in the disk plastic. Soft sectoring, the IBM standard method, allows software selection of sector sizes. With this technique, each data sector is preceded by a unique sector identifier that is read/written by the disk controller.

A floppy disk may also contain a write protect notch punched at the edge of the outer jacket of the disk. This notch is detected by the drive and passed to the controller as a write protect signal.

### The Floppy Disk Drive

The floppy disk drive is an electromechanical device that records data on, or reads data from, the surface of a floppy disk. The disk drive contains head control electronics that move the head assembly one increment (step) forward (toward the center of the disk) or backward (toward the edge of the disk). Since the recording head must be in contact with the disk material in order to read or write information, the disk drive also contains head-load electronics. Normally the read/write head is unloaded until it is necessary to read or write information on the floppy disk. Once the head assembly has been positioned over the correct track on the disk, the head is loaded (brought into contact with the disk). This sequence prevents excessive disk wear. A small time penalty is paid when the head is loaded. Approximately thirty to fifty milliseconds are needed before data may be reliably read from, or written to, the disk. This time is known as the head load time. If desired, the head may be moved from cylinder to cylinder while loaded. In this manner, only a small time interval (head settling time) is required before data may be read from the new cylinder. The head settling time is often shorter than the head load time. Typically, disk drives also contain drive select logic that allows more than one physical drive to be connected to the same interface cable (from the controller). By means of a jumper on the drive, the drive number may be selected by the OEM or end user. The drive is enabled only when selected; when not selected, all control signals on the cable are ignored.

Finally, the drive provides additional signals to the system controller regarding the status of the drive and disk. These signals include:

**Drive Ready** — Signals the system that the drive door is closed and that a floppy disk is inserted into the drive.

**Track Zero** — Indicates that the head assembly is located over the outermost track of the disk. This signal may be used for calibration of the disk drive at system initialization and after an error condition.

**Write Protect** — Indicates that the floppy disk loaded into the drive is write protected.

**Dual Sided** — Indicates that the floppy disk in the drive is dual-sided.

**Write Fault** — Indicates that an error occurred during a recording operation.

**Index** — Informs the system that the physical index mark of the floppy disk (signifying the start of a data track) has been sensed.

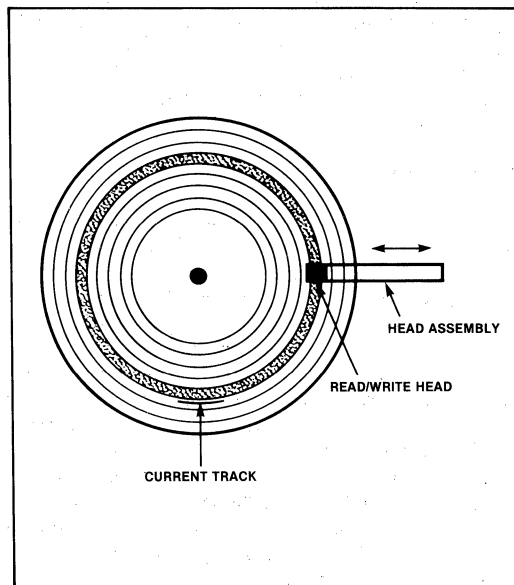


Figure 2. Concentric Cylinders on a Floppy Diskette

## 2. SUBSYSTEM OVERVIEW

A disk subsystem consists of the following functional electronic units:

1. Disk Controller Electronics
2. Disk Drive Electronics
3. Controller/Disk Interface (cables, drivers, terminators)
4. Controller/Microprocessor System Interface

The operation of these functional units is discussed in the following paragraphs.

### Controller Electronics

The disk controller is responsible for converting high-level disk commands (normally issued by software executing on the system processor) into disk drive commands. This function includes:

1. Disk Drive Selection — Disk controllers typically manage the operations of multiple floppy disk drives. This controller function permits the system processor to specify which drive is to be used in a particular operation.
2. Track Selection — The controller issues a timed sequence of step pulses to move the head from its current location to the proper disk cylinder from which data is to be read or to which data is to be written. The controller stores the current cylinder number and computes the stepping distance from the current cylinder to the specified cylinder. The controller also manages the head select signal to select the correct side of the floppy disk.
3. Sector Selection — The controller monitors the data on a track until the requested sector is sensed.
4. Head Loading — The disk controller determines the times at which the head assembly is to be brought in contact with the disk surface in order to read or write data. The controller is also responsible for waiting until the head has settled before reading or writing information. Often the controller maintains the head loaded condition for up to 16 disk revolutions (approximately 2 seconds) after a read or write operation has been completed. This feature eliminates the head load time during periods of heavy disk I/O activity.
5. Data Separation — The actual signal recorded on a floppy disk is a combination of timing information (clock) and data. The serial READ DATA input (from the disk drive) must be converted into two signal streams: clock and data. (The READ DATA input operates at 250K bits/second for single-density disks and 500K bits/second for double-density

disks.) The serial data must also be assembled into 8-bit bytes for transfer to system memory. A byte must be assembled and transferred every 32 microseconds for single-density disks and every 16 microseconds for double-density.

6. Error Checking — Information recorded on a floppy disk is subject to both hard and soft errors. Hard (permanent) errors are caused by media defects. Soft errors, on the other hand, are temporary errors caused by electromagnetic noise or mechanical interference. Disk controllers use a standard error checking technique known as a Cyclic Redundancy Check (CRC). As data is written to a disk, a 16-bit CRC character is computed and also stored on the disk. When the data is subsequently read, the CRC character allows the controller to detect data errors. Typically, when CRC errors are detected, the controlling software retries the failed operation (attempting to recover from a soft error). If data cannot reliably be read or written after a number of retries, the system software normally reports the error to the operator. Multiple CRC errors normally indicate unrecoverable media error on the current disk track. Subsequent recovery attempts must be defined by the system designers and tailored to meet system interfacing requirements.

Today, single-chip digital LSI floppy disk controllers such as the 8272 perform all the above functions with the exception of data separation. A data separation circuit (a combination of digital and analog electronics) synchronizes itself to the actual data rate of the disk drive. This data rate varies from drive to drive (due to mechanical factors such as motor tolerances) and varies from disk to disk (due to temperature effects). In order to operate reliably with both single- and double-density storage, the data separation circuit must be based on phase-locked loop (PLL) technology. The phase-locked loop data separation logic is described in section 5. The separation logic, after synchronizing with the data stream, supplies a data window to the LSI disk controller. This window differentiates data information from clock information within the serial stream. The controller uses this window to reconstruct the data previously recorded on the floppy disk.

### Drive Electronics

Each floppy disk drive contains digital electronic circuits that translate TTL-compatible command signals into electromechanical operations (such as drive selection and head movement/loading) and that sense and report disk or drive status to the controller (e.g., drive ready, write fault, and write protect). In addition, the drive electronics contain analog components to sense, amplify, and shape data pulses read from, or written to, the floppy disk surface by the read/write head.

## Controller/Drive Interface

The controller/drive interface consists of high-current line drivers, Schmitt triggered input gates, and flat or twisted pair cable(s) to connect the disk drive electronics to the controller electronics. Each interface signal line is resistively terminated at the end of the cable farthest from the line drivers. Eight-inch drives may be directly interfaced by means of 50-conductor flat cable. Generally, cable lengths should be less than ten feet in order to maintain noise immunity.

Normally, provisions are made for up to four disk drives to share the same interface cable. The controller may operate as many cable assemblies as practical. LSI floppy disk controllers typically operate one to four drives on a single cable.

## Processor/Memory Interface

The disk controller must interface to the system processor and memory for two distinct purposes. First, the processor must specify disk control and command parameters to the controller. These parameters include the selection of the recording density and specification of disk formatting information (discussed in section 3). In addition to disk parameter specification, the processor must also send commands (e.g., read, write, seek, and scan) to the controller. These commands require the specification of the command code, drive number, cylinder address, sector address, and head address. Most LSI controllers receive commands and parameters by means of processor I/O instructions.

In addition to this I/O interface, the controller must also be designed for high-speed data transfer between memory and the disk drive. Two implementation methods may be used to coordinate this data transfer. The lowest-cost method requires direct processor intervention in the transfer. With this method, the controller issues an interrupt to the processor for each data transfer. (An equivalent method allows the processor to poll an interrupt flag in the controller status word.) In the case of a disk write operation, the processor writes a data byte (to be encoded into the serial output stream) to the disk controller following the receipt of each controller interrupt. During a disk read operation, the processor reads a data byte (previously assembled from the input data stream) from the controller after each interrupt. The processor must transfer a data byte from the controller to memory or transfer a data byte from memory to the disk controller within 16 or 32 microseconds after each interrupt (double-density and single-density response times, respectively).

If the system processor must service a variety of other interrupt sources, this interrupt method may not be practical, especially in double-density systems. In this case, the disk controller may be interfaced to a Direct

Memory Access (DMA) controller. When the disk controller requires the transfer of a data byte, it simply activates the DMA request line. The DMA controller interfaces to the processor and, in response to the disk controller's request, gains control of the memory interface for a short period of time—long enough to transfer the requested data byte to/from memory. See section 6 for a detailed DMA interface description.

## 3. DISK FORMAT

New floppy disks must be written with a fixed format by the controller before these disks may be used to store data. Formatting is a method of taking raw media and adding the necessary information to permit the controller to read and write data without error. All formatting is performed by the disk controller on a track-by-track basis under the direction of the system processor. Generally, a track may be formatted at any time. However, since formatting "initializes" a complete disk track, all previously written data is lost (after a format operation). A format operation is normally used only when initializing new floppy disks. Since soft-sectoring in such a predominant formatting technique (due to IBM's influence), the following discussion will limit itself to soft-sectored formats.

### Data Recording Techniques

Two standard data recording techniques are used to combine clock and data information for storage on a floppy disk. The single-density technique is referred to as FM encoding. In FM encoding (see Figure 3), a double frequency encoding technique is used that inserts a data bit between two adjacent clock bits. (The presence of a data bit represents a binary "one" while the absence of a data bit represents a binary "zero.") The two adjacent clock bits are referred to as a bit cell, and except for unique field identifiers, all clock bits written on the disk are binary "ones." In FM encoding, each data bit is written at the center of the bit cell and the clock bits are written at the leading edge of the bit cell.

The encoding used for double-density recording is termed MFM encoding (for "Modified FM"). In MFM encoding (Figure 3) the data bits are again written at the center of the bit cell. However, a clock bit is written at the leading edge of the bit cell only if no data bit was written in the previous bit cell and no data bit will be written in the present bit cell.

### Sectors

Soft-sectored floppy disks divide each track into a number of data sectors. Typically, sector sizes of 128, 256, 512, or 1024 data bytes are permitted. The sector size is specified when the track is initially formatted by the controller. Table 1 lists the single- and double-

## APPLICATIONS

density data storage capacities for each of the four sector sizes. Each sector within a track is composed of the following four fields (illustrated in Figure 4):

1. Sector ID Field — This field, consisting of seven bytes, is written only when the track is formatted. The ID field provides the sector identification that is used by the controller when a sector must be read or written. The first byte of the field is the ID address mark, a unique coding that specifies the beginning of the ID field. The second, third, and fourth bytes are the cylinder, head, and sector addresses, respectively, and the fifth byte is the sector length code. The last two bytes are the 16-bit CRC character for the ID field. During formatting, the controller supplies the address mark. The cylinder, head, and sector addresses and the sector length code are supplied to the controller by the processor software. The CRC character is derived by the controller from the data in the first five bytes.
2. Post ID Field Gap — The post ID field gap (gap 2) is written initially when the track is formatted. During subsequent write operations, the drive's write circuitry is enabled within the gap and the trailing bytes of the gap are rewritten each time the sector is updated (written). During subsequent read operations, the trailing bytes of the gap are used to synchronize the data separator logic with the upcoming data field.
3. Data Field — The length (number of data bytes) of the data field is determined by software when the track is formatted. The first byte of the data field is the data address mark, a unique coding that specifies

the beginning of the data field. When a sector is to be deleted, (e.g., a hard error on the disk), a deleted data address mark is written in place of the data address mark. The last two bytes of the data field comprise the CRC character.

4. Post Data Field Gap — The post data field gap (gap 3) is written when the track is formatted and separates the preceding data field from the next physical ID field on the track. Note that a post data field gap is not written following the last physical sector on a track. The gap itself contains a program-selectable number of bytes. Following a sector update (write) operation, the drive's write logic is disabled during the gap. The actual size of gap 3 is determined by the maximum number of data bits that can be recorded on a track, the number of sectors per track and the total sector size (data plus overhead information). The gap size must be adjusted so that it is large enough to contain the discontinuity generated on the floppy disk when the write current is turned on or off (at the start or completion of a disk write operation) and to contain a synchronization field for the upcoming ID field (of the next sector). On the other hand, the gaps must be small enough so that the total number of data bits required on the track (sectors plus gaps) is less than the maximum number of data bits that can be recorded on the track. The gap size must be specified for all read, write, and format operations. The gap size used during disk reads and writes must be smaller than the size used to format the disk to avoid the splice points between contiguous physical sectors. Suggested gap sizes are listed in Table 9.

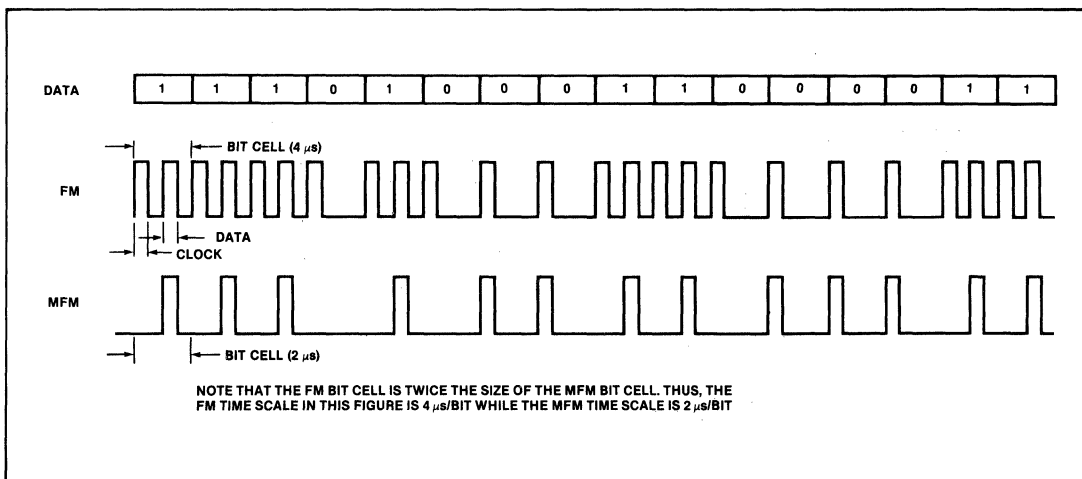


Figure 3. FM and MFM Encoding

# APPLICATIONS

## Tracks

The overall format for a track is illustrated in Figure 4. Each track consists of the following fields:

1. Pre-Index Gap — The pre-index gap (gap 5) is written only when the track is formatted.
2. Index Address Mark — The index address mark consists of a unique code that indicates the beginning of a data track. One index mark is written on each track when the track is formatted.
3. Post Index Gap — The post index gap (gap 1) is used during disk read and write operations to syn-

- chronize the data separator logic with the data to be read from the ID field (of the first sector). The post index gap is written only when the disk is formatted.
4. Sectors — The sector information (discussed above) is repeated once for each sector on the track.
5. Final Gap — The final gap (gap 4) is written when the track is formatted and extends from the last physical data field on the track to the physical index mark. The length of this gap is dependent on the number of bytes per sector specified, the lengths of the program-selectable gaps specified, and the drive speed.

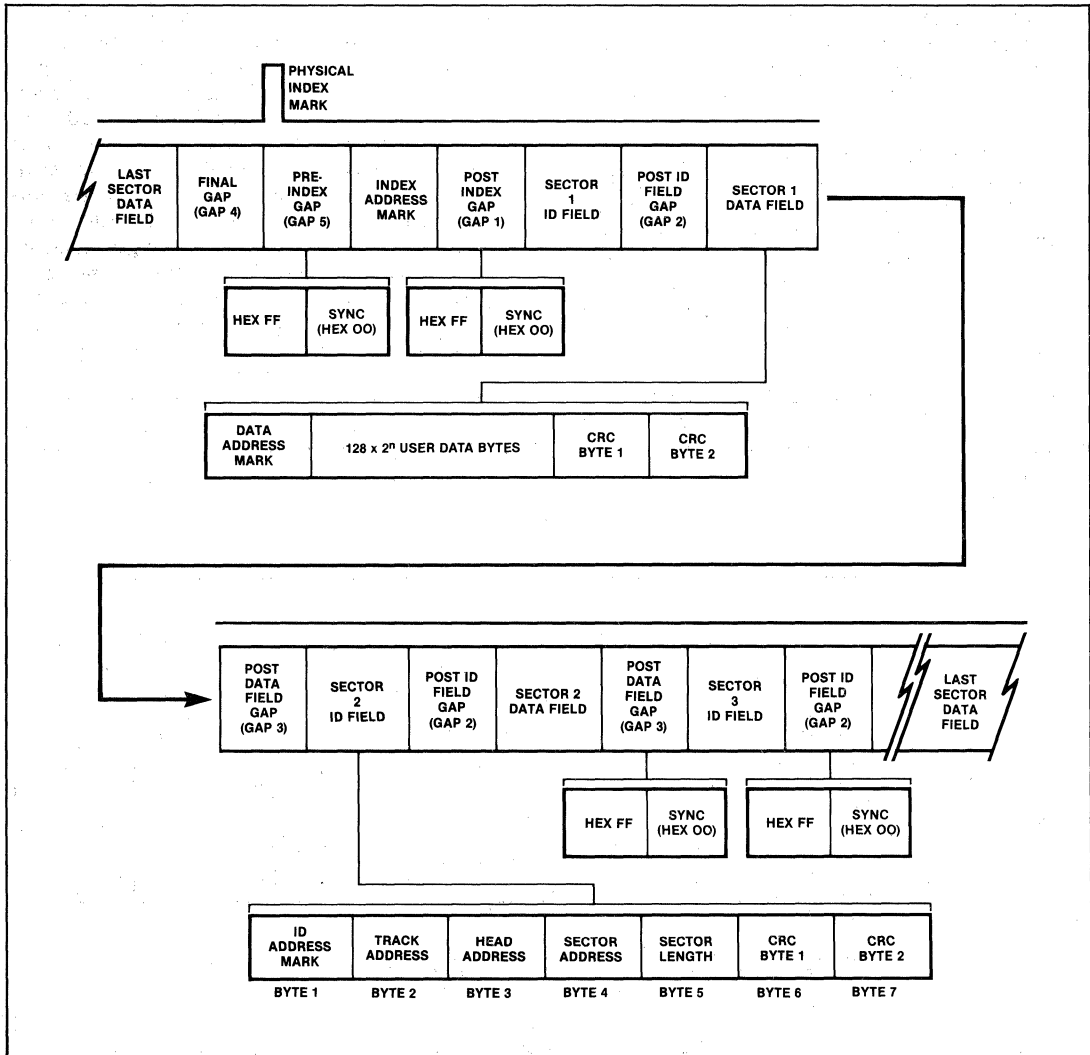


Figure 4. Standard Floppy Diskette Track Format (From SBC 204 Manual)

## Sector Interleaving

The initial formatting of a floppy disk determines where sectors are located within a track. It is not necessary to allocate sectors sequentially around the track (i.e., 1,2,3,...,26). In fact, it is often advantageous to place the sectors on the track in a non-sequential order. Sequential sector ordering optimizes sector access times during multi-sector transfers (e.g., when a program is loaded) by permitting the number of sectors specified (up to an entire track) to be transferred within a single revolution of the disk. A technique known as sector interleaving optimizes access times when, although sectors are accessed sequentially, a small amount of processing must be performed between sector reads/writes. For example, an editing program performing a text search reads sectors sequentially, and after each sector is read, performs a software search. If a match is not found, the software issues a read request for the next sector. Since the floppy disk continues to rotate during the time that the software executes, the next physical sector is already passing under the read/write head when the read request is issued, and the processor must wait for another complete revolution of the disk (approximately 166 milliseconds) before the data may actually be input. With interleaving, the sectors are not stored sequentially on a track; rather, each sector is physically removed from the previous sector by some number (known as the interleave factor) of physical sectors as shown in Figure 5. This method of sector allocation provides the processor additional execution time between sectors on the disk. For example, with a 26 sector/track format, an interleave factor of 2 provides 6.4 milliseconds of processing time between sequential 128 byte sector accesses.

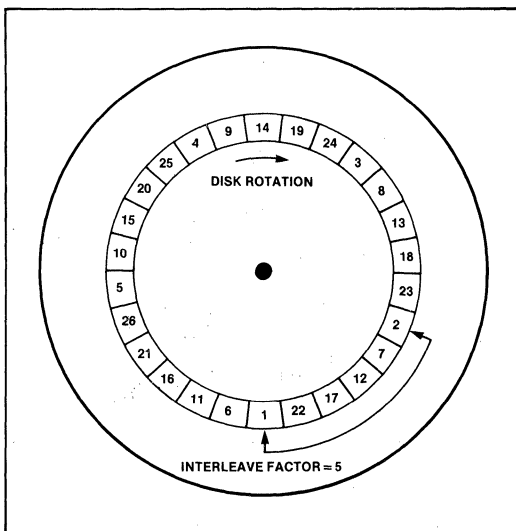


Figure 5. Interleaved Sector Allocation Within a Track

To calculate the correct interleave factor, the maximum processor time between sector operations must be divided by the time required for a complete sector to pass under the disk read/write head. After determining the interleave factor, the correct sector numbers are passed to the disk controller (in the exact order that they are to physically appear on the track) during the execution of a format operation.

## 4. THE 8272 FLEXIBLE DISKETTE CONTROLLER

The 8272 is a single-chip LSI Floppy Disk Controller (FDC) that contains the circuitry necessary to implement both single- and double-density floppy disk storage subsystems (with up to four dual-sided disk drives per FDC). The 8272 supports the IBM 3740 single-density recording format (FM) and the IBM System 34 double-density recording format (MFM). With the 8272, less than 30 ICs are needed to implement a complete disk subsystem. The 8272 accepts and executes high-level disk commands such as format track, seek, read sector, write sector, and read track. All data synchronization and error checking is automatically performed by the FDC to ensure reliable data storage and subsequent retrieval. External logic is required only for the generation of the FDC master clock and write clock (see Section 6) and for data separation (Section 5). The FDC provides signals that control the startup and base frequency selection of the data separator. These signals greatly ease the design of a phase-locked loop data separator.

In addition to the data separator interface signals, the 8272 also provides the necessary signals to interface to microprocessor systems with or without Direct Memory Access (DMA) capabilities. In order to interface to a large number of commercially available floppy disk drives, the FDC permits software specification of the track stepping rate, the head load time, and the head unload time.

The pin configuration and internal block diagram of the 8272 is shown in Figure 6. Table 2 contains a description for each FDC interface pin.

### Floppy Disk Commands

The 8272 executes fifteen high-level disk interface commands:

- |                        |                    |
|------------------------|--------------------|
| Specify                | Write Data         |
| Sense Drive Status     | Write Deleted Data |
| Sense Interrupt Status | Read Track         |
| Seek                   | Read ID            |
| Recalibrate            | Scan Equal         |
| Format Track           | Scan High or Equal |
| Read Data              | Scan Low or Equal  |
| Read Deleted Data      |                    |

# APPLICATIONS

Each command is initiated by a multi-byte transfer from the processor to the FDC (the transferred bytes contain command and parameter information). After complete command specification, the FDC automatically executes the command. The command result data (after execution of the command) may require a multi-byte transfer of status information back to the processor. It is convenient to consider each FDC command as consisting of the following three phases:

**COMMAND PHASE:** The executing program transfers to the FDC all the information required to perform a particular disk operation. The 8272 automatically enters the command phase after RESET and following the completion of the result phase (if any) of a previous command.

**EXECUTION PHASE:** The FDC performs the operation as instructed. The execution phase is entered immediately after the last command parameter is written to the FDC in the preceding command phase. The execution phase normally ends when the last data byte is transferred to/from the disk (signalled by the TC input to the FDC) or when an error occurs.

**RESULT PHASE:** After completion of the disk operation, status and other housekeeping information are made available to the processor. After the processor reads this information, the FDC reenters the command phase and is ready to accept another command.

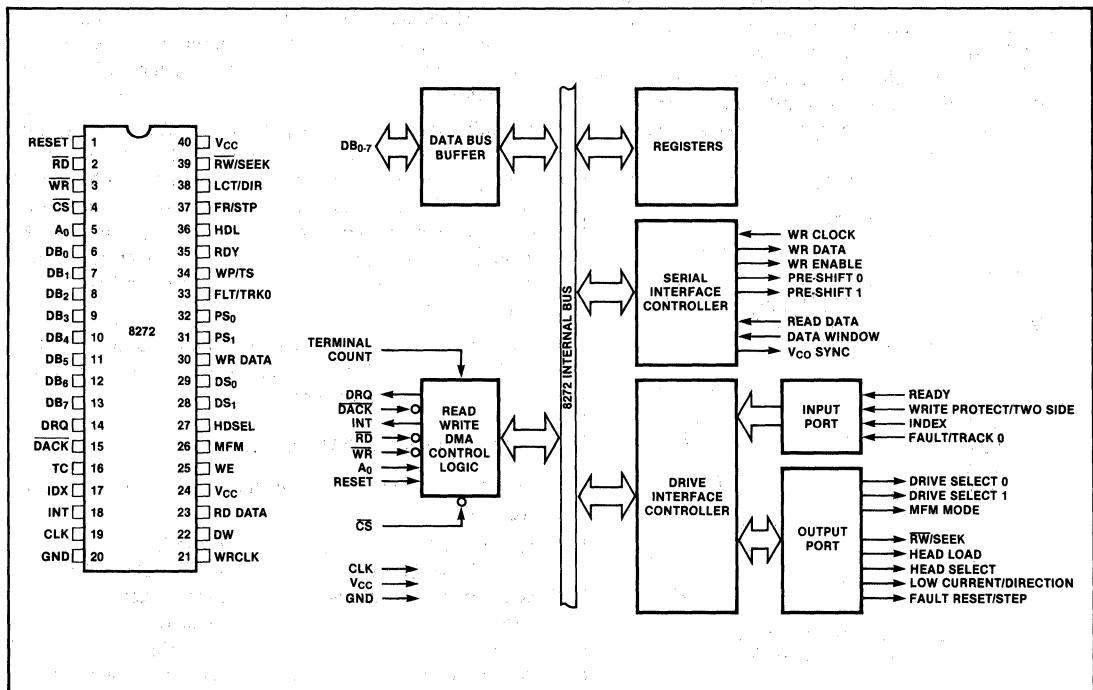


Figure 6. 8272 Pin Configuration and Internal Block Diagram

# APPLICATIONS

**Table 2. 8272 FDC Pin Description**

Number	Pin Symbol	I/O	To/From	Description
1	RST	I	uP	Reset. Active-high signal that places the FDC in the "idle" state and all disk drive output signals are forced inactive (low). This input must be held active during power on reset while the RD and WR inputs are active.
2	$\overline{RD}$	I*	uP	Read. Active-low control signal that enables data transfer from the FDC to the data bus.
3	$\overline{WR}$	I*	uP	Write. Active-low control signal that enables data transfer from the data bus into the FDC.
4	$\overline{CS}$	I	uP	Chip Select. Active-low control signal that selects the FDC. No reading or writing will occur unless the FDC is selected.
5	A <sub>0</sub>	I*	uP	Address. Selects the Data Register or Main Status Register for input/output in conjunction with the RD and WR inputs. (See Table 3.)
6-13	DB <sub>0</sub> -DB <sub>7</sub>	I/O*	uP	Data Bus. Bidirectional three-state 8-bit data bus.
14	DRQ	O	DMA	DMA Request. Active-high output that indicates an FDC request for DMA services.
15	$\overline{DACK}$	I	DMA	DMA Acknowledge. Active-low control signal indicating that the requested DMA transfer is in progress.
16	TC	I	DMA	Terminal Count. Active-high signal that causes the termination of a command. Normally, the terminal count input is directly connected to the TC/EOP output from the DMA controller, signalling that the DMA transfer has been completed. In a non-DMA environment, the processor must count data transfers and supply a TC signal to the FDC.
17	IDX	I	Drive	Index. Indicates detection of the physical index mark (the beginning of a track) on the selected disk drive.
18	INT	O	uP	Interrupt Request. Active-high signal indicating an 8272 interrupt service request.
19	CLK	I		Clock. Signal phase 8 MHz clock (50% duty cycle).
20	GND			Ground. DC power return.
21	WR CLK	I		Write Clock. 500 kHz (FM) or 1 MHz (MFM) write clock with a constant pulse width of 250 ns (for both FM and MFM recording). The write clock must be present at all times.
22	DW	I	PLL	Data Window. Data sample signal from the phase-locked loop indicating that the FDC should sample input data from the disk drive.
23	RD DATA	I	Drive	Read Data. FDC input data from the selected disk drive.
24	VCO	O	PLL	VCO Sync. Active-high output that enables the phase-locked loop to synchronize with the input data from the disk drive.
25	WE	O	Drive	Write Enable. Active-high output that enables the disk drive write gate.
26	MFM	O	PLL	MFM Mode. Active-high output used by external logic to enable the MFM double-density recording mode. When the MFM output is low, single-density FM recording is indicated.
27	HDSEL	O	Drive	Head Select. Selects head 0 or head 1 on a dual-sided disk.
28,29	DS <sub>1</sub> ,DS <sub>0</sub>	O	Drive	Drive Select. Selects one of four disk drives.
30	WR DATA	O	Drive	Write Data. Serial data stream (combination of clock and data bits) to be written on the disk.
31,32	PS <sub>1</sub> ,PS <sub>0</sub>	O	Drive	Precompensation (pre-shift) Control. Write precompensation output control during MFM mode. Specifies early, late, and normal timing signals. See the discussion in Section 5.



**Table 2. 8272 FDC Pin Description (continued)**

Number	Pin Symbol	I/O	To/From	Description
33	FLT/TRKO	I	Drive	Fault/Track 0. Senses the disk drive fault condition in the Read/Write mode and the Track 0 condition in the Seek mode.
34	WP/TS	I	Drive	Write Protect/Two-Sided. Senses the disk write protect status in the Read/Write mode and the dual-sided media status in the Seek mode.
35	RDY	I	Drive	Ready. Senses the disk drive ready status.
36	HDL	O	Drive	Head Load. Loads the disk drive read/write head. (The head is placed in contact with the disk.)
37	FR/STP	O	Drive	Fault Reset/Step. Resets the fault flip-flop in the disk drive when operating in the Read/Write mode. Provides head step pulses (to move the head from one cylinder to another cylinder) in the Seek mode.
38	LCT/DIR	O	Drive	Low Current/Direction. Signals that the recording head has been positioned over the inner cylinders (44-77) of the floppy disk in the Read/Write mode. (The write current must be lowered when recording on the physically shorter inner cylinders of the disk. Most drives do not track the actual head position and require that the FDC supply this signal.) Determines the head step direction in the Seek mode. In the Seek mode, a high level on this pin steps the read/write head toward the spindle (step-in); a low level steps the head away from the spindle (step-out).
39	RW/SEEK	O	Drive	Read, Write/Seek Mode Selector. A high level selects the Seek mode; a low level selects the Read/Write mode.
40	V <sub>CC</sub>			+ 5V DC Power.

\*Disabled when CS is high.

## Interface Registers

To support information transfer between the FDC and the system processor, the 8272 contains two 8-bit registers: the Main Status Register and the Data Register. The Main Status Register (read only) contains FDC status information and may be accessed at any time. The Main Status Register (Table 4) provides the system processor with the status of each disk drive, the status of the FDC, and the status of the processor interface. The Data Register (read/write) stores data, commands, parameters, and disk drive status information. The Data Register is used to program the FDC during the command phase and to obtain result information after completion of FDC operations. Data is read from, or written to, the FDC registers by the combination of the A<sub>0</sub>, RD, WR, and CS signals, as described in Table 3.

In addition to the Main Status Register, the FDC contains four additional status registers (ST0, ST1, ST2, and ST3). These registers are only available during the result phase of a command.

**Table 3. FDC Read/Write Interface**

CS	A <sub>0</sub>	RD	WR	Function
0	0	0	1	Read Main Status Register
0	0	1	0	Illegal
0	0	0	0	Illegal
0	1	0	0	Illegal
0	1	0	1	Read from Data Register
0	1	1	0	Write into Data Register
1	X	X	X	Data Bus is three-stated

# APPLICATIONS

**Table 4. Main Status Register Bit Definitions**

Bit Number	Symbol	Description
0	D <sub>0</sub> B	Disk Drive 0 Busy. Disk Drive 0 is in the Seek mode.
1	D <sub>1</sub> B	Disk Drive 1 Busy. Disk Drive 1 is in the Seek mode.
2	D <sub>2</sub> B	Disk Drive 2 Busy. Disk Drive 2 is in the Seek mode.
3	D <sub>3</sub> B	Disk Drive 3 Busy. Disk Drive 3 is in the Seek mode.
4	CB	FDC Busy. A read or write command is in process.
5	NDM	Non-DMA Mode. The FDC is in the non-DMA mode when this bit is high. This bit is set only during the execution phase of commands in the non-DMA mode. Transition to a low level indicates that the execution phase has ended.
6	DIO	Data Input/Output. Indicates the direction of a data transfer between the FDC and the Data Register. When DIO is high, data is read from the Data Register by the processor; when DIO is low, data is written from the processor to the Data Register.
7	RQM	Request for Master. Indicates that the Data Register is ready to send data to, or receive data from, the processor.

## Command/Result Phases

Table 5 lists the 8272 command set. For each of the fifteen commands, command and result phase data transfers are listed. A list of abbreviations used in the table is given in Table 6, and the contents of the result status registers (ST0-ST3) are illustrated in Table 7.

The bytes of data which are sent to the 8272 during the command phase, and are read out of the 8272 in the result phase, must occur in the order shown in Table 5. That is, the command code must be sent first and the other bytes sent in the prescribed sequence. All bytes of the command and result phases must be read/written as described. After the last byte of data in the command phase is sent to the 8272 the execution phase automatically starts. In a similar fashion, when the last byte of data is read from the 8272 in the result phase,

the command is automatically ended and the 8272 is ready for a new command. A command may be aborted by simply raising the terminal count signal (pin 16). This is a convenient means of ensuring that the processor may always gain control of the 8272 (even if the disk system hangs up in an abnormal manner).

It is important to note that during the result phase all bytes shown in Table 5 must be read. The Read Data command, for example, has seven bytes of data in the result phase. All seven bytes must be read in order to successfully complete the Read Data command. The 8272 will not accept a new command until all seven bytes have been read. The number of command and result bytes varies from command-to-command.

In order to read data from, or write data to, the Data Register during the command and result phases, the system processor must examine the Main Status Register to determine if the Data Register is available. The DIO (bit 6) and RQM (bit 7) flags in the Main Status Register must be low and high, respectively, before each byte of the command word may be written into the 8272. Many of the commands require multiple bytes, and as a result, the Main Status Register must be read prior to each byte transfer to the 8272. To read status bytes during the result phase, DIO and RQM in the Main Status Register must both be high. Note, checking the Main Status Register in this manner before each byte transfer to/from the 8272 is required only in the command and result phases, and is NOT required during the execution phase.

## Execution Phase

All data transfers to (or from) the floppy drive occur during the execution phase. The 8272 has two primary modes of operation for data transfers (selected by the specify command):

1. DMA mode
2. non-DMA mode

In the DMA mode, DRQ (DMA Request) is activated for each transfer request. The DMA controller responds to DRQ with  $\overline{DACK}$  (DMA Acknowledge) and  $\overline{RD}$  (for read commands) or  $\overline{WR}$  (for write commands). DRQ is reset by the FDC during the transfer. INT is activated after the last data transfer, indicating the completion of the execution phase, and the beginning of the result phase. In the DMA mode, the terminal count (TC/EOP) output of the DMA controller should be connected to the 8272 TC input to properly terminate disk data transfer commands.

# APPLICATIONS

**Table 5. 8272 Command Set**

PHASE	R/W	DATA BUS								REMARKS	PHASE	R/W	DATA BUS								REMARKS		
		D <sub>7</sub>	D <sub>6</sub>	D <sub>5</sub>	D <sub>4</sub>	D <sub>3</sub>	D <sub>2</sub>	D <sub>1</sub>	D <sub>0</sub>				D <sub>7</sub>	D <sub>6</sub>	D <sub>5</sub>	D <sub>4</sub>	D <sub>3</sub>	D <sub>2</sub>	D <sub>1</sub>	D <sub>0</sub>			
<b>READ DATA</b>																							
Command	W	MT	MFM	SK	0	0	1	1	0	Command Codes	Command	W	0	MFM	SK	0	0	1	0	Command Codes			
	W	0	0	0	0	0	HDS	DS1	DS0	Sector ID information prior to Command execution		W	0	0	0	0	0	HDS	DS1	DS0	Sector ID information prior to Command execution		
	W				C							W				C						Sector ID information prior to Command execution	
	W				H							W				H							Sector ID information prior to Command execution
	W				R							W				R							Sector ID information prior to Command execution
	W				N							W				N							Sector ID information prior to Command execution
	W				EOT							W				EOT							Sector ID information prior to Command execution
Execution	W			GPL						Data transfer between the FDD and the main-system	W			GPL							Data transfer between the FDD and the main-system. FDC reads the complete track contents from the physical index mark to EOT		
	W			DTL							W			DTL									
	R										R												
Result	R				ST 0					Status information after Command execution	R				ST 0						Status information after Command execution		
	R				ST 1						R				ST 1							Status information after Command execution	
	R				ST 2						R				ST 2							Status information after Command execution	
	R				C						R				C							Sector ID information after Command execution	
	R				H						R				H							Sector ID information after Command execution	
	R				R						R				R							Sector ID information after Command execution	
	R				N						R				N							Sector ID information after Command execution	
<b>READ DELETED DATA</b>																							
Command	W	MT	MFM	SK	0	1	1	0	0	Command Codes	Command	W	0	MFM	0	0	1	0	0	Command Codes			
	W	0	0	0	0	0	HDS	DS1	DS0	Sector ID information prior to Command execution		W	0	0	0	0	0	HDS	DS1	DS0	Sector ID information prior to Command execution		
	W				C							W				C						Sector ID information prior to Command execution	
	W				H							W				H							Sector ID information prior to Command execution
	W				R							W				R							Sector ID information prior to Command execution
	W				N							W				N							Sector ID information prior to Command execution
	W				EC 1								W				EC 1						
Execution	W			GPL						Data transfer between the FDD and the main-system	W				GPL							Data transfer between the FDD and the main-system	
	W			DTL							W				DTL								
	R										R												
Result	R				ST 0					Status information after Command execution	R				ST 0							Status information after Command execution	
	R				ST 1						R				ST 1							Status information after Command execution	
	R				ST 2						R				ST 2							Status information after Command execution	
	R				C						R				C							Sector ID information after Command execution	
	R				H						R				H							Sector ID information after Command execution	
	R				R						R				R							Sector ID information after Command execution	
	R				N						R				N							Sector ID information after Command execution	
<b>READ ID</b>																							
Command	W	0	MFM	0	0	1	0	1	0	Command Codes	Command	W	0	MFM	0	0	1	0	1	0	Command Codes		
	W	0	0	0	0	0	HDS	DS1	DS0	The first correct ID information on the track is stored in Data Register		W	0	0	0	0	0	HDS	DS1	DS0	The first correct ID information on the track is stored in Data Register		
Execution	R									Status information after Command execution	R											Status information after Command execution	
	R										R											Status information after Command execution	
	R										R											Status information after Command execution	
	R										R											Sector ID information during Execution Phase	
	R										R											Sector ID information during Execution Phase	
	R										R											Sector ID information during Execution Phase	
	R										R											Sector ID information during Execution Phase	
<b>WRITE DATA</b>																							
Command	W	MT	MFM	0	0	1	0	1	0	Command Codes	Command	W	0	MFM	0	0	1	1	0	1	Command Codes		
	W	0	0	0	0	0	HDS	DS1	DS0	Sector ID information prior to Command execution		W	0	0	0	0	0	HDS	DS1	DS0	Sector ID information prior to Command execution		
	W				C							W				N						Bytes/Sector	
	W				H							W				SC						Sectors/Track	
	W				R							W				GPL						Gap 3	
	W				N							W				D						Filter Byte	
	W				EOT								W										FDC formats an entire track
Execution	W			GPL						Data transfer between the main-system and the FDD	W											FDC formats an entire track	
	W			DTL							W												
	R										R												
Result	R				ST 0					Status information after Command execution	R				ST 0							Status information after Command execution	
	R				ST 1						R				ST 1							Status information after Command execution	
	R				ST 2						R				ST 2							Status information after Command execution	
	R				C						R				C							Sector ID information after Command execution	
	R				H						R				H							In this case, the ID information has no meaning	
	R				R						R				R							In this case, the ID information has no meaning	
	R				N						R				N							In this case, the ID information has no meaning	
<b>FORMAT A TRACK</b>																							
Command	W	0	MFM	0	0	1	1	0	1	Command Codes	Command	W	0	MFM	0	0	1	1	0	1	Command Codes		
	W	0	0	0	0	0	HDS	DS1	DS0	Bytes/Sector		W	0	0	0	0	0	HDS	DS1	DS0	Bytes/Sector		
Execution	W										W										Sectors/Track		
	W										W										Gap 3		
	W										W										Filter Byte		
	W										W											FDC formats an entire track	
	W										W												
	W										W												
	W										W												
Result	R				ST 0					Status information after Command execution	R				ST 0							Status information after Command execution	
	R				ST 1						R				ST 1							Status information after Command execution	
	R				ST 2						R				ST 2							Status information after Command execution	
	R				C						R				C							Sector ID information after Command execution	
	R				H						R				H							In this case, the ID information has no meaning	
	R				R						R				R							In this case, the ID information has no meaning	
	R				N						R				N							In this case, the ID information has no meaning	
<b>SCAN EQUAL</b>																							
Command	W	MT	MFM	SK	1	0	0	0	1	Command Codes	Command	W	0	MFM	0	0	0	0	1	Command Codes			
	W	0	0	0	0	0	HDS	DS1	DS0	Sector ID information prior to Command execution		W	0	0	0	0	0	HDS	DS1	DS0	Sector ID information prior to Command execution		
Execution	W				C					Sector ID information prior to Command execution	W				C						Sector ID information prior to Command execution		
	W				H						W				H							Sector ID information prior to Command execution	
	W				R						W				R							Sector ID information prior to Command execution	
	W				N						W				N							Sector ID information prior to Command execution	
	W				EOT							W				EOT							Data transfer between the FDD and the main-system
	W				GPL							W				GPL							Data transfer between the FDD and the main-system
	W				DTL							W				DTL							
Result	R				ST 0					Status information after Command execution	R				ST 0							Status information after Command execution	
	R				ST 1		</																

# APPLICATIONS

**Table 5. Command Set (Continued)**

PHASE	R/W	DATA BUS								REMARKS	PHASE	R/W	DATA BUS								REMARKS					
		D7	D6	D5	D4	D3	D2	D1	D0				D7	D6	D5	D4	D3	D2	D1	D0						
<b>SCAN LOW OR EQUAL</b>																										
Command	W	MT	MFM	SK	1	1	0	0	1	Command Codes	Command	W	0	0	0	0	0	1	1	1	Command Codes					
	W	0	0	0	0	0	HDS	DS1	DS0	Execution		W	0	0	0	0	0	0	DS1	DS0	Head retracted to Track 0					
Execution	W	C _____								Sector ID information prior Command execution	<b>SENSE INTERRUPT STATUS</b>															
	W	H _____									Command	W	0	0	0	0	1	0	0	0	Command Codes					
	W	R _____										Result	R	ST 0 _____								Status information at the end of each seek operation about the FDC				
	W	N _____									Command		R	C _____												
	W	EOT _____										Data compared between the FDD and the main-system	<b>SPECIFY</b>													
	W	GPL _____									Result		W	0	0	0	0	0	0	1	1	Command Codes				
W	STP _____								R	SPT _____ HUT _____								Timer Settings								
<b>SCAN HIGH OR EQUAL</b>																										
Command	W	MT	MFM	SK	1	1	1	0	1	Command Codes	Command	W	0	0	0	0	0	1	0	0	Command Codes					
	W	0	0	0	0	0	HDS	DS1	DS0	Execution		W	0	0	0	0	0	HDS	DS1	DS0	Status information about the FDD					
Execution	W	C _____								Sector ID information prior Command execution	<b>SEEK</b>															
	W	H _____									Command	W	0	0	0	0	1	1	1	1	Command Codes					
	W	R _____										Execution	W	0	0	0	0	0	HDS	DS1		DS0	Head is positioned over proper Cylinder on Diskette			
	W	N _____									Command		W	C _____												
	W	EOT _____										Data compared between the FDD and the main-system	<b>INVALID</b>													
	W	GPL _____									Result		W	Invalid Codes _____								Invalid Command Codes (NoOp - FDC goes into Standby State) ST 0 = 80 (16)				
W	STP _____								R	ST 0 _____																

**Table 6. Command/Result Parameter Abbreviations**

Symbol	Description	Symbol	Description
C	Cylinder Address. The currently selected cylinder address (0 to 76) on the disk.	EOT	End of Track. The final sector number of the current track.
D	Data Pattern. The pattern to be written in each sector data field during formatting.	GPL	Gap Length. The gap 3 size. (Gap 3 is the space between sectors excluding the VCO synchronization field as defined in section 3.)
DS0,DS1	Disk Drive Select. DS1 DS0 0 0 Drive 0 0 1 Drive 1 1 0 Drive 2 1 1 Drive 3	H	Head Address. Selected head: 0 or 1 (disk side 0 or 1, respectively) as encoded in the sector ID field.
DTL	Special Sector Size. During the execution of disk read/write commands, this parameter is used to temporarily alter the effective disk sector size. By setting N to zero, DTL may be used to specify a sector size from 1 to 256 bytes in length. If the actual sector (on the diskette) is larger than DTL specifies, the remainder of the actual sector is not passed to the system during read commands; during write commands, the remainder of the actual sector is written with all-zeroes bytes. DTL should be set to FF hexadecimal when N is not zero.	HLT	Head Load Time. Defines the time interval that the FDC waits after loading the head before initiating a read or write operation. Programmable from 2 to 254 milliseconds (in increments of 2 ms).
		HUT	Head Unload Time. Defines the time interval from the end of the execution phase (of a read or write command) until the head is unloaded. Programmable from 16 to 240 milliseconds (in increments of 16 ms).
		MFM	MFM/FM Mode Selector. Selects MFM double-density recording mode when high, FM single-density mode when low.

# APPLICATIONS

**Table 6. Command/Result Parameter Abbreviations (continued)**

Symbol	Description	Symbol	Description
MT	Multi-Track Selector. When set, this flag selects the multi-track operating mode. In this mode (used only with dual-sided disks), the FDC treats a complete cylinder (under both read/write head 0 and read/write head 1) as a single track. The FDC operates as if this expanded track started at the first sector under head 0 and ended at the last sector under head 1. With this flag set (high), a multi-sector read operation will automatically continue to the first sector under head 1 when the FDC finishes operating on the last sector under head 0.	SK	Skip Flag. When this flag is set, sectors containing deleted data address marks will automatically be skipped during the execution of multi-sector Read Data or Scan commands. In the same manner, a sector containing a data address mark will automatically be skipped during the execution of a multi-sector Read Deleted Data command.
N	Sector Size. The number of data bytes within a sector. (See Table 9.)	SRT	Step Rate Interval. Defines the time interval between step pulses issued by the FDC (track-to-track access time). Programmable from 1 to 16 milliseconds (in increments of 1 ms).
ND	Non-DMA Mode Flag. When set (high), this flag indicates that the FDC is to operate in the non-DMA mode. In this mode, the processor is interrupted for each data transfer. When low, the FDC interfaces to a DMA controller by means of the DRQ and DACK signals.	ST0 ST1 ST2 ST3	Status Register 0-3. Registers within the FDC that store status information after a command has been executed. This status information is available to the processor during the Result Phase after command execution. These registers may only be read after a command has been executed (in the exact order shown in Table 5 for each command). These registers should not be confused with the Main Status Register.
R	Sector Address. Specifies the sector number to be read or written. In multi-sector transfers, this parameter specifies the sector number of the first sector to be read or written.	STP	Scan Sector Increment. During Scan operations, this parameter is added to the current sector number in order to determine the next sector to be scanned.
SC	Number of Sectors per Track. Specifies the number of sectors per track to be initialized by the Format Track command.		

**Table 7. Status Register Definitions**

Bit Number	Symbol	Description
<b>Status Register 0</b>		
7,6	IC	Interrupt Code. 00 — Normal termination of command. The specified command was properly executed and completed without error. 01 — Abnormal termination of command. Command execution was started but could not be successfully completed. 10 — Invalid command. The requested command could not be executed. 11 — Abnormal termination. During command execution, the disk drive ready signal changed state.
5	SE	Seek End. This flag is set (high) when the FDC has completed the Seek command and the read/write head is positioned over the correct cylinder.
4	EC	Equipment Check Error. This flag is set (high) if a fault signal is received from the disk drive or if the track 0 signal fails to become active after 77 step pulses (Recalibrate command).
3	NR	Not Ready Error. This flag is set if a read or write command is issued and either the drive is not ready or the command specifies side 1 (head 1) of a single-sided disk.
2	H	Head Address. The head address at the time of the interrupt.
1,0	DS1,DS0	Drive Select. The number of the drive selected at the time of the interrupt.

# APPLICATIONS

**Table 7. Status Register Definitions (continued)**

Bit Number	Symbol	Description
<b>Status Register 1</b>		
7	EN	End of Track Error. This flag is set if the FDC attempts to access a sector beyond the final sector of the track.
6		Not used. This bit is always low.
5	DE	Data Error. Set when the FDC detects a CRC error in either the ID field or the data field of a sector.
4	OR	Overrun Error. Set (during data transfers) if the FDC does not receive DMA or processor service within the specified time interval.
3		Not used. This bit is always low.
2	ND	Sector Not Found Error. This flag is set by any of the following conditions: a) The FDC cannot locate the sector specified in the Read Data, Read Deleted Data, or Scan command. b) The FDC cannot locate the starting sector specified in the Read Track command. c) The FDC cannot read the ID field without error during a Read ID command.
1	NW	Write Protect Error. This flag is set if the FDC detects a write protect signal from the disk drive during the execution of a Write Data, Write Deleted Data, or Format Track command.
0	MA	Missing Address Mark Error. This flag is set by either of the following conditions: a) The FDC cannot detect the ID address mark on the specified track (after two occurrences of the physical index mark). b) The FDC cannot detect the data address mark or deleted data address mark on the specified track. (See also the MD bit of Status Register 2.)
<b>Status Register 2</b>		
7		Not used. This bit is always low.
6	CM	Control Mark. This flag is set when the FDC encounters one of the following conditions: a) A deleted data address mark during the execution of a Read Data or Scan command. b) A data address mark during the execution of a Read Deleted Data command.
5	DD	Data Error. Set (high) when the FDC detects a CRC error in a sector data field. This flag is not set when a CRC error is detected in the ID field.
4	WC	Cylinder Address Error. Set when the cylinder address from the disk sector ID field is different from the current cylinder address maintained within the FDC.
3	SH	Scan Hit. Set during the execution of the Scan command if the scan condition is satisfied.
2	SN	Scan Not Satisfied. Set during execution of the Scan command if the FDC cannot locate a sector on the specified cylinder that satisfies the scan condition.
1	BC	Bad Track Error. Set when the cylinder address from the disk sector ID field is FF hexadecimal and this cylinder address is different from the current cylinder address maintained within the FDC. This all "ones" cylinder number indicates a bad track (one containing hard errors) according to the IBM soft-sectored format specifications.
0	MD	Missing Data Address Mark Error. Set if the FDC cannot detect a data address mark or deleted data address mark on the specified track.

**Table 7. Status Register Definitions (continued)**

Bit Number	Symbol	Description
<b>Status Register 3</b>		
7	FT	Fault. This flag indicates the status of the fault signal from the selected disk drive.
6	WP	Write Protected. This flag indicates the status of the write protect signal from the selected disk drive.
5	RDY	Ready. This flag indicates the status of the ready signal from the selected disk drive.
4	TO	Track 0. This flag indicates the status of the track 0 signal from the selected disk drive.
3	TS	Two-Sided. This flag indicates the status of the two-sided signal from the selected disk drive.
2	H	Head Address. This flag indicates the status of the side select signal for the currently selected disk drive.
1,0	DS1,DS0	Drive Select. Indicates the currently selected disk drive number.

In the non-DMA mode, transfer requests are indicated by activation of both the INT output signal and the RQM flag (bit 7) in the Main Status Register. INT can be used for interrupt-driven systems and RQM can be used for polled systems. The system processor must respond to the transfer request by reading data from (activating RD), or writing data to (activating WR), the FDC. This response removes the transfer request (INT and RQM are set inactive). After completing the last transfer, the 8272 activates the INT output to indicate the beginning of the result phase. In the non-DMA mode, the processor must activate the TC signal to the FDC (normally by means of an I/O port) after the transfer request for the last data byte has been received (by the processor) and before the appropriate data byte has been read from (or written to) the FDC.

In either mode of operation (DMA or non-DMA), the execution phase ends when a terminal count signal is sensed or when the last sector on a track (the EOT parameter—Table 5) has been read or written. In addition, if the disk drive is in a “not ready” state at the beginning of the execution phase, the “not ready” flag (bit 3 in Status Register 0) is set (high) and the command is terminated.

If a fault signal is received from the disk drive at the end of a write operation (Write Data, Write Deleted Data, or Format), the FDC sets the “equipment check” flag (bit 4 in Status Register 0), and terminates the command after setting the interrupt code (bits 7 and 6 of Status Register 0) to “01” (bit 7 low, bit 6 high).

### Multi-sector and Multi-track Transfers

During disk read/write transfers (Read Data, Write Data, Read Deleted Data, and Write Deleted Data), the FDC will continue to transfer data from sequential sectors until the TC input is sensed. In the DMA mode, the

TC input is normally connected to the TC/EOP (terminal count) output of the DMA controller. In the non-DMA mode, the processor directly controls the FDC TC input as previously described. Once the TC input is received, the FDC stops requesting data transfers (from the system processor or DMA controller). The FDC, however, continues to read data from, or write data to, the floppy disk until the end of the current disk sector. During a disk read operation, the data read from the disk (after reception of the TC input) is discarded, but the data CRC is checked for errors; during a disk write operation, the remainder of the sector is filled with all-zero bytes.

If the TC signal is not received before the last byte of the current sector has been transferred to/from the system, the FDC increments the sector number by one and initiates a read or write command for this new disk sector.

The FDC is also designed to operate in a multi-track mode for dual-sided disks. In the multi-track mode (specified by means of the MT flag in the command byte—Table 5) the FDC will automatically increment the head address (from 0 to 1) when the last sector (on the track under head 0) has been read or written. Reading or writing is then continued on the first sector (sector 1) of head 1.

### Drive Status Polling

After the power-on reset, the 8272 automatically enters a drive status polling mode. If a change in drive status is detected (all drives are assumed to be “not ready” at power-on), an interrupt is generated. The 8272 continues this status polling between command executions (and between step pulses in the Seek command). In this manner, the 8272 automatically notifies the system processor when a floppy disk is inserted, removed, or changed by the operator.

## Command Details

During the command phase, the Main Status Register must be polled by the CPU before each byte is written into the Data Register. The DI0 (bit 6) and RQM (bit 7) flags in the Main Status Register must be low and high, respectively, before each byte of the command may be written into the 8272. The beginning of the execution phase for any of these commands will cause DI0 to be set high and RQM to be set low.

The following paragraphs describe the fifteen FDC commands in detail.

## Specify

The Specify command is used prior to performing any disk operations (including the formatting of a new disk) to define drive/FDC operating characteristics. The Specify command parameters set the values for three internal timers:

1. **Head Load Time (HLT)** — This seven-bit value defines the time interval that the FDC waits after loading the head before initiating a read or write operation. This timer is programmable from 2 to 254 milliseconds in increments of 2 ms.
2. **Head Unload Time (HUT)** — This four-bit value defines the time from the end of the execution phase (of a read or write command) until the head is unloaded. This timer is programmable from 16 to 240 milliseconds in increments of 16 ms. If the processor issues another command before the head unloads, the head will remain loaded and the head load wait will be eliminated.
3. **Step Rate Time (SRT)** — This four-bit value defines the time interval between step pulses issued by the FDC (track-to-track access time). This timer is programmable from 1 to 16 milliseconds in increments of 1 ms.

The time intervals mentioned above are a direct function of the FDC clock (CLK on pin 19). Times indicated above are for an 8 MHz clock.

The Specify command also indicates the choice of DMA or non-DMA operation (by means of the ND bit). When this bit is high the non-DMA mode is selected; when ND is low, the DMA mode is selected.

## Sense Drive Status

This command may be used by the processor whenever it wishes to obtain the status of the disk drives. Status Register 3 (returned during the result phase) contains the drive status information as described in Table 7.

## Sense Interrupt Status

An interrupt signal is generated by the FDC when one or more of the following events occurs:

1. The FDC enters the result phase for:
  - a. Read Data command
  - b. Read Track command
  - c. Read ID command
  - d. Read Deleted Data command
  - e. Write Data command
  - f. Format Track command
  - g. Write Deleted Data command
  - h. Scan commands
2. The ready signal from one of the disk drives changes state.
3. A Seek or Recalibrate command completes operation.
4. The FDC requires a data transfer during the execution phase of a command in the non-DMA mode.

Interrupts caused by reasons (1) and (4) above occur during normal command operations and are easily discernible by the processor. However, interrupts caused by reasons (2) and (3) above are uniquely identified with the aid of the Sense Interrupt Status command. This command, when issued, resets the interrupt signal and by means of bits 5, 6, and 7 of Status Register 0 (returned during the result phase) identifies the cause of the interrupt (see Table 8).

**Table 8. Interrupt Codes**

Seek End Bit 5	Interrupt Code		Cause
	Bit 6	Bit 7	
0	1	1	Ready Line changed state, either polarity
1	0	0	Normal Termination of Seek or Recalibrate Command
1	1	0	Abnormal Termination of Seek or Recalibrate Command

Neither the Seek nor the Recalibrate command has a result phase. Therefore, it is mandatory to use the Sense Interrupt Status Command after these commands to effectively terminate them and to provide verification of the disk head position.



When an interrupt is received by the processor, the FDC busy flag (bit 4) and the non-DMA flag (bit 5) may be used to distinguish the above interrupt causes:

bit 5	bit 4	
0	0	Asynchronous event-(2) or (3) above
0	1	Result phase-(1) above
1	1	Data transfer required-(4) above

A single interrupt request to the processor may, in fact, be caused by more than one of the above events. The processor should continue to issue Sense Interrupt Status commands (and service the resulting conditions) until an invalid command code is received. In this manner, all "hidden" interrupts are serviced.

## Seek

The Seek command causes the drive's read/write head to be positioned over the specified cylinder. The FDC determines the difference between the current cylinder address and the desired (specified) address, and issues the appropriate number of step pulses. If the desired cylinder address is larger than the current address, the direction signal (LCT/DIR, pin 38) is set high (step-in); the direction signal is set low (step-out) if the desired cylinder address is less than the current address. No head movement occurs (no step pulses are issued) if the desired cylinder is the same as the current cylinder.

The rate at which step pulses are issued is controlled by the step rate time (SRT) in the Specify command. After each step pulse is issued, the desired cylinder address is compared against the current cylinder address. When the cylinder addresses are equal, the "seek end" flag (bit 5 in Status Register 0) is set (high) and the command is terminated. If the disk drive becomes "not ready" during the seek operation, the "not ready" flag (in Status Register 0) is set (high) and the command is terminated.

During the command phase of the Seek operation the FDC is in the FDC busy state, but during the execution phase it is in the non-busy state. While the FDC is in the non-busy state, another Seek command may be issued. In this manner parallel seek operations may be in operation on up to four floppy disk drives at once. The Main Status Register contains a flag for each drive (Table 4) that indicates whether the associated drive is currently operating in the seek mode. When a drive has completed a seek operation, the FDC generates an interrupt. In response to this interrupt, the system software must issue a Sense Interrupt Status command. During the result phase of this command, Status Register 0 (containing the drive number in bits 0 and 1) is read by the processor.

## Recalibrate

This command causes the read/write head of the disk drive to retract to the track 0 position. The FDC clears the contents of its internal cylinder counter, and checks the status of the track 0 signal from the disk drive. As long as the track 0 signal is low, the direction signal remains high and step pulses are issued. When the track 0 signal goes high, the seek end flag (in Status Register 0) is set (high) and the command is terminated. If the track 0 signal is still low after 77 step pulses have been issued, the seek end and equipment check flags (in Status Register 0) are both set and the Recalibrate command is terminated.

Recalibrate commands for multiple drives can be overlapped in the same manner that Seek commands are overlapped.

## Format Track

The Format Track command formats or "initializes" a track on a floppy disk by writing the ID field, gaps, and address marks for each sector. Before issuing the Format command, the Seek command must be used to position the read/write head over the correct cylinder. In addition, a table of ID field values (cylinder, head, and sector addresses and sector length code) must be prepared before the command is executed. During command execution, the FDC accesses the table and, using the values supplied, writes each sector on the track. The ID field address mark originates from the FDC and is written automatically as the first byte of each sector's ID field. The cylinder, head, and sector addresses are taken, in order, from the table. The ID field CRC character (derived from the data written in the first five bytes) is written as the last two bytes of the ID field. Gaps are written automatically by the FDC, with the length of the variable gap determined by one of the Format command parameters.

The data field address mark is generated by the FDC and is written automatically as the first byte of the data field. The data pattern specified in the command phase is written into each data byte of each sector. A CRC character is derived from the data address mark and the data written in the sector's data field. The two CRC bytes are appended to the last data byte.

The formatting of a track begins at the physical index mark. As previously mentioned, the order of sector assignment is taken directly from the formatting table. Four entries are required for each sector: a cylinder address, a head address, a sector address, and a sector length code. The cylinder address in the ID field should be equal to the cylinder address of the track currently being formatted.

# APPLICATIONS

The sector addresses must be unique (no two equal). The order of the sector entries in the table is the sequence in which sector numbers appear on the track when it is formatted. The number of entry sets (cylinder, head, and sector address and sector length code) must equal the number of sectors allocated to the track (specified in the command phase).

Since the sector address is supplied, in order, for each sector, tracks can be formatted sequentially (the first sector following the index mark is assigned sector address 1, the adjacent sector is assigned sector address 2, and so on) or sector numbers can be interleaved (see section 3) on a track.

Table 9 lists recommended gap sizes and sectors/track for various sector sizes.

## Read Data

Nine (9) bytes are required to complete the command phase specification for the Read Data command. During the execution phase, the FDC loads the head (if it is in the unloaded state), waits the specified head load time (defined in the Specify command), and begins reading ID address marks and ID fields. When the requested sector address compares with the sector address read from the disk, the FDC outputs data (from the data field) byte-by-byte to the system. The Read Data command automatically operates in the multi-sector mode described earlier. In addition, multi-track operation may be specified by means of the MT command flag (Table 5). The amount of data that can be transferred with a single command to the FDC depends on the multi-track flag, the recording density flag, and the number of bytes per sector.

During the execution of read and write commands, the special sector size parameter (DTL) is used to temporarily alter the effective disk sector size. By setting the sector size code (N) to zero, DTL may be used to specify a sector size from 1 to 256 bytes in length. If the actual sector (on the disk) is larger than DTL specifies, only the number of bytes specified by the DTL parameter are

passed to the system; the remainder of the actual disk sector is not transferred (although the data is checked for CRC errors). Multi-sector read operations are performed in the same manner as they are when the sector size code is non-zero. (The N and DTL parameters are always present in the command sequence. DTL should be set to FF hexadecimal when N is not zero.)

If the FDC detects the physical index mark twice without finding the requested sector, the FDC sets the "sector not found error" flag (bit 2 in Status Register 1) and terminates the Read Data command. The interrupt code (bits 7 and 6 of Status Register 0) is set to "01." Note that the FDC searches for each sector in a multi-sector operation. Therefore, a "sector not found" error may occur after successful transfer of one or more preceding sectors. This error could occur if a particular sector number was not included when the track was first formatted or if a hard error on the disk has invalidated a sector ID field.

After reading the ID field and data field in each sector, the FDC checks the CRC bytes. If a read error is detected (incorrect CRC in the ID field), the FDC sets the "data error" flag in Status Register 1; if a CRC error occurs in the data field, the FDC sets the "data error" flag in Status Register 2. In either error condition, the FDC terminates the Read Data command. The interrupt code (bits 7 and 6 in Status Register 0) is set to "01."

If the FDC reads a deleted data address mark from the disk, and the skip flag (specified during the command phase) is not set, the FDC sets the "control mark" flag (bit 6 in Status Register 2) and terminates the Read Data command (after reading all the data in the sector). If the skip flag is set, the FDC skips the sector with the deleted data address mark and reads the next sector. Thus, the skip flag may be used to cause the FDC to ignore deleted data sectors during a multi-sector read operation.

During disk data transfers between the FDC and the system, the FDC must be serviced by the system (processor or DMA controller) every 27  $\mu$ s in the FM mode, and every 13  $\mu$ s in the MFM mode. If the FDC is not

**Table 9. Sector Size Relationships**

Format	Sector Size	N Sector Size Code	SC Sectors/ Track	GPL <sup>1</sup> Gap 3 Length	GPL <sup>2</sup> Gap 3 Length	Remarks
FM Mode	128 bytes/Sector	00	1A <sub>(16)</sub>	07 <sub>(16)</sub>	1B <sub>(16)</sub>	IBM Diskette 1
	256	01	0F <sub>(16)</sub>	0E <sub>(16)</sub>	2A <sub>(16)</sub>	
	512	02	08	1B <sub>(16)</sub>	3A <sub>(16)</sub>	
MFM Mode	256	01	1A <sub>(16)</sub>	0E <sub>(16)</sub>	36 <sub>(16)</sub>	IBM Diskette 2D
	512	02	0F <sub>(16)</sub>	1B <sub>(16)</sub>	54 <sub>(16)</sub>	IBM Diskette 2D
	1024	03	08	35 <sub>(16)</sub>	74 <sub>(16)</sub>	

**Notes:** 1. Suggested values of GPL in Read or Write commands to avoid splice point between data field and ID field of contiguous sectors.  
2. Suggested values of GPL in Format command.

serviced within this interval, the "overrun error" flag (bit 4 in Status Register 1) is set and the Read Data command is terminated.

If the processor terminates a read (or write) operation in the FDC, the ID information in the result phase is dependent upon the state of the multi-track flag and end of track byte. Table 11 shows the values for C, H, R, and N, when the processor terminates the command.

## Write Data

Nine (9) bytes are required to complete the command phase specification for the Write Data command. During the execution phase the FDC loads the head (if it is in the unloaded state), waits the specified head load time (defined by the Specify command), and begins reading sector ID fields. When the requested sector address compares with the sector address read from the disk, the FDC reads data from the processor one byte at a time via the data bus and outputs the data to the data field of that sector. The CRC is computed on this data and two CRC bytes are written at the end of the data field.

The FDC reads the ID field of each sector and checks the CRC bytes. If the FDC detects a read error (incorrect CRC) in one of the ID fields, it sets the "data error" flag (bit 5 in Status Register 1) and terminates the Write Data command. The interrupt code (bits 7 and 6 in Status Register 0) is set to "01."

The Write Data command operates in much the same manner as the Read Data command. The following items are the same; refer to the Read Data command for details:

- Multi-sector and Multi-track operation
- Data transfer capacity
- "End of track error" flag
- "Sector not found error" flag
- "Data error" flag
- Head unload time interval
- ID information when the processor terminates the command (see Table 11)
- Definition of DTL when  $N=0$  and when  $N \neq 0$

During the Write Data execution phase, data transfers between the processor and FDC must occur every 31  $\mu\text{s}$  in the FM mode, and every 15  $\mu\text{s}$  in the MFM mode. If the time interval between data transfers is longer than this, the FDC sets the "overrun error" flag (bit 4 in Status Register 1) and terminates the Write Data command.

## Read Deleted Data

This command operates in almost the same manner as the Read Data command operates. The only difference involves the treatment of the data address mark and the

skip flag. When the FDC detects a data address mark at the beginning of a data field (and the skip flag is not set), the FDC reads all the data in the sector, sets the "control mark" flag (bit 6 in Status Register 2), and terminates the command. If the skip flag is set, the FDC skips the sector with the data address mark and continues reading at the next sector. Thus, the skip flag may be used to cause the FDC to read only deleted data sectors during a multi-sector read operation.

## Write Deleted Data

This command operates in the same manner as the Write Data command operates except that a deleted data address mark is written at the beginning of the data field instead of the normal data address mark. This command is used to mark a bad sector (containing a hard error) on the floppy disk.

## Read Track

The Read Track command is similar to the Read Data command except that the entire data field is read continuously from each of the sectors of a track. Immediately after encountering the physical index mark, the FDC starts reading all data fields on the track as continuous blocks of data. If the FDC finds an error in the ID field or data field CRC check bytes, it continues to read data from the track. The FDC compares the ID information read from each sector with the values specified during the command phase. If the specified ID field information is not found on the track, the "sector not found error" flag (in Status Register 1) is set. Multi-track and skip operations are not allowed with this command.

This command terminates when the last sector on the track has been read. (The number of sectors on the track is specified by the end of track parameter byte during the command phase.) If the FDC does not find an ID address mark on the disk after it encounters the physical index mark for the second time, it sets the "missing address mark error" flag (bit 0 in Status Register 1) and terminates the command. The interrupt code (bits 7 and 6 of Status Register 0) is set to "01."

## Read ID

The Read ID command transfers (reads) the first correct ID field from the current disk track (following the physical index mark) to the processor. If no correct ID address mark is found on the track, the "missing address mark error" flag is set (bit 0 in Status Register 1). If no data mark is found on the track, the "sector not found error" flag is also set (bit 2 in Status Register 1). Either error condition causes the command to be terminated.

# APPLICATIONS

## Scan Commands

The Scan commands allow the data being read from the disk to be compared against data supplied by the system (by the processor in non-DMA mode, and by the DMA controller in DMA mode). The FDC compares the data on a byte-by-byte basis, and searches for a sector of data that meets the conditions of "disk data equal to system data", "disk data less than or equal to system data", or "disk data greater than or equal to system data". Simple binary (ones complement) arithmetic is used for comparison (FF = largest number, 00 = smallest number). If, after a complete sector of data is compared, the conditions are not met, the sector number is incremented by the scan sector increment (specified in the command phase), and the scan operation is continued. The scan operation continues until one of the following conditions occurs; the conditions for scan are met (equal, low, or high), the last sector on the track is reached, or the terminal count signal is received.

If the conditions for scan are met, the FDC sets the "scan hit" flag (bit 3 in Status Register 2) and terminates the Scan command. If the conditions for scan

are not met between the starting sector and the last sector on the track (specified in the command phase), the FDC sets the "scan not satisfied" flag (bit 2 in Status Register 2) and terminates the Scan command. The receipt of a terminal count signal from the processor or DMA controller during the scan operation will cause the FDC to complete the comparison of the particular byte which is in process, and to terminate the command. Table 10 shows the status of the "scan hit" and "scan

**Table 10. Scan Status Codes**

Command	Status Register 2		Comments
	Bit 2 = SN	Bit 3 = SH	
Scan Equal	0	1	$D_{FDD} = D_{Processor}$
	1	0	$D_{FDD} \neq D_{Processor}$
Scan Low or Equal	0	1	$D_{FDD} = D_{Processor}$
	0	0	$D_{FDD} < D_{Processor}$
	1	0	$D_{FDD} \not\leq D_{Processor}$
Scan High or Equal	0	1	$D_{FDD} = D_{Processor}$
	0	0	$D_{FDD} > D_{Processor}$
	1	0	$D_{FDD} \not\geq D_{Processor}$

**Table 11. ID Information When Processor Terminates Command**

MT	EOT	Final Sector Transferred to Processor	ID Information at Result Phase			
			C	H	R	N
0	1A 0F 08	Sector 1 to 25 at Side 0 Sector 1 to 14 at Side 0 Sector 1 to 7 at Side 0	NC	NC	R + 1	NC
	1A 0F 08	Sector 26 at Side 0 Sector 15 at Side 0 Sector 8 at Side 0	C + 1	NC	R = 01	NC
	1A 0F 08	Sector 1 to 25 at Side 1 Sector 1 to 14 at Side 1 Sector 1 to 7 at Side 1	NC	NC	R + 1	NC
	1A 0F 08	Sector 26 at Side 1 Sector 15 at Side 1 Sector 8 at Side 1	C + 1	NC	R = 01	NC
1	1A 0F 08	Sector 1 to 25 at Side 0 Sector 1 to 14 at Side 0 Sector 1 to 7 at Side 0	NC	NC	R + 1	NC
	1A 0F 08	Sector 26 at Side 0 Sector 15 at Side 0 Sector 8 at Side 0	NC	$\overline{\text{LSB}}$	R = 01	NC
	1A 0F 08	Sector 1 to 25 at Side 1 Sector 1 to 14 at Side 1 Sector 1 to 7 at Side 1	NC	NC	R + 1	NC
	1A 0F 08	Sector 26 at Side 1 Sector 15 at Side 1 Sector 8 at Side 1	C + 1	$\overline{\text{LSB}}$	R = 01	NC

- Notes: 1. NC (No Change): The same value as the one at the beginning of command execution.  
 2. LSB (Least Significant Bit): The least significant bit of H is complemented.

not satisfied" flags under various scan termination conditions.

If the FDC encounters a deleted data address mark in one of the sectors and the skip flag is low, it regards the sector as the last sector on the cylinder, sets the "control mark" flag (bit 6 in Status Register 2) and terminates the command. If the skip flag is high, the FDC skips the sector with the deleted address mark, and reads the next sector. In this case, the FDC also sets the "control mark" flag (bit 6 in Status Register 2) in order to show that a deleted sector had been encountered.

**NOTE:** During scan command execution, the last sector on the track must be read for the command to terminate properly. For example, if the scan sector increment is set to 2, the end of track parameter is set to 26, and the scan begins at sector 21, sectors 21, 23, and 25 will be scanned. The next sector, 27 will not be found on the track and an abnormal command termination will occur. The command would be completed in a normal manner if either a) the scan had started at sector 20 or b) the end of track parameter had been set to 25.

During the Scan command, data is supplied by the processor or DMA controller for comparison against the data read from the disk. In order to avoid having the "overrun error" flag set (bit 4 in Status Register 1), it is necessary to have the data available in less than 27  $\mu$ s (FM Mode) or 13  $\mu$ s (MFM Mode). If an overrun error occurs, the FDC terminates the command.

## Invalid Commands

If an invalid (undefined) command is sent to the FDC, the FDC will terminate the command. No interrupt is generated by the 8272 during this condition. Bit 6 and bit 7 (DIO and RQM) in the Main Status Register are both set indicating to the processor that the 8272 is in the result phase and the contents of Status Register 0 must be read. When the processor reads Status Register 0 it will find an 80H code indicating that an invalid command was received.

A Sense Interrupt Status command must be sent after a Seek or Recalibrate interrupt; otherwise the FDC will consider the next command to be an invalid command. Also, when the last "hidden" interrupt has been serviced, further Sense Interrupt Status commands will result in invalid command codes.

In some applications the user may wish to use this command as a No-Op command to place the FDC in a stand-by or no operation state.

## 5. THE DATA SEPARATOR

As briefly discussed in section 2, LSI disk controllers such as the 8272 require external circuitry to generate a data window signal. This signal is used within the FDC to isolate the data bits contained within the READ DATA input signal from the disk drive. (The disk READ DATA signal is a composite signal constructed from both clock and data information.) After isolating the data bits from this input signal, the FDC assembles the data bits into 8-bit bytes for transfer to the system processor or memory.

### Single Density

In single-density (FM) recording (Figure 3), the bit cell is 4 microseconds wide. Each bit cell contains a clock bit at the leading edge of the cell. The data bit (if present) is always located at the center of the cell. The job of data separation is relatively straightforward for single-density; simply generate a data window 2  $\mu$ s wide starting 1  $\mu$ s after each clock bit. Since every cell has a clock bit, a fixed window reference is available for every data bit and because the window is 2  $\mu$ s wide, a slightly shifted data bit will still remain within the data window.

A single-density data separator with these specifications may be easily generated using a digital or analog one-shot triggered by the clock bit.

### Double-Density

Double-density (MFM) bit cells are reduced to 2  $\mu$ s (in order to double the disk data storage capacity). Clock bits are inserted into the data stream only if data bits are not present in both the current and preceding bit cells (Figure 3). The data bit (if present) still occurs at the center of the bit cell and the clock bit (if present) still occurs at the leading edge of the bit cell.

MFM data separation has two problems. First, only some bit cells contain a clock bit. In this manner, MFM encoding loses the fixed bit cell reference pulse present in FM encoding. Second, the bit cell for MFM is one-half the size of the bit cell for FM. This shorter bit cell means that MFM cannot tolerate as large a playback data-shift (as FM can tolerate) without errors.

Since most playback data-shift is predictable, the FDC can precompensate the write data stream so that data/clock pulses will be correctly positioned for subsequent playback. This function is completely controlled by the FDC and is only required for MFM recording. During write operations, the FDC specifies an early, normal, or late bit positioning. This timing information is specified with respect to the FDC write clock. Early and late timing is typically 125 ns to 250 ns before or after the write clock transition (depending on disk drive requirements).

# APPLICATIONS

The data separator circuitry for double-density recording must continuously analyze the total READ DATA stream, synchronizing its operation (window generation) with the actual clock/data bits of the data stream. The data separation circuit must track the disk input data frequency very closely—unpredictable bit shifts leave less than 50 ns margin to the window edges.

## Phase-Locked Loop

Only an analog phase-locked loop (PLL) can provide the reliability required for a double-density data separation circuit. (A phase-locked loop is an electronic circuit that constantly analyzes the frequency of an input signal and locks another oscillator to that frequency.) Using analog PLL techniques, a data separator can be designed with  $\pm 1$  ns resolution (this would require a 100 MHz clock in a digital phase-locked loop). The analog PLL determines the clock and data bit positions by sampling each bit in the serial data stream. The phase relationship between a data bit and the PLL generated data window is constantly fed back to adjust the position of the data window, enabling the PLL to track input data frequency changes, and thereby reliably read previously recorded data from a floppy disk.

## PLL Design

A block diagram of the phase-locked loop described in this application note is shown in Figure 7. Basically, the phase-locked loop operates by comparing the frequency of the input data (from the disk drive) against the frequency of a local oscillator. The difference of these frequencies is used to increase or decrease the frequency of the local oscillator in order to bring its frequency closer to that of the input. The PLL synchronizes the local oscillator to the frequency of the input during the all “zeroes” synchronization field on the floppy disk (immediately preceding both the ID field and the data field).

The PLL consists of nine ICs and is located on page 3 of the schematics in the Appendix. The 8272 VCO output essentially turns the PLL circuitry on and off. When the PLL is off, it “idles” at its center frequency. The VCO output turns the PLL on only when valid data is being received from the disk drive. The VCO turns the PLL on after the read/write head has been loaded and the head load time has elapsed. The PLL is turned off in the gap between the ID field and the data field and in the gap after the data field (before the next sector ID field). The GPL parameter in the FDC read and write commands specifies the elapsed time (number of data bytes) that the PLL is turned off in order to blank out discontinuities that appear in the gaps when the write current is turned on and off. The PLL operates with either MFM or FM input data. The MFM output from the FDC controls the PLL operation frequency.

The PLL consists of six functional blocks as follows:

1. Pulse Shaping — A 96LS02 senses a READ DATA pulse and provides a clean output signal to the FDC and to the PLL Phase Comparator and Frequency Discriminator circuitry.
2. Phase Comparator — The phase difference between the PLL oscillator and the READ DATA input is compared. Pump up (PU) and pump down (PD) error signals are derived from this phase difference and output to the filter. If there is no phase difference between the PLL oscillator and the READ DATA input, the PU and PD pulse widths are equal. If the READ DATA pulse occurs early, the PU duration is shorter than the PD duration. If the data pulse occurs late, the PU duration is longer than the PD duration.
3. Filter — This analog circuit filters the PU and PD pulses into an error voltage. This error voltage is buffered by an LM358 operational amplifier.

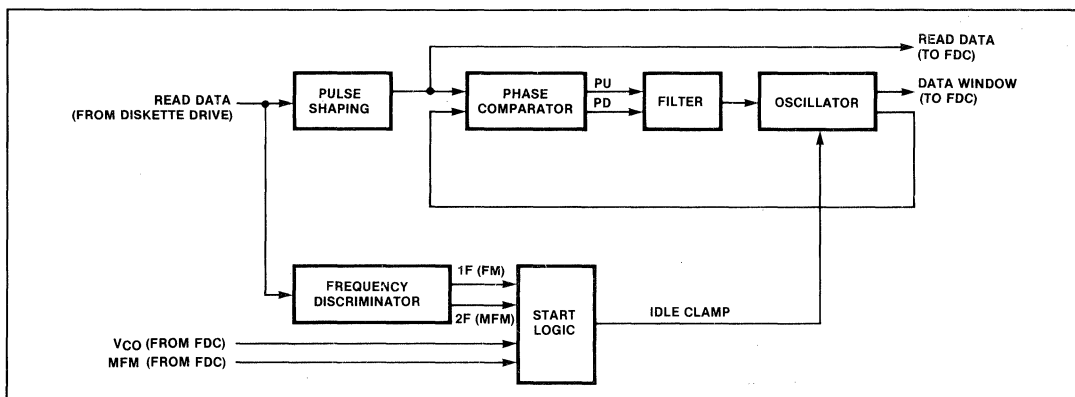


Figure 7. Phase-Locked Loop Data Separator

# APPLICATIONS

4. PLL Oscillator — This oscillator is composed of a 74LS393, 74LS74, and 96LS02. The oscillator frequency is controlled by the error voltage output by the filter. This oscillator also generates the data window signal to the FDC.
5. Frequency Discriminator — This logic tracks the READ DATA input from the disk drive and discriminates between the synchronization gap for FM recording (250 KHz) and the gap for MFM recording (500 KHz). Synchronization gaps immediately precede address marks.
6. Start Logic — The function of this logic is to clamp the PLL oscillator to its center frequency (2 MHz) until the FDC VCO signal is enabled and a valid data pattern is sensed by the frequency discriminator. The start logic (consisting of a 74LS393 and 74LS74) ensures that the PLL oscillator is started with zero phase error.

## PLL Adjustments

The PLL must be initially adjusted to operate at its center frequency with the VCO output off and the adjustment jumper removed. The 5K trimpot should be adjusted until the frequency at the test point (Q output of the 96LS02) is 2 MHz. The jumper should then be replaced for normal operation.

## PLL Design Details

The following paragraphs describe the operational and design details of the phase-locked loop data separator il-

lustrated in the appendix. Note that the analog section is operated from a separately filtered +5V supply.

## Initialization

As long as the 8272 maintains a low VCO signal, the data separator logic is "turned off". In this state, the PLL oscillator (96LS02) is not oscillating and therefore the 2XBR signal is constantly low. In addition, the pump up (PU) and pump down (PD) signals are inactive (PU low and PD high), the CNT8 signal is inactive (low), and the filter input voltage is held at 2.5 volts by two 1Mohm resistors between ground and +5 volts.

## Floppy Disk Data

The data separator frequency discriminator, the input pulse shaping circuitry, and the start logic are always enabled and respond to rising edges of the READ DATA signal. The rising edge of every data bit from the disk drive triggers two pulse shaping one-shots. The first pulse shaper generates a stable and well-defined 200 ns read data pulse for input to the 8272 and other portions of the data separator logic. The second one-shot generates a 2.5  $\mu$ s data pulse that is used for input data frequency discrimination.

The frequency discriminator operates as illustrated in Figure 8. The 2F output signal is active (high) during reception of valid MFM (double-density) sync fields on the disk while the 1F signal is active (high) during reception of valid FM (single-density) sync fields. A multiplexer (controlled by the 8272 MFM signal) selects the appropriate 1F or 2F signal depending on the programmed mode.

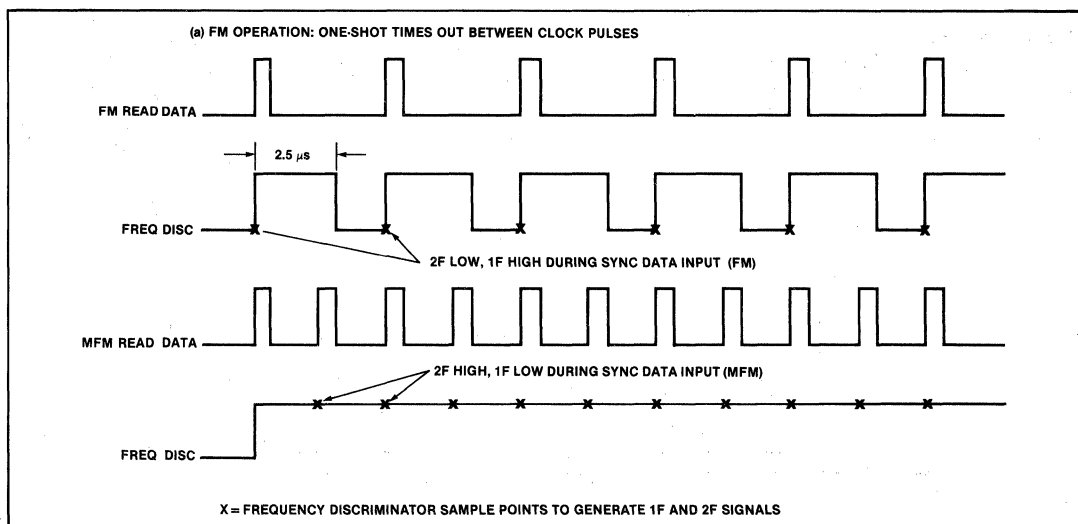


Figure 8. Input Data Frequency Discrimination

# APPLICATIONS

## Startup

The data separator is designed to require reception of eight valid sync bits (one sync byte) before enabling the PLL oscillator and attempting to synchronize with the input data stream (see Figure 9). This delay ensures that the PLL will not erroneously synchronize outside a valid sync field in the data stream if the VCO signal is enabled slightly early. The sync bit counter is asynchronously reset by the CNTEN signal when valid sync data is not being received by the drive.

Once the VCO signal is active and eight sync bits have been counted, the CNT8 signal is enabled. This signal turns on the PLL oscillator. Note that this oscillator starts synchronously with the rising edge of the disk input data (because CNT8 is synchronous with the data rising edge) and the oscillator also starts at its center frequency of 2 MHz (because the LM348 filter input is held at its center voltage of approximately 2.5 volts). This frequency is divided by two and four to generate the 2XBR signal (1 MHz for MFM and 500 KHz for FM).

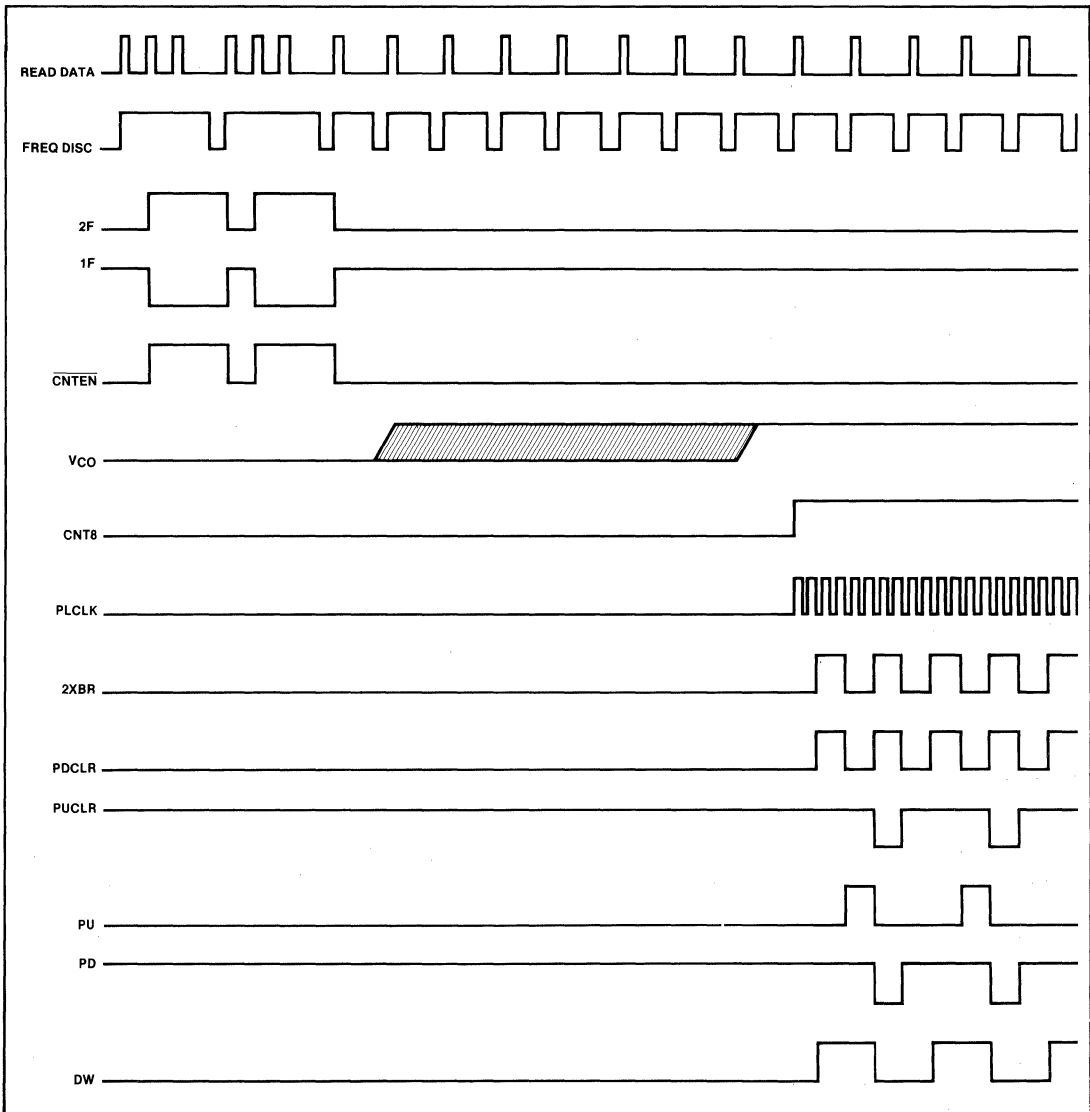


Figure 9. Typical Data Separator Startup Timing Diagram



## PLL Synchronization

At this point, the PLL is enabled and begins to synchronize with the input data stream. This synchronization is accomplished very simply in the following manner. The pump up (PU) signal is enabled on the rising edge of the READ DATA from the disk drive. (When the PLL is synchronized with the data stream, this point will occur at the same time as the falling edge of the 2XBR signal as shown in Figure 9). The PU signal is turned off and the PD signal is activated on the next rising edge of the 2XBR clock. With this scheme, the difference between PU active time and the PD active time is equal to the difference between the input bit rate and the PLL clock rate. Thus, if PU is turned on longer than PD is on, the input bit rate is faster than the PLL clock.

As long as PU and PD are both inactive, no charge is transferred to or from the LM358 input holding capacitor, and the PLL output frequency is maintained (the LM358 operational amplifier has a very high input impedance). Whenever PU is turned on, current flows from the +5 volt supply through a 20K resistor into the holding capacitor. When the PD signal is turned on, current flows from the holding capacitor to ground through a 20K resistor. In this manner, both the pump up and pump down charging rates are balanced.

The change in capacitor charge (and therefore voltage) after a complete PU/PD cycle is proportional to the difference between the PU and PD pulse widths and is also proportional to the frequency difference between the incoming data stream and the PLL oscillator. As the capacitor voltage is raised (PU active longer than PD), the PLL oscillator time constant (RC of the 96LS02) is modified by the filter output (LM358) to raise the oscillator frequency. As the capacitor voltage is lowered (PD active longer than PD), the oscillator frequency is lowered. If both frequencies are equal, the voltage on the holding capacitor does not change, and the PLL oscillator frequency remains constant.

## 6. AN INTELLIGENT DISKETTE DATA BASE SYSTEM

The system described in this application note is designed to function as an intelligent data base controller. The schematics for this data base unit are presented in Appendix A; a block diagram of the unit is illustrated in Figure 10. As designed, the unit can access over four million bytes of mass storage on four floppy disk drives (using a single 8272 FDC); the system can easily be expanded to four FDC devices (and 16 megabytes of on-line disk storage). Three serial data links are also included. These data links may be used by CRT terminals or other microprocessor systems to access the data base.

## Processor and Memory

A high-performance 8088 eight-bit microprocessor (operating at 5 MHz with no wait states) controls system operation. The 8088 was selected because of its memory addressing capabilities and its sophisticated string handling instructions. These features improve the speed of data base search operations. In addition, these capabilities allow the system to be easily upgraded with additional memory, disk drives, and if required, a bubble memory or winchester disk unit.

The schematics for the basic design provide 8K bytes of 2732A high-speed EPROM program storage and 8K bytes of disk directory and file buffer RAM. This memory can easily be expanded to 1 megabyte for performance upgrades.

An 8259A Programmable Interrupt Controller (PIC) is also included in the design to field interrupts from both the serial port and the FDC. This interrupt controller provides a large degree of programming flexibility for the implementation of data base functions in an asynchronous, demand driven environment. The PIC allows the system to accumulate asynchronous data base requests from all serial I/O ports while previously specified data base operations are currently in progress. This feature is made possible by the ability of the 8251A RXRDY signal to cause a processor interrupt. After receiving this interrupt, the processor can temporarily halt work on existing requests and enter the incoming information into a data base request buffer. Once the information has been entered into the buffer, the system can resume its previous processing.

In addition, the PIC permits some portions of data base requests to be processed in parallel. For example, once a disk record has been loaded into a memory buffer, a memory search can proceed in parallel with the loading of the next record. After the FDC completes the record transfer, the memory search will be interrupted and the processor can begin another disk transfer before resuming the memory search.

The bus structure of the system is split into three functional buffered units. A 20-bit address from the processor is latched by three-state transparent 74LS373 devices. When the processor is in control of the address and data busses, these devices are output enabled to the system buffered address bus. All I/O devices are placed directly on the local data bus. Finally, the memory data bus is isolated from the local data bus by an 8286 octal transceiver. The direction of this transceiver is determined by the Memory Read signal, while its output enable is activated by a Memory Read or Memory Write command.

# APPLICATIONS

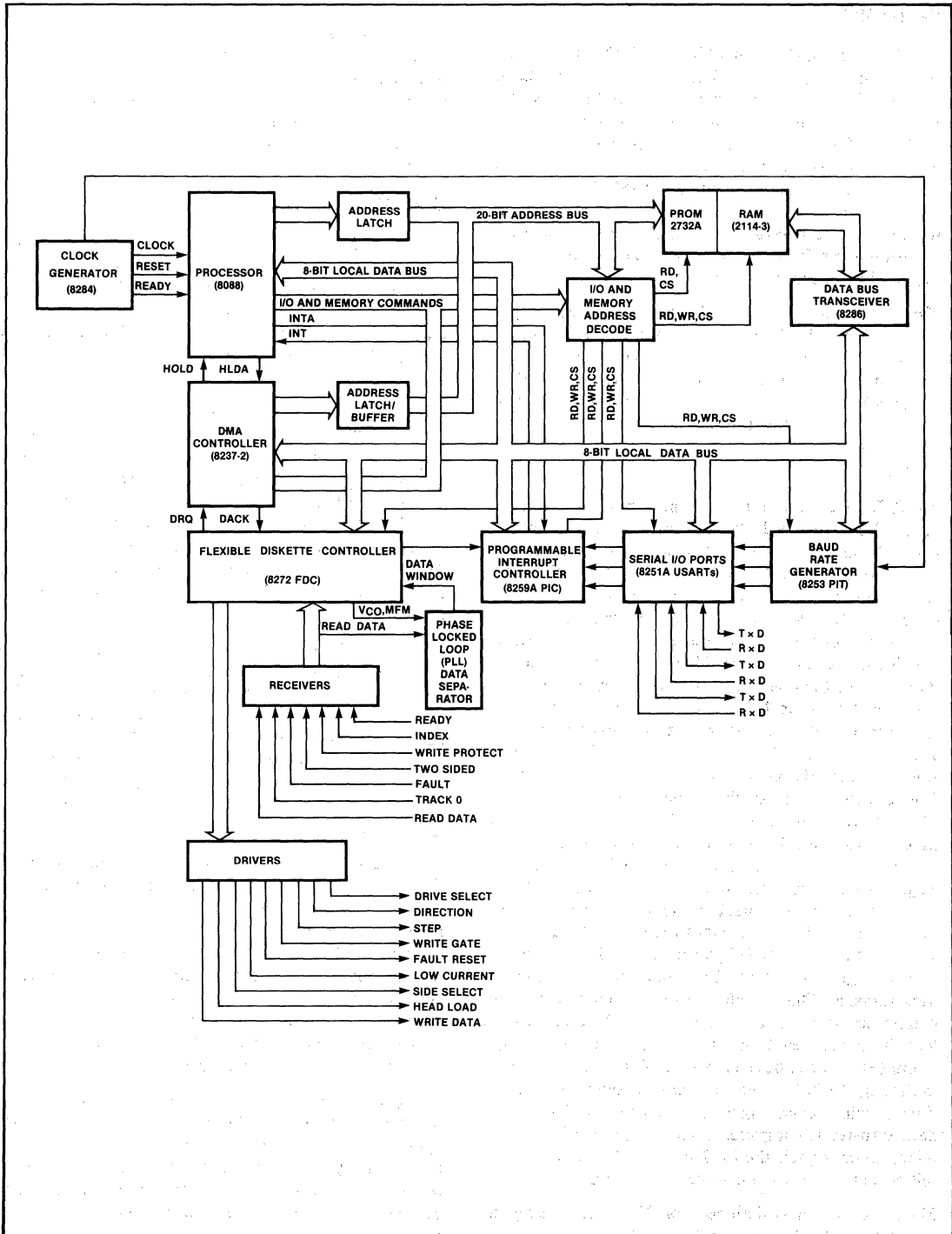


Figure 10. Intelligent Data Base Block Diagram

## Serial I/O

The three RS-232-C compatible serial I/O ports operate at software-programmable baud rates to 19.2K. Each I/O port is controlled by an 8251A USART (Universal Synchronous/Asynchronous Receiver/Transmitter). Each USART is individually programmable for operation in many synchronous and asynchronous serial data transmission formats (including IBM Bi-sync). In operation, USART error detection circuits can check for parity, data overrun, and framing errors. An 8253 Programmable Interval Timer is employed to generate the baud rates for the serial I/O ports.

The Transmitter Ready and Receiver Ready output signals of the 8251As are routed to the interrupt inputs of the 8259A interrupt controller. These signals interrupt processor execution when a data byte is received by a USART and also when the USART is ready to accept another data byte for transmission.

## DMA

The 8272 FDC interfaces to system memory by means of an 8237-2 high-speed DMA controller. Transfers between the disk controller and memory also operate with no wait states when 2114-3 (150 ns) or faster static RAM is used. In operation, the 8272 presents a DMA request to the 8237 for every byte of data to be transferred. This request causes the 8273 to present a HOLD request to the 8088. As soon as the 8088 is able to relinquish data/address bus control, the processor signals a HOLD acknowledge to the 8237. The 8237 then assumes control over the data and address busses. After latching the address for the DMA transfer, the 8237 generates simultaneous I/O Read and Memory Write commands (for a disk read) or simultaneous I/O Write and Memory Read commands (for a disk write). At the same time, the 8272 is selected as the I/O device by means of the DMA acknowledge signal from the 8237. After this single byte has been transferred between the FDC and memory, the DMA controller releases the data/address busses to the 8088 by deactivating the HOLD request. In a short period of time (13  $\mu$ s for double-density and 27  $\mu$ s for single-density) the FDC requests a subsequent data transfer. This transfer occurs in exactly the same manner as the previous transfer. After all data transfers have been completed (specified by the word count programmed into the 8237 before the FDC operation was initiated), the 8237 signals a terminal count (EOP pin). This terminal count signal informs the 8272 that the data transfer is complete. Upon reception of this terminal count signal, the 8272 halts DMA requests and initiates an "operation complete" interrupt.

Since the system is designed for 20-bit addressing, a four-bit DMA-address latch is included as a processor

addressable I/O port. The processor writes the upper four DMA address bits before a data transfer. When the DMA controller assumes bus control, the contents of this latch are output enabled on the upper four bits of the address bus. The only restriction in the use of this address latch is that a single disk read or write transfer cannot cross a 64K memory boundary.

## Disk Drive Interface

The 8272 FDC may be interfaced to a maximum of four eight-inch floppy disk drives. Both single- and double-density drives are accommodated using the data separation circuit described in section 5. In addition, single- or dual-sided disk drives may be used. The 8272 is driven by an 8 MHz crystal controller clock produced by an 8224 clock generator.

Drive select signals are decoded by means of a 74LS139 from the DS0, DS1 outputs of the FDC. The fault reset, step, low current, and direction outputs to the disk drives are generated from the FR/STEP, LCT/DIR, and RW/SEEK FDC output signals by means of a 74LS240. The other half of the 74LS240 functions as an input multiplexer for the disk write protect, two-sided, fault, and track zero status signals. These signals are multiplexed into the WP/TS and FLT/TRK0 inputs to the 8272.

The 8272 write clock (WR CLK) is generated by a ring counter/multiplexer combination. The write clock frequency is 1 MHz for MFM recording and 500 KHz for FM recording (selected by the MFM output of the 8272). The pulse width is a constant 250 ns. The write clock is constantly generated and input to the FDC (during both read and write operations). The FDC write enable output (WE) is transmitted directly to the write gate disk drive input.

Write data to the disk drive is preshifted (according to the PS0, PS1 FDC outputs) by the combination of a 74LS175 four-bit latch and a 74LS153 multiplexer. The amount of preshift is completely controlled within the 8272 FDC. Three cases are possible: the data may be written one clock cycle early, one clock cycle late, or with no preshift. The data preshift circuit is activated by the FDC only in the double-density mode. The preshift is required to cancel predictable playback data shifts when recorded data is later read from the floppy disk.

A single 50-conductor flat cable connects the board to the floppy disk drives. FDC outputs are driven by 7438 open collector high-current line-drivers. These drivers are resistively terminated on the last disk drive by means of a 150 ohm resistor to +5V. The line receivers are 7414 Schmitt triggered inverters with 150 ohm pull-up resistors on board.

# APPLICATIONS

## 7. SPECIAL CONSIDERATIONS

This section contains a quick review of key features and issues, most of which have been mentioned in other sections of this application note. Before designing with the 8272 FDC, it is advisable that the information in this section be completely understood.

### 1. Multi-Sector Transfers

The 8272 always operates in a multi-sector transfer mode. The 8272 continues to transfer data until the TC input is activated. In a DMA configuration, the TC input of the 8272 must always be connected to the EOP/TC output of the DMA controller. When multiple DMA channels are used on a single DMA controller, EOP must be gated with the select signal for the proper FDC. If the TC signal is not gated, a terminal count on another channel will abort FDC operation.

In a processor driven configuration with no DMA controller, the system must count the transfers and supply a TC signal to the FDC. In a DMA environment, ORing a programmable TC with the TC from the DMA controller is a convenient means of ensuring that the processor may always gain control of the FDC (even if the diskette system hangs up in an abnormal manner).

### 2. Processor Command/Result Phase Interface

In the command phase, the processor must write the exact number of parameters in the exact order shown in Table 5. During the result phase, the processor must read the complete result status. For example, the Format Track command requires six command bytes and presents seven result bytes. The 8272 will not accept a new command until all result bytes are read. Note that the number of command and result bytes varies from command-to-command. Command and result phases cannot be shortened.

During both the command and result phases, the Main Status Register must be read by the processor before each byte of information is read from, or written to, the FDC Data Register. Before each command byte is written, DIO (bit 6) must be low (indicating a data transfer from the processor) and RQM (bit 7) must be high (indicating that the FDC is ready for data). During the result phase, DIO must be high (indicating a data transfer to the processor) and RQM must also be high (indicating that data is ready for the processor).

**NOTE:** After the 8272 receives a command byte, the RQM flag may remain set for 12 microseconds (with an 8 MHz clock). Software should not attempt to read the Main Status Register before this time interval has elapsed; otherwise, the software will erroneously assume that the FDC is ready to accept the next byte.

### 3. Sector Sizes

The 8272 does not support 128 byte sectors in the MFM (double-density) mode.

### 4. Write Clock

The FDC Write Clock input (WR CLK) must be present at all times.

### 5. Reset

The FDC Reset input (RST) must be held active during power-on reset while the RD and WR inputs are active. If the reset input becomes inactive while RD and WR are still active, the 8272 enters the test mode. Once activated, the test mode can only be deactivated by a power-down condition.

### 6. Drive Status

The 8272 constantly polls (starting after the power-on reset) all drives for changes in the drive ready status. At power-on, the FDC assumes that all drives are not ready. If a drive application requires that the ready line be strapped active, the FDC will generate an interrupt immediately after power is applied.

### 7. Gap Length

Only the gap 3 size is software programmable. All other gap sizes are fixed. In addition, different gap 3 sizes must be specified in format, read, write, and scan commands. Refer to Section 3 and Table 9 for gap size recommendations.

### 8. Seek Command

The drive busy flag in the Main Status Register remains set after a Seek command is issued until the Sense Interrupt Status command is issued (following reception of the seek complete interrupt).

The FDC does not perform implied seeks. Before issuing data read or write commands, the read/write head must be positioned over the correct cylinder. If the head is not positioned correctly, a cylinder address error is generated.

After issuing a step pulse, the 8272 resumes drive status polling. For correct stepper operation in this mode, the stepper motor must be constantly enabled. (Most drives provide a jumper to permit the stepper motor to be constantly enabled.)

### 9. Step Rate

The 8272 can emit a step pulse that is one millisecond faster than the rate programmed by the SRT parameter in the Specify command. This action may cause subsequent sector not found errors. The step rate time should be programmed to be 1 ms longer than the step rate time required by the drive.

### 10. Cable Length

A cable length of less than 10 feet is recommended for drive interfacing.

## 11. Scan Commands

The current 8272 has several problems when using the scan commands. These commands should not be used at this time.

## 12. Interrupts

When the processor receives an interrupt from the FDC, the FDC may be reporting one of two distinct events:

- a) The beginning of the result phase of a previously requested read, write, or scan command.
- b) An asynchronous event such as a seek/recalibrate completion, an attention, an abnormal command termination, or an invalid command.

These two cases are distinguished by the FDC busy flag (bit 4) in the Main Status Register. If the FDC busy flag is high, the interrupt is of type (a). If the FDC busy flag is low, the interrupt was caused by an asynchronous event (b).

A single interrupt from the FDC may signal more than one of the above events. After receiving an interrupt, the processor must continue to issue Sense Interrupt Status commands (and service the resulting conditions) until an invalid command code is received. In this manner, all "hidden" interrupts are ferreted out and serviced.

## 13. Skip Flag (SK)

The skip flag is used during the execution of Read Data, Read Deleted Data, Read Track, and various Scan commands. This flag permits the FDC to skip unwanted sectors on a disk track.

When performing a Read Data, Read Track, or Scan command, a high SK flag indicates that the FDC is to skip over (not transfer) any sector containing a deleted data address mark. A low SK flag indicates that the FDC is to terminate the command (after reading all the data in the sector) when a deleted data address mark is encountered.

When performing a Read Deleted Data command, a high SK flag indicates that sectors containing normal data address marks are to be skipped. Note that this is just the opposite situation from that described in the last paragraph. When a data address mark is encountered during a Read Deleted Data command (and the SK flag

is low), the FDC terminates the command after reading all the data in the sector.

## 14. Bad Track Maintenance

The 8272 does not internally maintain bad track information. The maintenance of this information must be performed by system software. As an example of typical bad track operation, assume that a media test determines that track 31 and track 66 of a given floppy disk are bad. When the disk is formatted for use, the system software formats physical track 0 as logical cylinder 0 ( $C=0$  in the command phase parameters), physical track 1 as logical track 1 ( $C=1$ ), and so on, until physical track 30 is formatted as logical cylinder 30 ( $C=30$ ). Physical track 31 is bad and should be formatted as logical cylinder FF (indicating a bad track). Next, physical track 32 is formatted as logical cylinder 31, and so on, until physical track 67 is formatted as logical cylinder 64. Next, bad physical track 66 is formatted as logical cylinder FF (another bad track marker), and physical track 67 is formatted as logical cylinder 65. This formatting continues until the last physical track (77) is formatted as logical cylinder 75. Normally, after this formatting is complete, the bad track information is stored in a prespecified area on the floppy disk (typically in a sector on track 0) so that the system will be able to recreate the bad track information when the disk is removed from the drive and reinserted at some later time.

To illustrate how the system software performs a transfer operation disk with bad tracks, assume that the disk drive head is positioned at track 0 and the disk described above is loaded into the drive. If a command to read track 36 is issued by an application program, the system software translates this read command into a seek to physical track 37 (since there is one bad track between 0 and 36, namely 31) followed by a read of logical cylinder 36. Thus, the cylinder parameter C is set to 37 for the Seek command and 36 for the Read Sector command.

## 15. Head Load versus Head Settle Times

The 8272 does not permit separate specification of the head load time and the head settle time. When the Specify command is issued for a given disk drive, the proper value for the HLT parameter is the maximum of the head load time and the head settle time.

# APPLICATIONS

---

## APPENDIX

# APPLICATIONS

## Power Distribution

Part	Ref Desig	+5	GND	+12	-12
8088	A2	40	1,20		
8224	I6	9,16	8		
8237-2	A6	31	20		
8251A	A9,B9,C9	26	4		
8253-5	A10	24	12		
8259A	B10	28	14		
8272	D10	40	20		
8284	A1	18	9		
8286	B6,F4	20	10		
2114	F1,F2,G1,G2,H1,H2,I1,I2	18	9		
2732A	D1,D2	24	12		
74LS00	E1	14	7		
74LS04	B2,E6,E8,F8	14	7		
74LS27	E2,E5	14	7		
74LS32	B1	14	7		
74LS74	A4,G5,H6	14	7		
74LS138	F3	16	8		
74LS139	E10	16	8		
74LS153	I3	16	8		
74LS157	F6	16	8		
74LS164	F5	14	7		
74LS173	G3	16	8		
74LS175	G4	16	8		
74LS240	G10	20	10		
74LS257	D3	16	8		
74LS367	C3,E9	16	8		
74LS373	B4,C4,D4,C6	20	10		
74LS393	I5,F7	14	7		
74S08	E4	14	7		
74S138	D6,E3	16	8		
7414	H7	14	7		
7438	H8,H9,H10	14	7		
1488	H3		7	14	1
1489	H4	14	7		
96LS02	G7	16	8		
96LS02	G6			16	8
LM358	H5			8	4

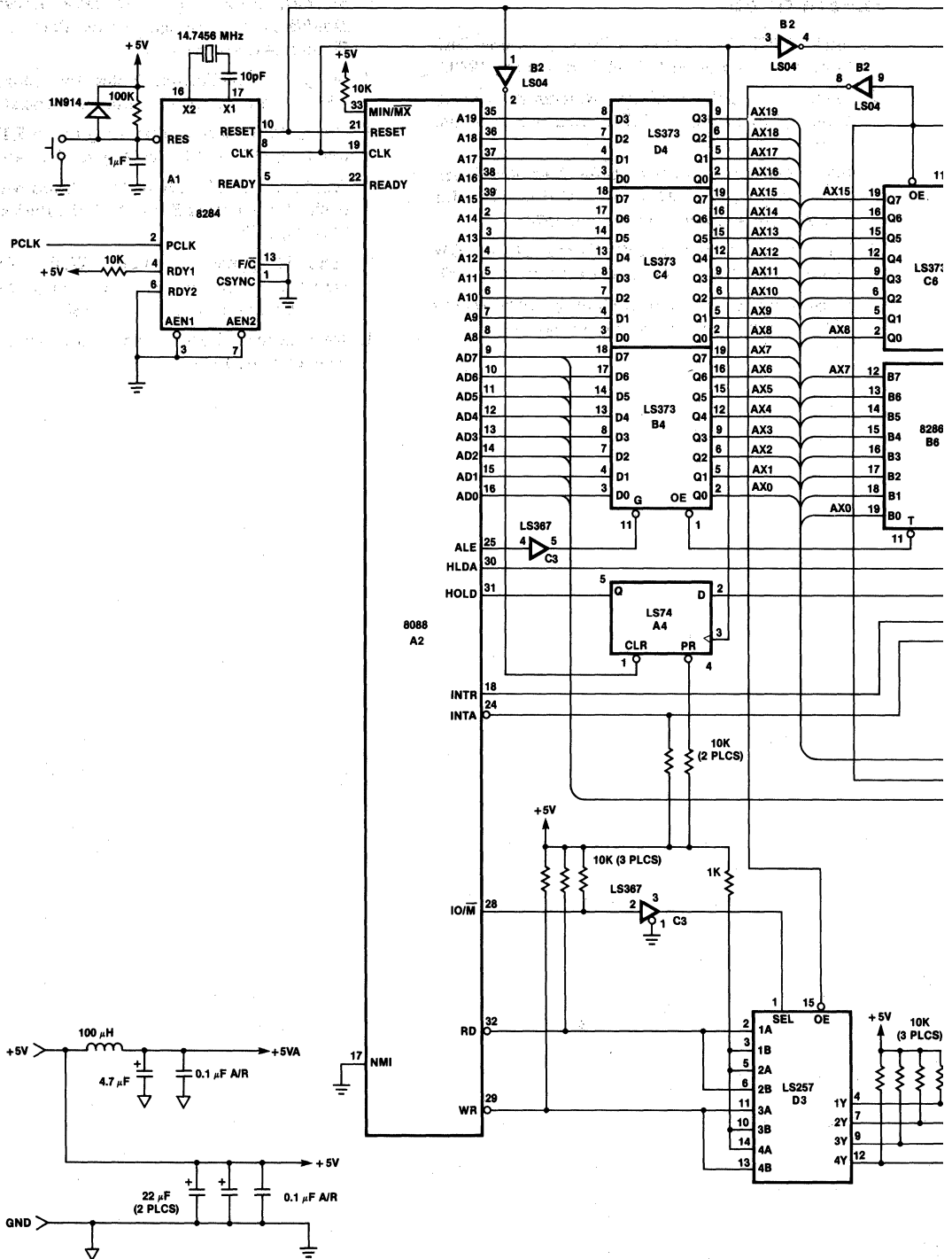
# APPLICATIONS

---

## REFERENCES

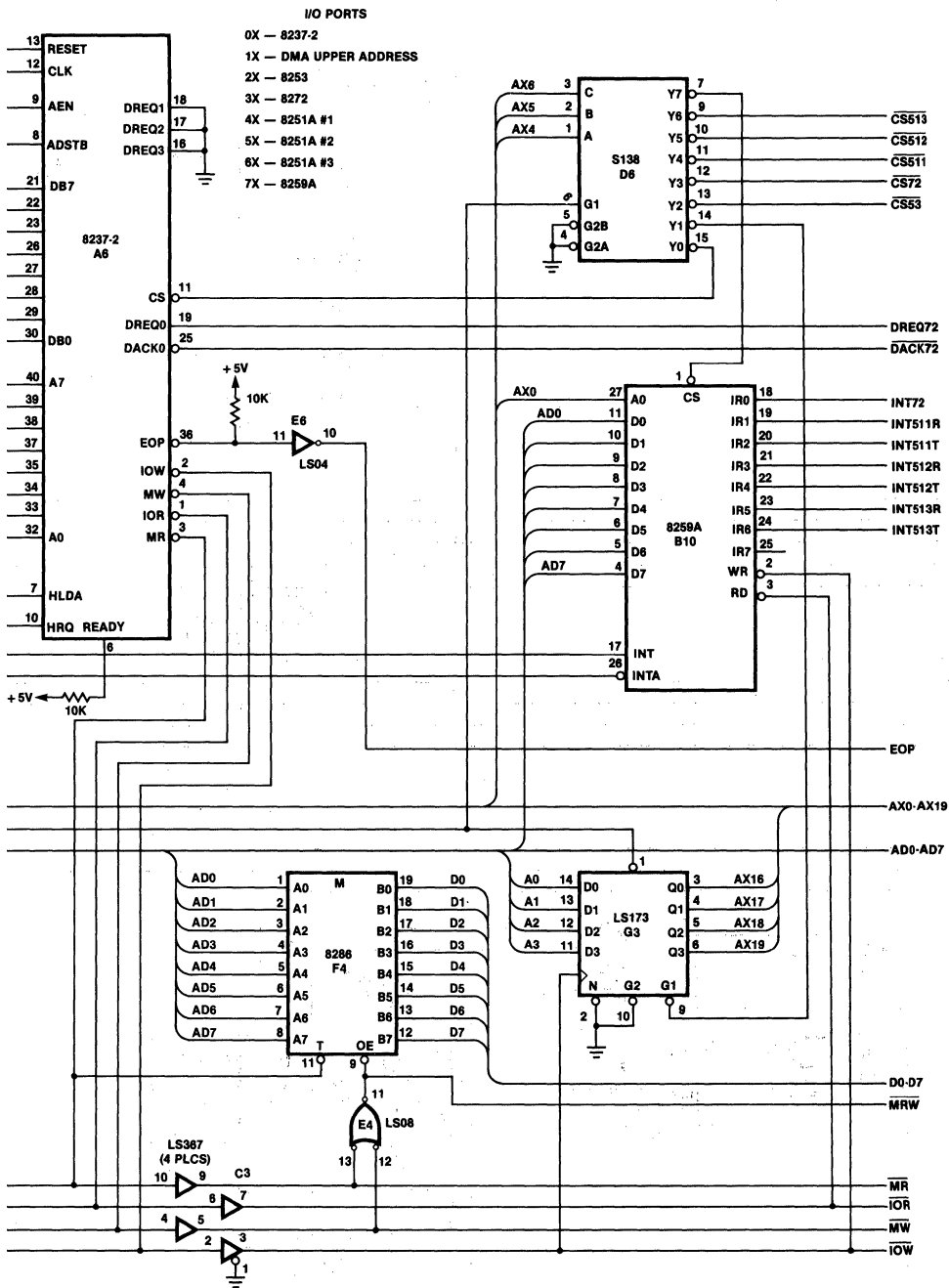
1. Intel, "8272 Single/Double Density Floppy Disk Controller Data Sheet," Intel Corporation, 1980.
2. Intel, *iSBC 208 Hardware Reference Manual*, Manual Order No. 143078, Intel Corporation, 1980.
3. Intel, *iSBC 204 Flexible Diskette Controller Hardware Reference Manual*, Manual Order No. 9800568A, Intel Corporation, 1978.
4. Shugart, *SA800/801 Diskette Storage Drive OEM Manual*, Part No. 50574, Shugart Associates, 1977.
5. Shugart, *SA800/801 Diskette Storage Drive Theory of Operations*, Part No. 50664, Shugart Associates, 1977.
6. Shugart, *SA800 Series Diskette Storage Drive Double Density Design Guide*, Part No. 39000, Shugart Associates, 1977.
7. Shugart, "Application Notes for Shugart Dual VFO," Part No. 39101, Shugart Associates, 1980.
8. Pertec, "Soft-sector Formatting for PERTEC Flexible Disk Drives," Pertec Application Note, 1977.
9. Austin Lesea and Rodney Zaks, "Floppy-disc Controller Design Must Begin With the Basics," EDN, May 20, 1978.
10. John Hoepfner and Larry Wall, "Encoding/Decoding Techniques Double Floppy Disc Capacity," Computer Design, Feb 1980.
11. John Zarrella, *System Architecture*, Mirocomputer Applications, 1980.



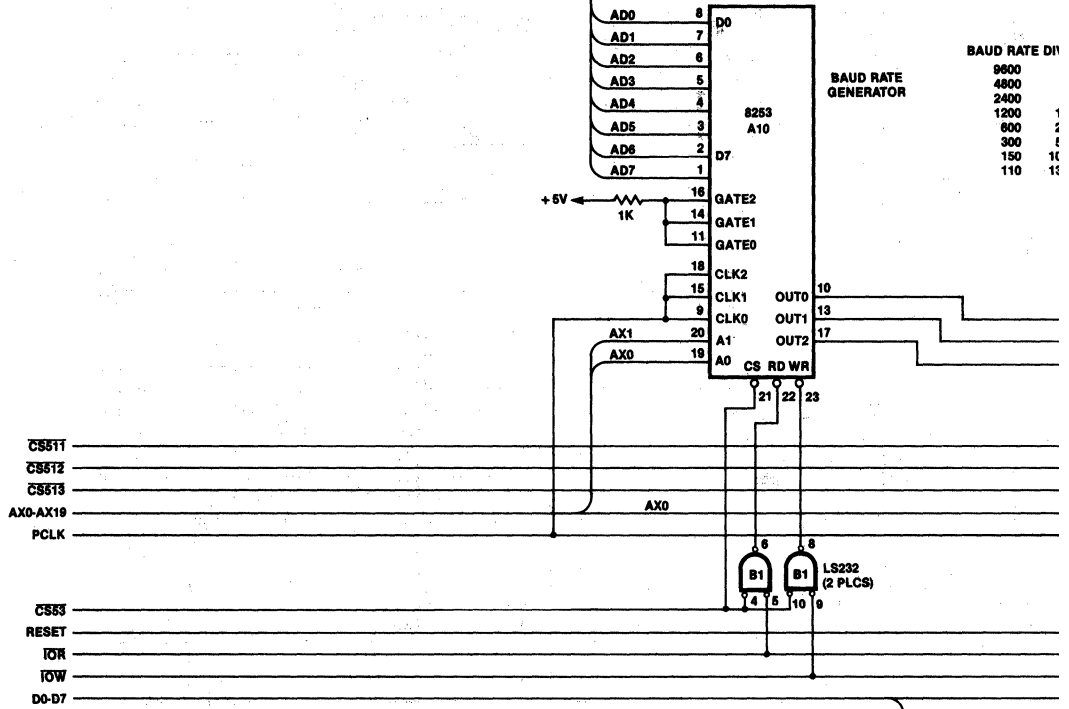


# APPLICATIONS

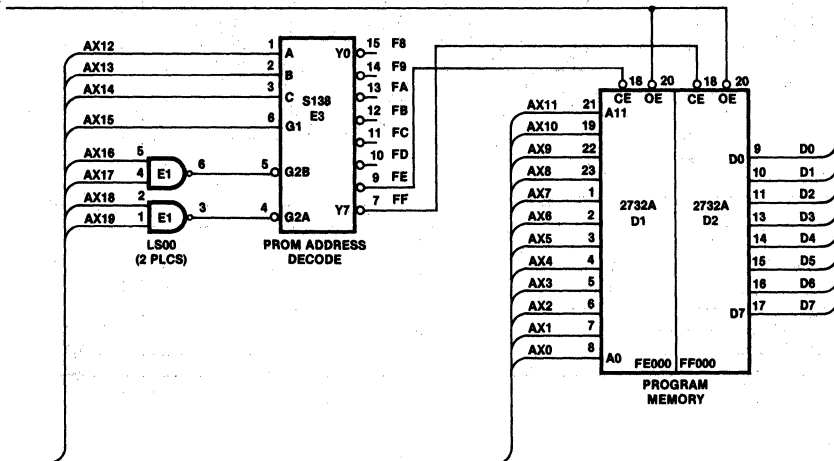
RESET



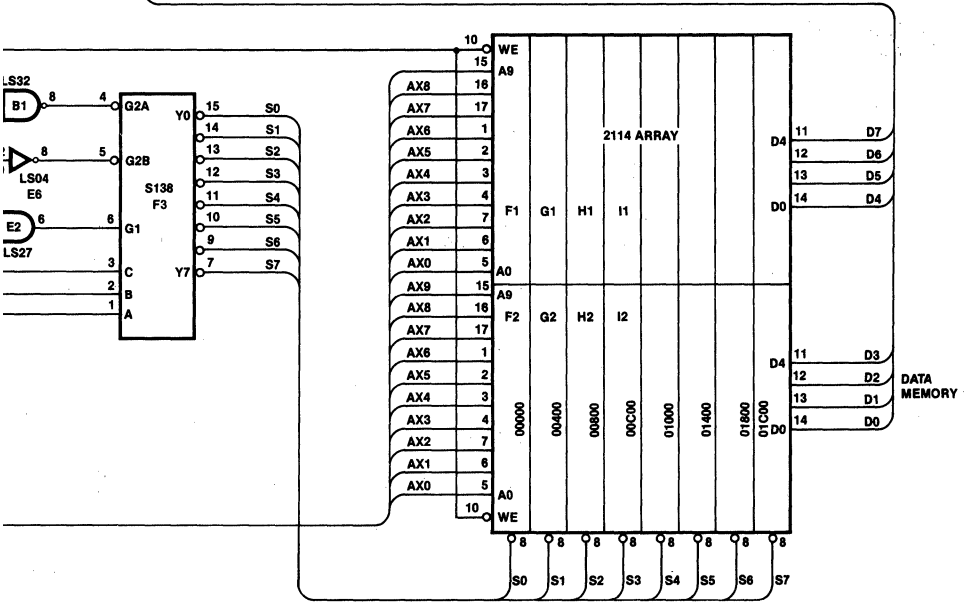
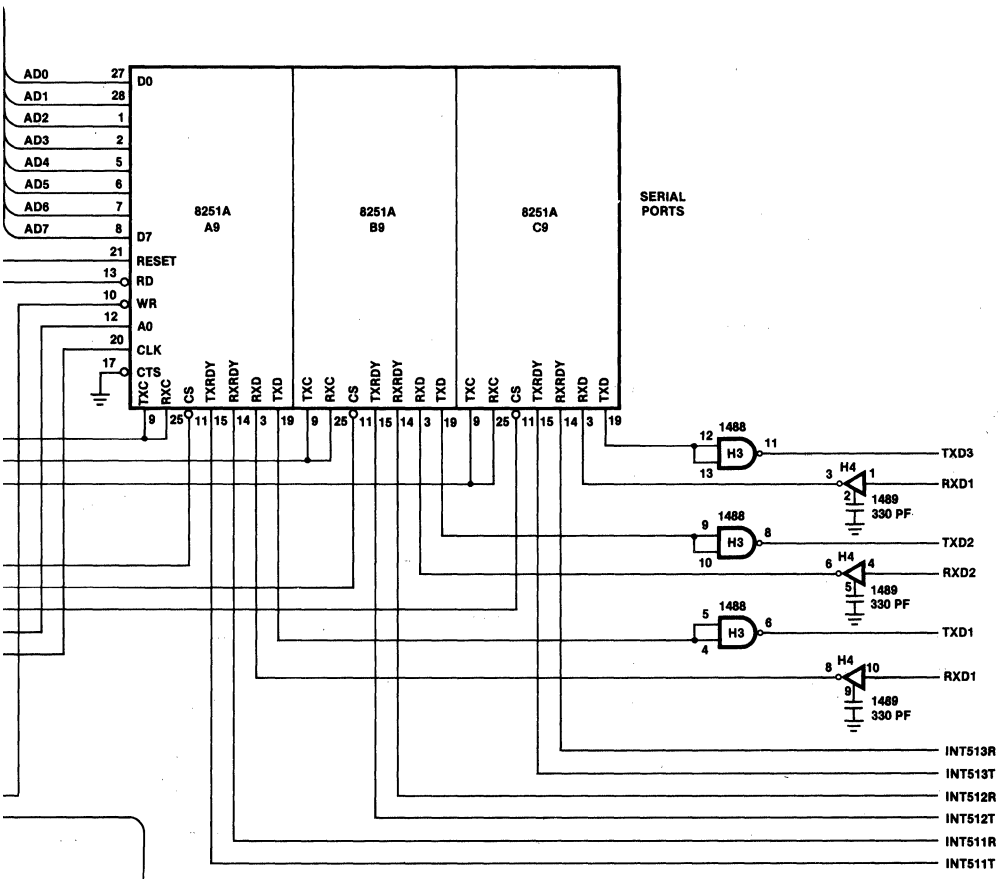
AD0-AD7

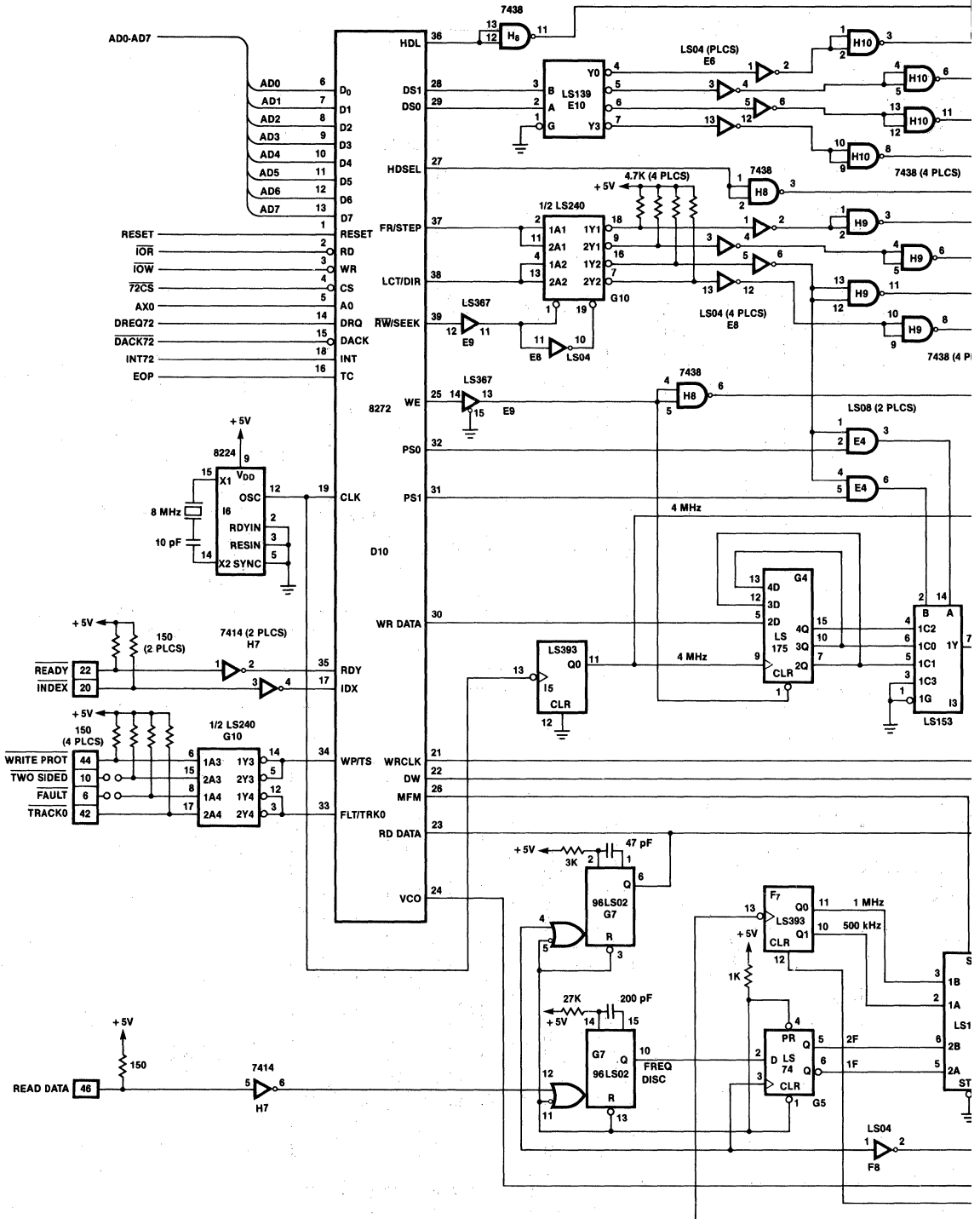


MRW  
MWR  
MR



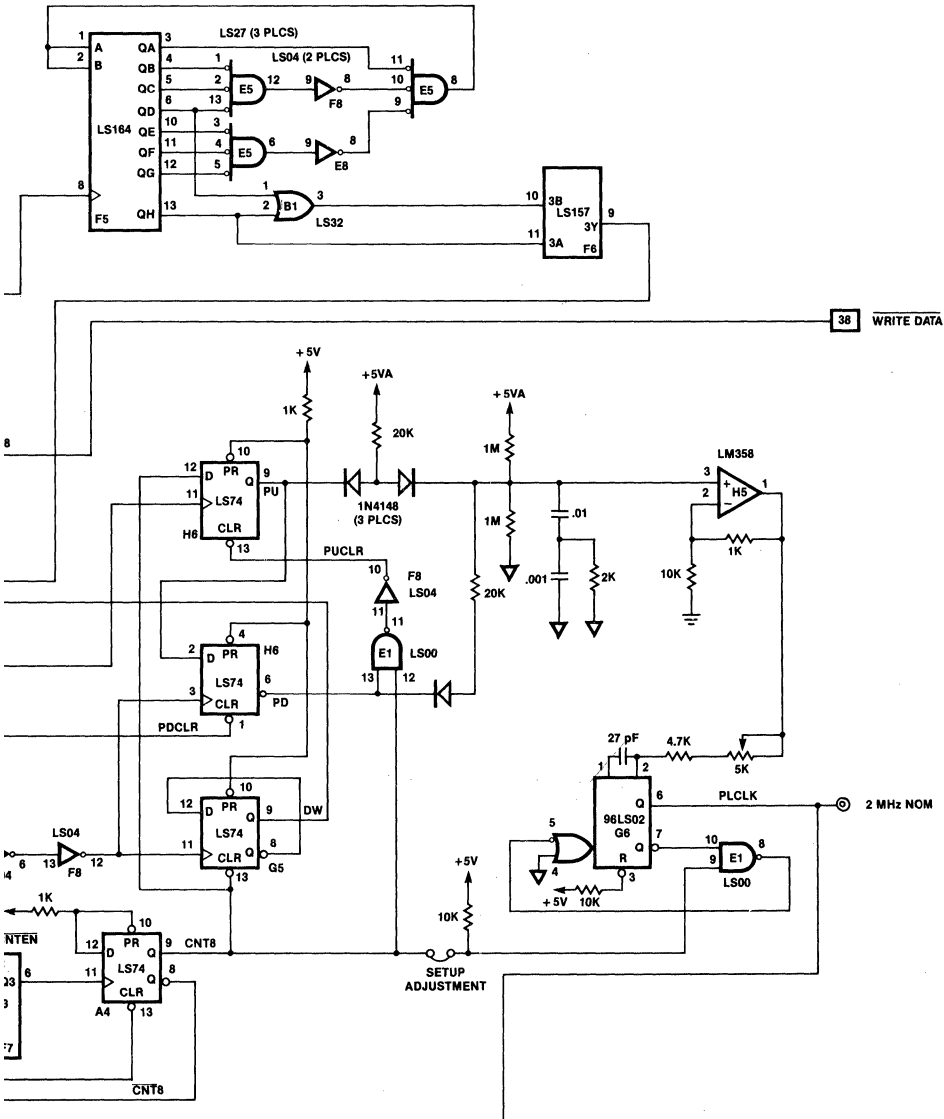
# APPLICATIONS





# APPLICATIONS

DSEL0	28	DRIVE SELECT0
DSEL1	28	DRIVE SELECT1
DSEL2	30	DRIVE SELECT2
DSEL3	32	DRIVE SELECT3
DIR	34	DIRECTION SELECT
STEP	36	STEP
WRGT	40	WRITE GATE
FRES	4	FAULT RESET
LCT	2	LOW CURRENT
SSEL	14	SIDE SELECT
HDL	18	HEAD LOAD



---

# Software Design and Implementation of Floppy Disk Subsystems

## Contents

<b>1. INTRODUCTION</b>	<b>2-174</b>
The Physical Interface Level	
The Logical Interface Level	
The File System Interface Level	
Scope of this Note	
<b>2. DISK I/O TECHNIQUES</b>	<b>2-179</b>
FDC Data Transfer Interface	
Overlapped Operations	
Buffers	
<b>3. THE 8272 FLOPPY DISK CONTROLLER</b>	<b>2-185</b>
Floppy Disk Commands	
Interface Registers	
Command/Result Phases	
Execution Phase	
Multi-sector and Multi-track Transfers	
Drive Status Polling	
Command Details	
Invalid Commands	
<b>4. 8272 PHYSICAL INTERFACE SOFTWARE</b>	<b>2-197</b>
INITIALIZE\$DRIVERS	
EXECUTE\$DOCB	
FDCINT	
OUTPUT\$CONTROLS\$TO\$DMA	
OUTPUT\$COMMAND\$TO\$FDC	
INPUT\$RESULT\$FROM\$FDC	
OUTPUT\$BYTE\$TO\$FDC	
INPUT\$BYTE\$FROM\$FDC	
FDC\$READY\$FOR\$COMMAND	
FDC\$READY\$FOR\$RESULT	
OPERATION\$CLEAN\$UP	
Modifications for Polling Operation	
<b>5. 8272 LOGICAL INTERFACE SOFTWARE</b>	<b>2-204</b>
SPECIFY	
RECALIBRATE	
SEEK	
FORMAT	
WRITE	
READ	
Coping With Errors	

---

## **Contents (Continued)**

<b>6. FILE SYSTEMS</b>	<b>2-207</b>
File Allocation	
The Intel File System	
Disk File System Functions	
<b>7. KEY 8272 SOFTWARE     INTERFACING CONSIDERATIONS</b>	<b>2-212</b>
<b>REFERENCES</b>	<b>2-215</b>
<b>APPENDIX A—8272 FDC     DEVICE DRIVER SOFTWARE</b>	<b>2-216</b>
<b>APPENDIX B—8272 FDC     EXERCISER PROGRAM</b>	<b>2-225</b>
<b>APPENDIX C—8272 DRIVER FLOWCHARTS</b>	<b>2-232</b>



# APPLICATIONS

---

## 1. Introduction

Disk interface software is a major contributor to the efficient and reliable operation of a floppy disk subsystem. This software must be a well-designed compromise between the needs of the application software modules and the capabilities of the floppy disk controller (FDC). In an effort to meet these requirements, the implementation of disk interface software is often divided into several levels of abstraction. The purpose of this application note is to define these software interface levels and describe the design and implementation of a modular and flexible software driver for the 8272 FDC. This note is a companion to AP-116, "An Intelligent Data Base System Using the 8272."

### The Physical Interface Level

The software interface level closest to the FDC hardware is referred to as the physical interface level. At this level, interface modules (often called disk drivers or disk handlers) communicate directly with the FDC device. Disk drivers accept floppy disk commands from other software modules, control and monitor the FDC execution of the commands, and finally return operational status information (at command termination) to the requesting modules.

In order to perform these functions, the drivers must support the bit/byte level FDC interface for status and data transfers. In addition, the drivers must field, classify, and service a variety of FDC interrupts.

### The Logical Interface Level

System and application software modules often specify disk operation parameters that are not directly compatible with the FDC device. This software incompatibility is typically caused by one of the following:

1. The change from an existing FDC to a functionally equivalent design. Replacing a TTL based controller with an LSI device is an example of a change that may result in software incompatibilities.
2. The upgrade of an existing FDC subsystem to a higher capability design. An expansion from a single-sided, single-density system to a dual-sided, double-density system to increase data storage capacity is an example of such a system change.
3. The abstraction of the disk software interface to avoid redundancy. Many FDC parameters (in particular the density, gap size, number of sectors per track and number of bytes per sector) are fixed for a floppy disk (after formatting). In fact, in many systems these parameters are never changed during the life of the system.

# APPLICATIONS

---

4. The requirement to support a software interface that is independent of the type of disk attached to the system. In this case, a system generated ("logical") disk address (drive, head, cylinder, and sector numbers) must be mapped into a physical floppy disk address. For example, to switch between single- and dual-sided disks, it may be easier and more cost-effective for the software to treat the dual-sided disk as containing twice as many sectors per track (52) rather than as having two sides. With this technique, accesses to sectors 1 through 26 are mapped onto head 0 while accesses to sectors 27 through 52 are mapped onto head 1.
5. The necessity of supporting a bad track map. Since bad tracks depend on the disk media, the bad track mapping varies from disk to disk. In general, the system and application software should not be concerned with calculating bad track parameters. Instead, these software modules should refer to cylinders logically (0 through 76). The logical interface level procedures must map these cylinders into physical cylinder positions in order to avoid the bad tracks.

The key to logical interface software design is the mapping of the "logical disk interface" (as seen by the application software) into the "physical disk interface" (as implemented by the floppy disk drivers). This logical to physical mapping is tightly coupled to system software design and the mapping serves to isolate both applications and system software from the peculiarities of the FDC device. Typical logical interface procedures are described in Table 1.

## The File System Interface Level

The file system typically comprises the highest level of disk interface software used by application programs. The file system is designed to treat the disk as a collection of named data areas (known as files). These files are cataloged in the disk directory. File system interface software permits the creation of new files and the deletion of existing files under software control. When a file is created, its name and disk address are entered into the directory; when a file is deleted, its name is removed from the directory. Application software requests the use of a file by executing an OPEN function. Once opened, a file is normally reserved for use by the requesting program or task and the file cannot be reopened by other tasks. When a task no longer needs to use an open file, the task closes the file, releasing it for use by other tasks.

Most file systems also support a set of file attributes that can be specified for each file. File attributes may be used to protect files (e.g., the WRITE PROTECT attribute ensures that an existing file cannot accidentally be overwritten) and to supply system configuration information (e.g., a FORMAT attribute may specify that a file should automatically be created on a new disk when the disk is formatted).

At the file system interface level, application programs need not be explicitly aware of disk storage allocation techniques, block sizes, or file coding strategies. Only a "file name" must be presented in order to open, read or write, and subsequently close a file. Typical file system functions are listed in Table 2.

# APPLICATIONS

**Table 1: Examples of Logical Interface Procedures**

Name	Description
FORMAT DISK	Controls physical disk formatting for all tracks on a disk. Formatting adds FDC recognized cylinder, head, and sector addresses as well as address marks and data synchronization fields (gaps) to the floppy disk media.
RECALIBRATE	Moves the disk read/write head to track 0 (at the outside edge of the disk).
SEEK	Moves the disk read/write head to a specified logical cylinder. The logical and physical cylinder numbers may be different if bad track mapping is used.
READ STATUS	Indicates the status of the floppy disk drive and media. One important use of this procedure is to determine whether a floppy disk is dual-sided.
READ SECTOR	Reads one or more complete sectors starting at a specified disk address (drive, head, cylinder, and sector).
WRITE SECTOR	Writes one or more complete sectors starting at a specified disk address (drive, head, cylinder, and sector).

# APPLICATIONS

**Table 2: Disk File System Functions**

Name	Description
OPEN	Prepare a file for processing. If the file is to be opened for input and the file name is not found in the directory, an error is generated. If the file is opened for output and the file name is not found in the directory, the file is automatically created.
CLOSE	Terminate processing of an open file.
READ	Transfer data from an open file to memory. The READ function is often designed to buffer one or more sectors of data from the disk drive and supply this data to the requesting program, as required.
WRITE	Transfer data from memory to an open file. The WRITE function is often designed to buffer data from the application program until enough data is available to fill a disk sector.
CREATE	Initialize a file and enter its name and attributes into the file directory.
DELETE	Remove a file from the directory and release its storage space.
RENAME	Change the name of a file in the directory.
ATTRIBUTE	Change the attributes of a file.
LOAD	Read a file of executable code into memory.
INITDISK	Initialize a disk by formatting the media and establishing the directory file, the bit map file, and other system files.

# APPLICATIONS

---

## Scope of this Note

This application note directly addresses the logical and physical interface levels. A complete 8272 driver (including interrupt service software) is listed in Appendix A. In addition, examples of recalibrate, seek, format, read, and write logical interface level procedures are included as part of the exerciser program found in Appendix B. Wherever possible, specific hardware configuration dependencies are parametrized to provide maximum flexibility without requiring major software changes.

# APPLICATIONS

---

## 2. Disk I/O Techniques

One of the most important software aspects of disk interfacing is the fixed sector size. (Sector sizes are fixed when the disk is formatted.) Individual bytes of disk storage cannot be read/written; instead, complete sectors must be transferred between the floppy disk and system memory.

Selection of the appropriate sector size involves a tradeoff between memory size, disk storage efficiency, and disk transfer efficiency. Basically, the following factors must be weighed:

1. Memory size. The larger the sector size, the larger the memory area that must be reserved for use during disk I/O transfers. For example, a 1K byte disk sector size requires that at least one 1K memory block be reserved for disk I/O.
2. Disk Storage efficiency. Both very large and very small sectors can waste disk storage space as follows. In disk file systems, space must be allocated somewhere on the disk to link the sectors of each file together. If most files are composed of many small sectors, a large amount of linkage overhead information is required. At the other extreme, when most files are smaller than a single disk sector, a large amount of space is wasted at the end of each sector.
3. Disk transfer efficiency. A file composed of a few large sectors can be transferred to/from memory more efficiently (faster and with less overhead) than a file composed of many small sectors.

Balancing these considerations requires knowledge of the intended system applications. Typically, for general purpose systems, sector sizes from 128 bytes to 1K bytes are used. For compatibility between single-density and double-density recording with the 8272 floppy disk controller, 256 byte sectors or 512 byte sectors are most useful.

### FDC Data Transfer Interface

Three distinct software interface techniques may be used to interface system memory to the FDC device during sector data transfers:

1. DMA - In a DMA implementation, the software is only required to set up the DMA controller memory address and transfer count, and to initiate the data transfer. The DMA controller hardware handshakes with the processor/system bus in order to perform each data transfer.
2. Interrupt Driven - The FDC generates an interrupt when a data byte is ready to be transferred to memory, or when a data byte is needed from memory. It is the software's responsibility to perform appropriate memory reads/writes in order to transfer data from/to the FDC upon receipt of the interrupt.
3. Polling - Software responsibilities in the polling mode are identical to the responsibilities in the interrupt driven mode. The polling mode, however, is used when interrupt service overhead (context switching) is too large to support the disk data

# APPLICATIONS

---

rate. In this mode, the software determines when to transfer data by continually polling a data request status flag in the FDC status register.

The DMA mode has the advantage of permitting the processor to continue executing instructions while a disk transfer is in progress. (This capability is especially useful in multiprogramming environments when the operating system is designed to permit other tasks to execute while a program is waiting for I/O.) Modes 2 and 3 are often combined and described as non-DMA operating modes. Non-DMA modes have the advantage of significantly lower system cost, but are often performance limited for double-density systems (where data bytes must be transferred to/from the FDC every 16 microseconds).

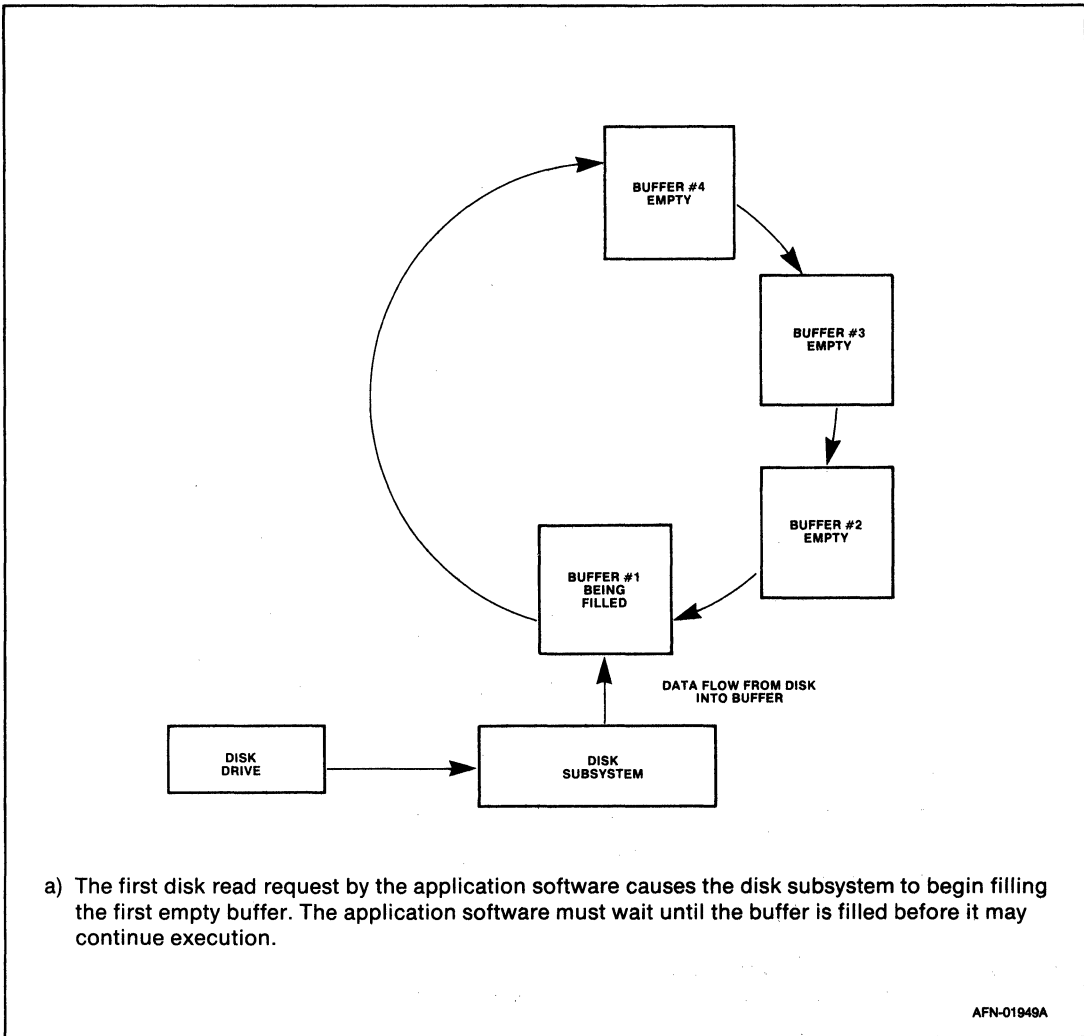
## Overlapped Operations

Some FDC devices support simultaneous disk operations on more than one disk drive. Normally seek and recalibrate operations can be overlapped in this manner. Since seek operations on most floppy drives are extremely slow, this mode of operation can often be used by the system software to reduce overall disk access times.

## Buffers

The buffer concept is an extremely important element in advanced disk I/O strategies. A buffer is nothing more than a memory area containing the same amount of data as a disk sector contains. Generally, when an application program requests data from a disk, the system software allocates a buffer (memory area) and transfers the data from the appropriate disk sector into the buffer. The address of the buffer is then returned to the application software. In the same manner, after the application program has filled a buffer for output, the buffer address is passed to the system software, which writes data from the buffer into a disk sector. In multitasking systems, multiple buffers may be allocated from a buffer pool. In these systems, the disk controller is often requested to read ahead and fill additional data buffers while the application software is processing a previous buffer. Using this technique, system software attempts to fill buffers before they are needed by the application programs, thereby eliminating program waits during I/O transfers. Figure 1 illustrates the use of multiple buffers in a ring configuration.

# APPLICATIONS

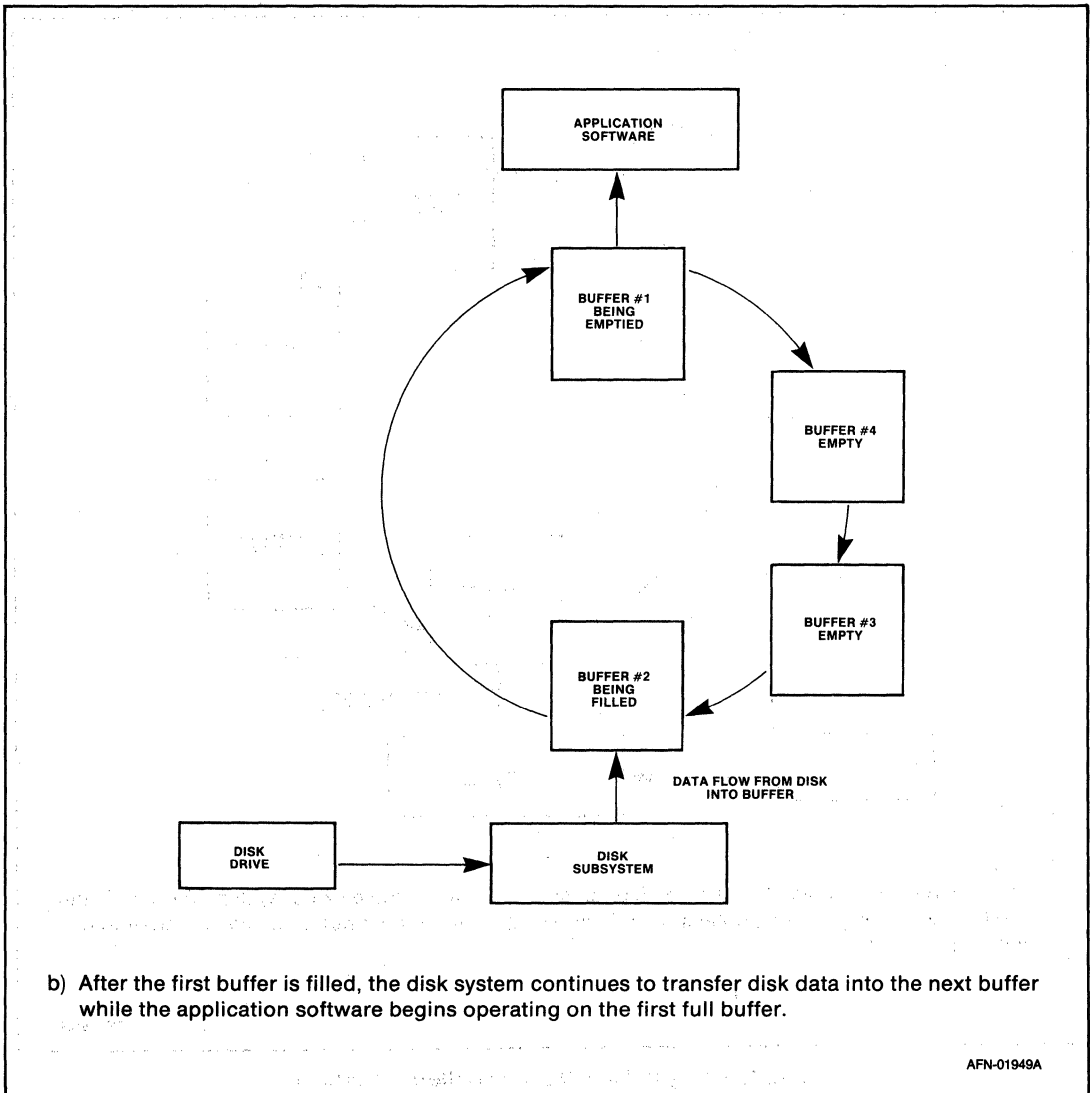


- a) The first disk read request by the application software causes the disk subsystem to begin filling the first empty buffer. The application software must wait until the buffer is filled before it may continue execution.

AFN-01949A

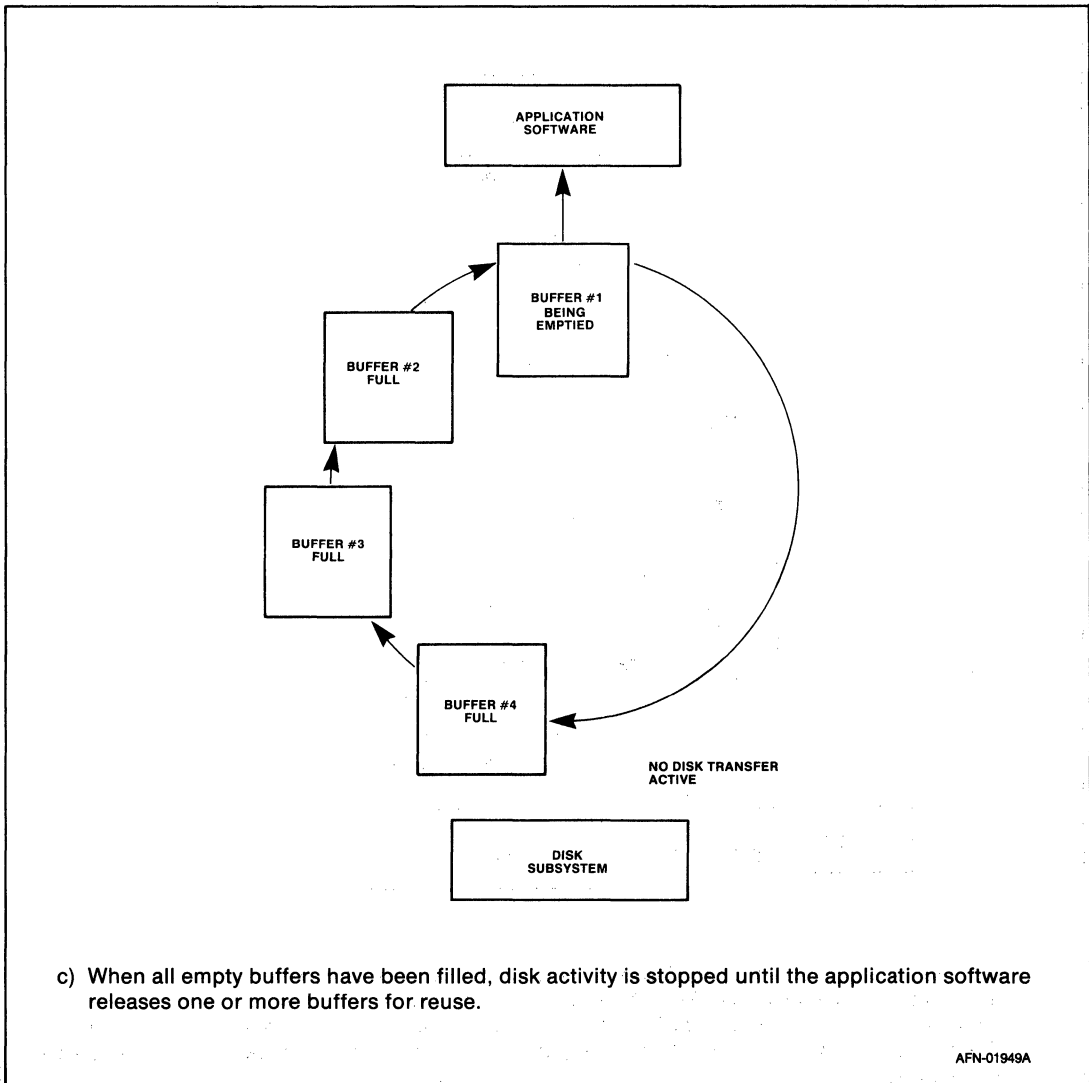
**Figure 1. Using Multiple Memory Buffers for Disk I/O**



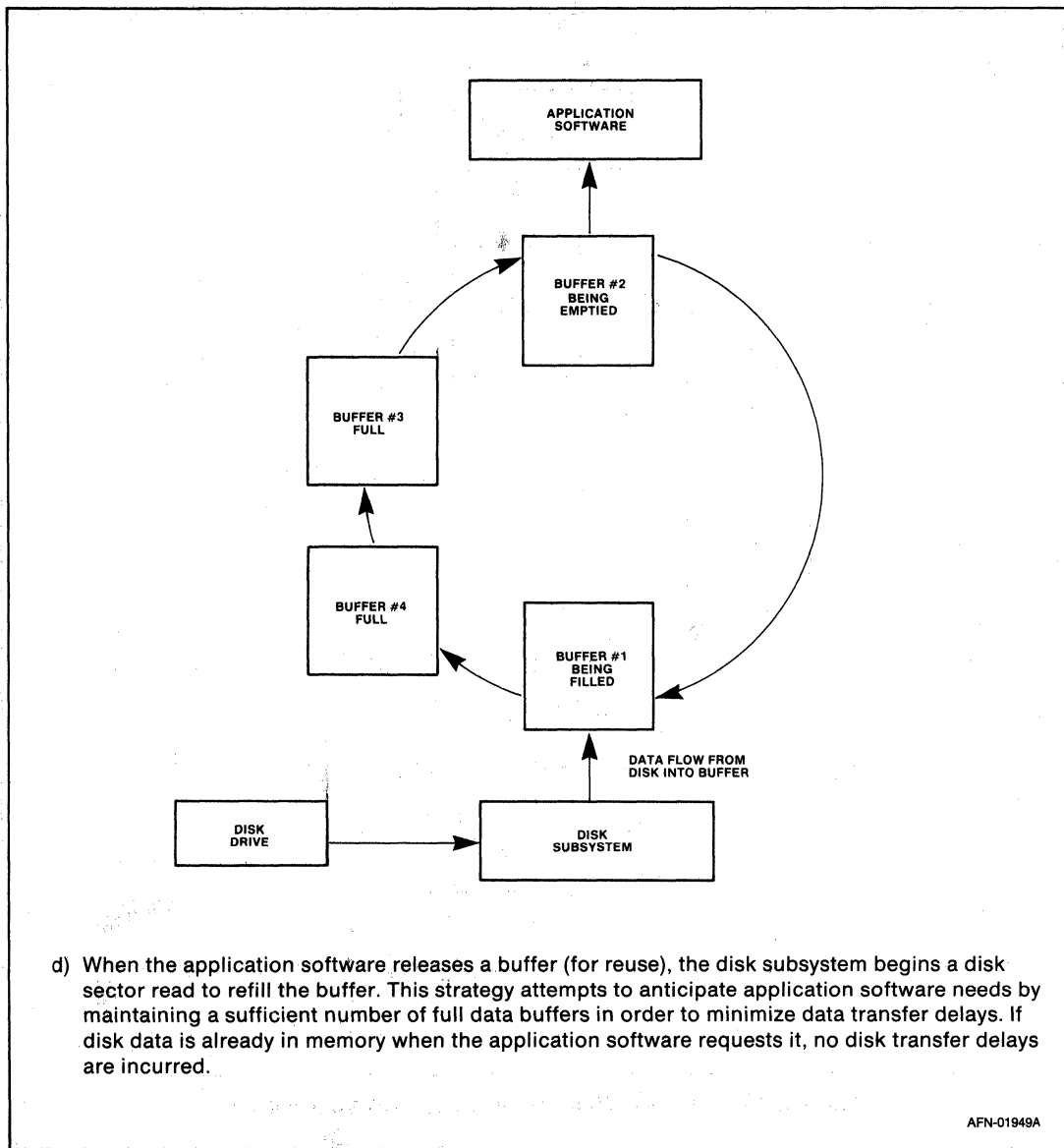


**Figure 1. Using Multiple Memory Buffers for Disk I/O (Continued)**

# APPLICATIONS



**Figure 1. Using Multiple Memory Buffers for Disk I/O (Continued)**



**Figure 1. Using Multiple Memory Buffers for Disk I/O (Continued)**

# APPLICATIONS

---

## 3. THE 8272 FLOPPY DISK CONTROLLER

The 8272 is a single-chip LSI Floppy Disk Controller (FDC) that implements both single- and double-density floppy disk storage subsystems (with up to four dual-sided disk drives per FDC). The 8272 supports the IBM 3740 single-density recording format (FM) and the IBM System 34 double-density recording format (MFM). The 8272 accepts and executes high-level disk commands such as format track, seek, read sector, and write sector. All data synchronization and error checking is automatically performed by the FDC to ensure reliable data storage and subsequent retrieval. The 8272 interfaces to microprocessor systems with or without Direct Memory Access (DMA) capabilities and also interfaces to a large number of commercially available floppy disk drives.

### Floppy Disk Commands

The 8272 executes fifteen high-level disk interface commands:

Specify	Write Data
Sense Drive Status	Write Deleted Data
Sense Interrupt Status	Read Track
Seek	Read ID
Recalibrate	Scan Equal
Format Track	Scan High or Equal
Read Data	Scan Low or Equal
Read Deleted Data	

Each command is initiated by a multi-byte transfer from the driver software to the FDC (the transferred bytes contain command and parameter information). After complete command specification, the FDC automatically executes the command. The command result data (after execution of the command) may require a multi-byte transfer of status information back to the driver. It is convenient to consider each FDC command as consisting of the following three phases:

- Command Phase:** The driver transfers to the FDC all the information required to perform a particular disk operation. The 8272 automatically enters the command phase after RESET and following the completion of the result phase (if any) of a previous command.
- Execution Phase:** The FDC performs the operation as instructed. The execution phase is entered immediately after the last command parameter is written to the FDC in the preceding command phase. The execution phase normally ends when the last data byte is transferred to/from the disk or when an error occurs.
- Result Phase:** After completion of the disk operation, status and other housekeeping information are made available to the driver software. After this information is read, the FDC reenters the command phase and is ready to accept another command.

# APPLICATIONS

---

## Interface Registers

To support information transfer between the FDC and the system software, the 8272 contains two 8-bit registers: the Main Status Register and the Data Register. The Main Status Register (read only) contains FDC status information and may be accessed at any time. The Main Status Register (Table 3) provides the system processor with the status of each disk drive, the status of the FDC, and the status of the processor interface. The Data Register (read/write) stores data, commands, parameters, and disk drive status information. The Data Register is used to program the FDC during the command phase and to obtain result information after completion of FDC operations.

In addition to the Main Status Register, the FDC contains four additional status registers (ST0, ST1, ST2, and ST3). These registers are only available during the result phase of a command.

## Command/Result Phases

Table 4 lists the 8272 command set. For each of the fifteen commands, command and result phase data transfers are listed. A list of abbreviations used in the table is given in Table 5, and the contents of the result status registers (ST0-ST3) are illustrated in Table 6.

The bytes of data which are sent to the 8272 by the drivers during the command phase, and are read out of the 8272 in the result phase, must occur in the order shown in Table 4. That is, the command code must be sent first and the other bytes sent in the prescribed sequence. All bytes of the command and result phases must be read/written as described. After the last byte of data in the command phase is sent to the 8272 the execution phase automatically starts. In a similar fashion, when the last byte of data is read from the 8272 in the result phase, the result phase is automatically ended and the 8272 reenters the command phase.

It is important to note that during the result phase all bytes shown in Table 4 must be read. The Read Data command, for example, has seven bytes of data in the result phase. All seven bytes must be read in order to successfully complete the Read Data command. The 8272 will not accept a new command until all seven bytes have been read. The number of command and result bytes varies from command-to-command.

In order to read data from, or write data to, the Data Register during the command and result phases, the software driver must examine the Main Status Register to determine if the Data Register is available. The DIO (bit 6) and RQM (bit 7) flags in the Main Status Register must be low and high, respectively, before each byte of the command word may be written into the 8272. Many of the commands require multiple bytes, and as a result, the Main Status Register must be read prior to each byte transfer to the 8272. To read status bytes during the result phase, DIO and RQM in the Main Status Register must both be high. Note, checking the Main Status Register in this manner before each byte transfer to/from the 8272 is required only in the command and result phases, and is NOT required during the execution phase.

# APPLICATIONS

---

**Table 3: Main Status Register Bit Definitions**

BIT NUMBER	SYMBOL	DESCRIPTION
0	D <sub>0</sub> B	Disk Drive 0 Busy. Disk Drive 0 is seeking.
1	D <sub>1</sub> B	Disk Drive 1 Busy. Disk Drive 1 is seeking.
2	D <sub>2</sub> B	Disk Drive 2 Busy. Disk Drive 2 is seeking.
3	D <sub>3</sub> B	Disk Drive 3 Busy. Disk Drive 3 is seeking.
4	CB	FDC Busy. A read or write command is in progress.
5	NDM	Non-DMA Mode. The FDC is in the non-DMA mode when this flag is set (1). This flag is set only during the execution phase of commands in the non-DMA mode. Transition of this flag to a zero (0) indicates that the execution phase has ended.
6	DIO	Data Input/Output. Indicates the direction of a data transfer between the FDC and the Data Register. When DIO is set (1), data is read from the Data Register by the processor; when DIO is reset (0), data is written from the processor to the Data Register.
7	RQM	Request for Master. When set (1), this flag indicates that the Data Register is ready to send data to, or receive data from, the processor.

# APPLICATIONS

## Table 4: 8272 Command Set

PHASE	R/W	DATA BUS								REMARKS	PHASE	R/W	DATA BUS								REMARKS	
		D <sub>7</sub>	D <sub>6</sub>	D <sub>5</sub>	D <sub>4</sub>	D <sub>3</sub>	D <sub>2</sub>	D <sub>1</sub>	D <sub>0</sub>				D <sub>7</sub>	D <sub>6</sub>	D <sub>5</sub>	D <sub>4</sub>	D <sub>3</sub>	D <sub>2</sub>	D <sub>1</sub>	D <sub>0</sub>		
READ DATA																						
Command	W	MT	MFM	SK	0	0	1	1	0	Command Codes	Command	W	0	MFM	SK	0	0	0	1	0	Command Codes	
	W	0	0	0	0	0	HDS	DS1	DS0		W	0	0	0	0	0	HDS	DS1	DS0			
	W									Sector ID information prior to Command execution		W	C								Sector ID information prior to Command execution	
	W											W	H									
	W											W	R									
	W											W	N									
	W											W	EOT									
	W											W	GPL									
	W											W	DTL									
Execution										Data transfer between the FDD and the main-system	Execution										Data transfer between the FDD and the main-system. FDC reads the complete track contents from the physical index mark to EOT	
Result	R									Status information after Command execution	Result	R									Status information after Command execution	
	R											R	ST 0									
	R											R	ST 1									
	R											R	ST 2									
	R											R	C									
	R											R	H									
	R											R	R									
	R											R	N									
READ DELETED DATA																						
Command	W	MT	MFM	SK	0	1	1	0	0	Command Codes	Command	W	0	MFM	0	0	1	0	1	0	Command Codes	
	W	0	0	0	0	0	HDS	DS1	DS0		W	0	0	0	0	0	HDS	DS1	DS0			
	W									Sector ID information prior to Command execution		W	C								Sector ID information prior to Command execution	
	W											W	H									
	W											W	R									
	W											W	N									
	W											W	EC 1									
	W											W	GPL									
	W											W	DTL									
Execution										Data transfer between the FDD and the main-system	Execution										The first correct ID information on the track is stored in Data Register	
Result	R									Status information after Command execution	Result	R									Status information after Command execution	
	R											R	ST 0									
	R											R	ST 1									
	R											R	ST 2									
	R											R	C									
	R											R	H									
	R											R	R									
	R											R	N									
READ ID																						
Command	W	0	MFM	0	0	1	0	1	0	Command Codes	Command	W	0	MFM	0	0	0	1	0	1	0	Command Codes
	W	0	0	0	0	0	HDS	DS1	DS0		W	0	0	0	0	0	HDS	DS1	DS0			
	W											W	C									
	W											W	H									
	W											W	R									
	W											W	N									
Execution											Execution											
Result	R									Status information after Command execution	Result	R									Status information after Command execution	
	R											R	ST 0									
	R											R	ST 1									
	R											R	ST 2									
	R											R	C									
	R											R	H									
	R											R	R									
	R											R	N									
WRITE DATA																						
Command	W	MT	MFM	0	0	0	1	0	1	Command Codes	Command	W	0	MFM	0	0	1	1	0	1	Command Codes	
	W	0	0	0	0	0	HDS	DS1	DS0		W	0	0	0	0	0	HDS	DS1	DS0			
	W									Sector ID information prior to Command execution		W	N								Bytes/Sector Sectors/Track Gap 3 Filter Byte	
	W											W	SC									
	W											W	GPL									
	W											W	D									
	W											W	EOT									
	W											W	GPL									
	W											W	DTL									
Execution										Data transfer between the main-system and the FDD	Execution										FDC formats an entire track	
Result	R									Status information after Command execution	Result	R									Status information after Command execution	
	R											R	ST 0									
	R											R	ST 1									
	R											R	ST 2									
	R											R	C									
	R											R	H									
	R											R	R									
	R											R	N									
WRITE DELETED DATA																						
Command	W	MT	MFM	0	0	1	0	0	1	Command Codes	Command	W	MT	MFM	SK	1	0	0	0	1	Command Codes	
	W	0	0	0	0	0	HDS	DS1	DS0		W	0	0	0	0	0	HDS	DS1	DS0			
	W									Sector ID information prior to Command execution		W	C								Sector ID information prior to Command execution	
	W											W	H									
	W											W	R									
	W											W	N									
	W											W	EOT									
	W											W	GPL									
	W											W	DTL									
Execution										Data transfer between the FDD and the main-system	Execution										Data compared between the FDD and the main-system	
Result	R									Status information after Command execution	Result	R									Status information after Command execution	
	R											R	ST 0									
	R											R	ST 1									
	R											R	ST 2									
	R											R	C									
	R											R	H									
	R											R	R									
	R											R	N									
FORMAT A TRACK																						
Command	W	0	MFM	0	0	1	1	0	1	Command Codes	Command	W	0	MFM	0	0	1	1	0	1	Command Codes	
	W	0	0	0	0	0	HDS	DS1	DS0		W	0	0	0	0	0	HDS	DS1	DS0			
	W											W	N									
	W											W	SC									
	W											W	GPL									
	W											W	D									
Execution											Execution										FDC formats an entire track	
Result	R									Status information after Command execution	Result	R									Status information after Command execution	
	R											R	ST 0									
	R											R	ST 1									
	R											R	ST 2									
	R											R	C									
	R											R	H									
	R											R	R									
	R											R	N									
SCAN EQUAL																						
Command	W	MT	MFM	SK	1	0	0	0	1	Command Codes	Command	W	MT	MFM	SK	1	0	0	0	1	Command Codes	
	W	0	0	0	0	0	HDS	DS1	DS0		W	0	0	0	0	0	HDS	DS1	DS0			
	W									Sector ID information prior to Command execution		W	C								Sector ID information prior to Command execution	
	W											W	H									
	W											W	R									
	W											W	N									
	W											W	EOT									
	W											W	GPL									
	W											W	STP									
Execution											Execution										Data compared between the FDD and the main-system	
Result	R									Status information after Command execution	Result	R									Status information after Command execution	
	R											R	ST 0									
	R											R	ST 1									
	R											R	ST 2									
	R											R	C									
	R											R	H									
	R											R	R									
	R											R	N									

Note: 1. A<sub>0</sub> = 1 for all operations.

# APPLICATIONS

PHASE	R/W	DATA BUS								REMARKS	PHASE	R/W	DATA BUS								REMARKS
		D7	D6	D5	D4	D3	D2	D1	D0				D7	D6	D5	D4	D3	D2	D1	D0	
<b>SCAN LOW OR EQUAL</b>																					
Command	W	MT	MFM	SK	1	1	0	0	1	Command Codes	W	0	0	0	0	0	1	1	1	Command Codes	
	W	0	0	0	0	0	HDS	DS1	DS0			W	0	0	0	0	0	0	DS1		DS0
Execution	W									Sector ID information prior Command execution	W									Head retracted to Track 0	
	W											C									
	W											H									
	W											R									
	W											N									
	W											EOT									
Result	R									Status information after Command execution	W									Status information at the end of each seek operation about the FDC	
	R											ST 0									
Execution	W									Data compared between the FDD and the main-system	R									Status information after Command execution	
	W											ST 1									
	W											ST 2									
	W											C									
	W											H									
	W											R									
Result	R									Sector ID information after Command execution	R									Sector ID information after Command execution	
	R											N									
<b>SCAN HIGH OR EQUAL</b>																					
Command	W	MT	MFM	SK	1	1	1	0	1	Command Codes	W	0	0	0	0	0	HDS	DS1	DS0	Command Codes	
	W	0	0	0	0	0	HDS	DS1	DS0			W	0	0	0	0	0	HDS	DS1		DS0
Execution	W									Sector ID information prior Command execution	W									Head is positioned over proper Cylinder on Diskette	
	W											C									
	W											H									
	W											R									
	W											N									
	W											EOT									
Result	R									Status information after Command execution	W									Invalid Command Codes (NoOp — FDC goes into Standby State) ST 0 = 80 (16)	
	R											ST 0									
Execution	W									Data compared between the FDD and the main-system	R									Sector ID information after Command execution	
	W											ST 1									
	W											ST 2									
	W											C									
	W											H									
	W											R									
Result	R									Sector ID information after Command execution	R									Sector ID information after Command execution	
	R											N									
<b>RECALIBRATE</b>																					
Command	W	0	0	0	0	0	0	1	1	1	Command Codes	W	0	0	0	0	0	0	DS1	DS0	Command Codes
	W	0	0	0	0	0	0	0	0	0			DS1	DS0	Head retracted to Track 0						
Execution	W									Sector ID information prior Command execution	W									Head retracted to Track 0	
	W											C									
	W											H									
	W											R									
	W											N									
	W											EOT									
Result	R									Status information after Command execution	R									Status information at the end of each seek operation about the FDC	
	R											ST 0									
Execution	W									Data compared between the FDD and the main-system	R									Sector ID information after Command execution	
	W											ST 1									
	W											ST 2									
	W											C									
	W											H									
	W											R									
Result	R									Sector ID information after Command execution	R									Sector ID information after Command execution	
	R											N									
<b>SENSE INTERRUPT STATUS</b>																					
Command	W	0	0	0	0	1	0	0	0	Command Codes	W	0	0	0	0	1	0	0	0	Command Codes	
	W	0	0	0	0	0	0	0	0			DS1	DS0	Status information at the end of each seek operation about the FDC							
Execution	W									Sector ID information prior Command execution	W									Head is positioned over proper Cylinder on Diskette	
	W											C									
	W											H									
	W											R									
	W											N									
	W											EOT									
Result	R									Status information after Command execution	R									Sector ID information after Command execution	
	R											N									
<b>SPECIFY</b>																					
Command	W	0	0	0	0	0	0	0	1	1	Command Codes	W	0	0	0	0	0	1	1	Command Codes	
	W	0	0	0	0	0	0	0	0	0			1	1	Timer Settings						
Execution	W									Sector ID information prior Command execution	W									Head is positioned over proper Cylinder on Diskette	
	W											SPT	HUT	ND							
	W											HLT	ND								
	W											H									
	W											R									
	W											N									
Result	R									Status information after Command execution	R									Sector ID information after Command execution	
	R											N									
<b>SENSE DRIVE STATUS</b>																					
Command	W	0	0	0	0	0	1	0	0	Command Codes	W	0	0	0	0	0	HDS	DS1	DS0	Command Codes	
	W	0	0	0	0	0	0	HDS	DS1			DS0	W	0	0	0	0	0	HDS		DS1
Execution	W									Sector ID information prior Command execution	W									Head is positioned over proper Cylinder on Diskette	
	W											C									
	W											H									
	W											R									
	W											N									
	W											EOT									
Result	R									Status information after Command execution	R									Sector ID information after Command execution	
	R											N									
<b>SEEK</b>																					
Command	W	0	0	0	0	1	1	1	1	Command Codes	W	0	0	0	0	0	HDS	DS1	DS0	Command Codes	
	W	0	0	0	0	0	0	HDS	DS1			DS0	W	0	0	0	0	0	HDS		DS1
Execution	W									Sector ID information prior Command execution	W									Head is positioned over proper Cylinder on Diskette	
	W											C									
	W											H									
	W											R									
	W											N									
	W											EOT									
Result	R									Status information after Command execution	R									Sector ID information after Command execution	
	R											N									
<b>INVALID</b>																					
Command	W									Invalid Codes	W									Invalid Command Codes (NoOp — FDC goes into Standby State) ST 0 = 80 (16)	
	W											Invalid Codes									
Execution	W									Sector ID information prior Command execution	W									Head is positioned over proper Cylinder on Diskette	
	W											C									
	W											H									
	W											R									
	W											N									
	W											EOT									
Result	R									Status information after Command execution	R									Sector ID information after Command execution	
	R											N									



# APPLICATIONS

**Table 5: Command/Result Parameter Abbreviations**

SYMBOL	DESCRIPTION															
C	Cylinder Address. The currently selected cylinder address (0 to 76) on the disk.															
D	Data Pattern. The pattern to be written in each sector data field during formatting.															
DS0,DS1	Disk Drive Select.  <table style="margin-left: 40px;"> <tr> <td>DS1</td> <td>DS0</td> <td></td> </tr> <tr> <td>0</td> <td>0</td> <td>Drive 0</td> </tr> <tr> <td>0</td> <td>1</td> <td>Drive 1</td> </tr> <tr> <td>1</td> <td>0</td> <td>Drive 2</td> </tr> <tr> <td>1</td> <td>1</td> <td>Drive 3</td> </tr> </table>	DS1	DS0		0	0	Drive 0	0	1	Drive 1	1	0	Drive 2	1	1	Drive 3
DS1	DS0															
0	0	Drive 0														
0	1	Drive 1														
1	0	Drive 2														
1	1	Drive 3														
DTL	Special Sector Size. During the execution of disk read/write commands, this parameter is used to temporarily alter the effective disk sector size. By setting N to zero, DTL may be used to specify a sector size from 1 to 256 bytes in length. If the actual sector (on the disk) is larger than DTL specifies, the remainder of the actual sector is not passed to the system during read commands; during write commands, the remainder of the actual sector is written with all-zeroes bytes. DTL should be set to FF hexadecimal when N is not zero.															
EOT	End of Track. The final sector number of the current track.															
GPL	Gap Length. The gap 3 size. (Gap 3 is the space between sectors.)															
H	Head Address. Selected head: 0 or 1 (disk side 0 or 1, respectively) as encoded in the sector ID field.															
HLT	Head Load Time. Defines the time interval that the FDC waits after loading the head before initiating a read or write operation. Programmable from 2 to 254 milliseconds (in increments of 2 ms).															
HUT	Head Unload Time. Defines the time interval from the end of the execution phase (of a read or write command) until the head is unloaded. Programmable from 16 to 240 milliseconds (in increments of 16 ms).															
MFM	MFM/FM Mode Selector. Selects MFM double-density recording mode when high, FM single-density mode when low.															
MT	Multi-Track Selector. When set, this flag selects the multi-track operating mode. In this mode (used only with dual-sided disks), the FDC treats a complete cylinder (under both read/write head 0 and read/write head 1) as a single track. The FDC operates as if this expanded track started at the first sector under head 0 and ended at the last sector under head 1. With this flag set (high), a multi-sector read operation will automatically continue to the first sector under head 1 when the FDC finishes operating on the last sector under head 0.															
N	Sector Size Code. The number of data bytes within a sector.															

## APPLICATIONS

ND	Non-DMA Mode Flag. When set (1), this flag indicates that the FDC is to operate in the non-DMA mode. In this mode, the processor participates in each data transfer (by means of an interrupt or by polling the RQM flag in the Main Status Register). When reset (0), the FDC interfaces to a DMA controller.
R	Sector Address. Specifies the sector number to be read or written. In multi-sector transfers, this parameter specifies the sector number of the first sector to be read or written.
SC	Number of Sectors per Track. Specifies the number of sectors per track to be initialized by the Format Track command.
SK	Skip Flag. When this flag is set, sectors containing deleted data address marks will automatically be skipped during the execution of multi-sector Read Data or Scan commands. In the same manner, a sector containing a data address mark will automatically be skipped during the execution of a multi-sector Read Deleted Data command.
SRT	Step Rate Interval. Defines the time interval between step pulses issued by the FDC (track-to-track access time). Programmable from 1 to 16 milliseconds (in increments of 1 ms).
ST0 ST1 ST2 ST3	Status Register 0-3. Registers within the FDC that store status information after a command has been executed. This status information is available to the processor during the Result Phase after command execution. These registers may only be read after a command has been executed (in the exact order shown in Table 4 for each command). These registers should not be confused with the Main Status Register.
STP	Scan Sector Increment. During Scan operations, this parameter is added to the current sector number in order to determine the next sector to be scanned.

# APPLICATIONS

**Table 6: Status Register Definitions**

Status Register 0		
BIT NUMBER	SYMBOL	DESCRIPTION
7,6	IC	<p>Interrupt Code.</p> <p>00 - Normal termination of command. The specified command was properly executed and completed without error.</p> <p>01 - Abnormal termination of command. Command execution was started but could not be successfully completed.</p> <p>10 - Invalid command. The requested command could not be executed.</p> <p>11 - Abnormal termination. During command execution, the disk drive ready signal changed state.</p>
5	SE	Seek End. This flag is set (1) when the FDC has completed the Seek command and the read/write head is positioned over the correct cylinder.
4	EC	Equipment Check Error. This flag is set (1) if a fault signal is received from the disk drive or if the track 0 signal is not received from the disk drive after 77 step pulses (Recalibrate command).
3	NR	Not Ready Error. This flag is set if a read or write command is issued and either the drive is not ready or the command specifies side 1 (head 1) of a single-sided disk.
2	H	Head Address. The head address at the time of the interrupt.
1,0	DSL,DS0	Drive Select. The number of the drive selected at the time of the interrupt.
Status Register 1		
BIT NUMBER	SYMBOL	DESCRIPTION
7	EN	End of Track Error. This flag is set if the FDC attempts to access a sector beyond the final sector of the track.
6		Undefined
5	DE	Data Error. Set when the FDC detects a CRC error in either the ID field or the data field of a sector.
4	OR	Overrun Error. Set (during data transfers) if the FDC does not receive DMA or processor service within the specified time interval.

# APPLICATIONS

3		Undefined
2	ND	<p>Sector Not Found Error. This flag is set by any of the following conditions.</p> <p>a) The FDC cannot locate the sector specified in the Read Data, Read Deleted Data, or Scan command.</p> <p>b) The FDC cannot locate the starting sector specified in the Read Track command.</p> <p>c) The FDC cannot read the ID field without error during a Read ID command.</p>
1	NW	Write Protect Error. This flag is set if the FDC detects a write protect signal from the disk drive during the execution of a Write Data, Write Deleted Data, or Format Track command.
0	MA	<p>Missing Address Mark Error. This flag is set by either of the following conditions:</p> <p>a) The FDC cannot detect the ID address mark on the specified track (after two rotations of the disk).</p> <p>b) The FDC cannot detect the data address mark or deleted data address mark on the specified track. (See also the MD bit of Status Register 2.)</p>

## Status Register 2

BIT NUMBER	SYMBOL	DESCRIPTION
7		Undefined
6	CM	<p>Control Mark. This flag is set when the FDC encounters one of the following conditions:</p> <p>a) A deleted data address mark during the execution of a Read Data or Scan command.</p> <p>b) A data address mark during the execution of a Read Deleted Data command.</p>
5	DD	Data Error. Set (1) when the FDC detects a CRC error in a sector data field. This flag is not set when a CRC error is detected in the ID field.
4	WC	Cylinder Address Error. Set when the cylinder address from the disk sector ID field is different from the current cylinder address maintained within the FDC.
3	SH	Scan Hit. Set during the execution of the Scan command if the scan condition is satisfied.
2	SN	Scan Not Satisfied. Set during execution of the Scan command if the FDC cannot locate a sector on the specified cylinder that satisfies the scan condition.

# APPLICATIONS

1	BC	Bad Track Error. Set when the cylinder address from the disk sector ID field is FF hexadecimal and this cylinder address is different from the current cylinder address maintained within the FDC. This all "ones" cylinder number indicates a bad track (one containing hard errors) according to the IBM soft-sectored format specifications.
0	MD	Missing Data Address Mark Error. Set if the FDC cannot detect a data address mark or deleted data address mark on the specified track.
<b>Status Register 3</b>		
BIT NUMBER	SYMBOL	DESCRIPTION
7	FT	Fault. This flag indicates the status of the fault signal from the selected disk drive.
6	WP	Write Protected. This flag indicates the status of the write protect signal from the selected disk drive.
5	RDY	Ready. This flag indicates the status of the ready signal from the selected disk drive.
4	T0	Track 0. This flag indicates the status of the track 0 signal from the selected disk drive.
3	TS	Two-Sided. This flag indicates the status of the two-sided signal from the selected disk drive.
2	H	Head Address. This flag indicates the status of the side select signal for the currently selected disk drive.
1,0	DS1,DS0	Drive Select. Indicates the currently selected disk drive number.

# APPLICATIONS

---

## Execution Phase

All data transfers to (or from) the floppy drive occur during the execution phase. The 8272 has two primary modes of operation for data transfers (selected by the specify command):

- 1) DMA mode
- 2) non-DMA mode

In the DMA mode, execution phase data transfers are handled by the DMA controller hardware (invisible to the driver software). The driver software, however, must set all appropriate DMA controller registers prior to the beginning of the disk operation. An interrupt is generated by the 8272 after the last data transfer, indicating the completion of the execution phase, and the beginning of the result phase.

In the non-DMA mode, transfer requests are indicated by generation of an interrupt and by activation of the RQM flag (bit 7 in the Main Status Register). The interrupt signal can be used for interrupt-driven systems and RQM can be used for polled systems. The driver software must respond to the transfer request by reading data from, or writing data to, the FDC. After completing the last transfer, the 8272 generates an interrupt to indicate the beginning of the result phase. In the non-DMA mode, the processor must activate the "terminal count" (TC) signal to the FDC (normally by means of an I/O port) after the transfer request for the last data byte has been received (by the driver) and before the appropriate data byte has been read from (or written to) the FDC.

In either mode of operation (DMA or non-DMA), the execution phase ends when a "terminal count" signal is sensed by the FDC, when the last sector on a track (the EOT parameter - Table 4) has been read or written, or when an error occurs.

## Multi-sector and Multi-track Transfers

During disk read/write transfers (Read Data, Write Data, Read Deleted Data, and Write Deleted Data), the FDC will continue to transfer data from sequential sectors until the TC input is sensed. In the DMA mode, the TC input is normally set by the DMA controller. In the non-DMA mode, the processor directly controls the FDC TC input as previously described. Once the TC input is received, the FDC stops requesting data transfers (from the system software or DMA controller). The FDC, however, continues to read data from, or write data to, the floppy disk until the end of the current disk sector. During a disk read operation, the data read from the disk (after reception of the TC input) is discarded, but the data CRC is checked for errors; during a disk write operation, the remainder of the sector is filled with all-zero bytes.

If the TC signal is not received before the last byte of the current sector has been transferred to/from the system, the FDC increments the sector number by one and initiates a read or write command for this new disk sector.

# APPLICATIONS

---

The FDC is also designed to operate in a multi-track mode for dual-sided disks. In the multi-track mode (specified by means of the MT flag in the command byte - Table 4) the FDC will automatically increment the head address (from 0 to 1) when the last sector (on the track under head 0) has been read or written. Reading or writing is then continued on the first sector (sector 1) of head 1.

## Drive Status Polling

After the power-on reset, the 8272 automatically enters a drive status polling mode. If a change in drive status is detected (all drives are assumed to be "not ready" at power-on), an interrupt is generated. The 8272 continues this status polling between command executions (and between step pulses in the Seek command). In this manner, the 8272 automatically notifies the system software whenever a floppy disk is inserted, removed, or changed by the operator.

## Command Details

During the command phase, the Main Status Register must be polled by the driver software before each byte is written into the Data Register. The DIO (bit 6) and RQM (bit 7) flags in the Main Status Register must be low and high, respectively, before each byte of the command may be written into the 8272. The beginning of the execution phase for any of these commands will cause DIO to be set high and RQM to be set low.

Operation of the FDC commands is described in detail in Application Note AP-116, "An Intelligent Data Base System Using the 8272."

## Invalid Commands

If an invalid (undefined) command is sent to the FDC, the FDC will terminate the command. No interrupt is generated by the 8272 during this condition. Bit 6 and bit 7 (DIO and RQM) in the Main Status Register are both set indicating to the processor that the 8272 is in the result phase and the contents of Status Register 0 must be read. When the processor reads Status Register 0 it will find an 80H code indicating that an invalid command was received. The driver software in Appendix B checks each requested command and will not issue an invalid command to the 8272.

A Sense Interrupt Status command must be sent after a Seek or Recalibrate interrupt; otherwise the FDC will consider the next command to be an invalid command. Also, when the last "hidden" interrupt has been serviced, further Sense Interrupt Status commands will result in invalid command codes.

# APPLICATIONS

---

## 4. 8272 Physical Interface Software

PL/M software driver listings for the 8272 FDC are contained in Appendix A. These drivers have been designed to operate in a DMA environment (as described in Application Note AP-116, "An Intelligent Data Base System Using the 8272"). In the following paragraphs, each driver procedure is described. (A description of the driver data base variables is given in Table 7.) In addition, the modifications necessary to reconfigure the drivers for operation in a polled environment are discussed.

### INITIALIZE\$DRIVERS

This initialization procedure must be called before any FDC operations are attempted. This module initializes the DRIVE\$READY, DRIVE\$STATUS\$CHANGE, OPERATION\$IN\$PROGRESS, and OPERATION\$COMPLETE arrays as well as the GLOBAL\$DRIVE\$NO variable.

### EXECUTE\$DOCB

This procedure contains the main 8272 driver control software and handles the execution of a complete FDC command. EXECUTE\$DOCB is called with two parameters: a) a pointer to a disk operation control block and b) a pointer to a result status byte. The format of the disk operation control block is illustrated in Figure 2 and the result status codes are described in Table 8.

Before starting the command phase for the specified disk operation, the command is checked for validity and to determine whether the FDC is busy. (For an overlapped operation, if the FDC BUSY flag is set - in the Main Status Register - the command cannot be started; non-overlapped operations cannot be started if the FDC BUSY flag is set, if any drive is in the process of seeking/recalibrating, or if an operation is currently in progress on the specified drive.)

After these checks are made, interrupts are disabled in order to set the OPERATION\$IN\$PROGRESS flag, reset the OPERATION\$COMPLETE flag, load a pointer to the current operation control block into the OPERATION\$DOCB\$PTR array and set GLOBAL\$DRIVE\$NO (if a non-overlapped operation is to be started).

At this point, parameters from the operation control block are output to the DMA controller and the FDC command phase is initiated. After completion of the command phase, a test is made to determine the type of result phase required for the current operation. If no result phase is needed, control is immediately returned to the calling program. If an immediate result phase is required, the result bytes are input from the FDC. Otherwise, the CPU waits until the OPERATION\$COMPLETE flag is set (by the interrupt service procedure).

Finally, if an error is detected in the result status code (from the FDC), an FDC operation error is reported to the calling program.



# APPLICATIONS

**Table 7: Driver Data Base**

NAME	DESCRIPTION
DRIVE\$READY	A public array containing the current "ready" status of each drive.
DRIVE\$STATUS\$CHANGE	A public array containing a flag for each drive. The appropriate flag is set whenever the ready status of a drive changes.
OPERATION\$DOCB\$PTR	An internal array of pointers to the operation control block currently in progress for each drive.
OPERATION\$IN\$PROGRESS	An internal array used by the driver procedures to determine if a disk operation is in progress on a given drive.
OPERATION\$COMPLETE	An internal array used by the driver procedures to determine when the execution phase of a disk operation is complete.
GLOBAL\$DRIVE\$NO	A data byte that records the current drive number for non-overlapped disk operations.
VALID\$COMMAND	A constant flag array that indicates whether a specified FDC command code is valid.
COMMAND\$LENGTH	A constant byte array specifying the number of command/parameter bytes to be transferred to the FDC during the command phase.
DRIVE\$NO\$PRESENT	A constant flag array that indicates whether a drive number is encoded into an FDC command.
OVERLAP\$OPERATION	A constant flag array that indicates whether an FDC command can be overlapped with other commands.
NO\$RESULT	A constant flag array that is used to determine when an FDC operation does not have a result phase.
IMMED\$RESULT	A constant flag array that indicates that an FDC operation has a result phase beginning immediately after the command phase is complete.
POSSIBLE\$ERROR	A constant flag array that indicates if an FDC operation should be checked for an error status indication during the result phase.

# APPLICATIONS

Address Offset	Disk Operation Control Block (DOCB)
0	DMA\$OP
1	DMA\$ADDR
3	DMA\$ADDR\$EXT
4	DMA\$COUNT
6	DISK\$COMMAND (0)
7	DISK\$COMMAND (1)
8	DISK\$COMMAND (2)
9	DISK\$COMMAND (3)
10	DISK\$COMMAND (4)
11	DISK\$COMMAND (5)
12	DISK\$COMMAND (6)
13	DISK\$COMMAND (7)
14	DISK\$COMMAND (8)
15	DISK\$RESULT (0)
16	DISK\$RESULT (1)
17	DISK\$RESULT (2)
18	DISK\$RESULT (3)
19	DISK\$RESULT (4)
20	DISK\$RESULT (5)
21	DISK\$RESULT (6)
22	MISC

AFN-01949A

Figure 2. Disk Operation Control Block (DOCB) Format

# APPLICATIONS

**Table 8: EXECUTE\$DOCB Return Status Codes**

Code	Description
0	No errors. The specified operation was completed without error.
1	FDC busy. The requested operation cannot be started. This error occurs if an attempt is made to start an operation before the previous operation is completed.
2	FDC error. An error was detected by the FDC during the execution phase of a disk operation. Additional error information is contained in the result data portion of the disk operation control block (DOCB.DISK\$RESULT) as described in the 8272 data sheet. This error occurs whenever the 8272 reports an execution phase error (e.g., missing address mark).
3	8272 command interface error. An 8272 interfacing error was detected during the command phase. This error occurs when the command phase of a disk operation cannot be successfully completed (e.g., incorrect setting of the DIO flag in the Main Status Register).
4	8272 result interface error. An 8272 interfacing error was detected during the result phase. This error occurs when the result phase of a disk operation cannot be successfully completed (e.g., incorrect setting of the DIO flag in the Main Status Register).
5	Invalid FDC Command.

# APPLICATIONS

---

## FDCINT

This procedure performs all interrupt processing for the 8272 interface drivers. Basically, two types of interrupts are generated by the 8272: (a) an interrupt that signals the end of a command execution phase and the beginning of the result phase and (b) an interrupt that signals the completion of an overlapped operation or the occurrence of an unexpected event (e.g., change in the drive "ready" status).

An interrupt of type (a) is indicated when the FDC BUSY flag is set (in the Main Status Register). When a type (a) interrupt is sensed, the result bytes are read from the 8272 and placed in the result portion of the disk operation control block, the appropriate OPERATION\$COMPLETE flag is set, and the OPERATION\$IN\$PROGRESS flag is reset.

When an interrupt of type (b) is indicated (FDC not busy), a sense interrupt status command is issued (to the FDC). The upper two bits of the result status register (Status Register Zero - ST0) are used to determine the cause of the interrupt. The following four cases are possible:

- 1) Operation Complete. An overlapped operation is complete. The drive number is found in the lower two bits of ST0. The ST0 data is transferred to the active operation control block, the OPERATION\$COMPLETE flag is set, and the OPERATION\$IN\$PROGRESS flag is reset.
- 2) Abnormal Termination. A disk operation has abnormally terminated. The drive number is found in the lower two bits of ST0. The ST0 data is transferred to the active control block, the OPERATION\$COMPLETE flag is set, and the OPERATION\$IN\$PROGRESS flag is reset.
- 3) Invalid Command. The execution of an invalid command (i.e., a sense interrupt command with no interrupt pending) has been attempted. This interrupt signals the successful completion of all interrupt processing.
- 4) Drive Status Change. A change has occurred in the "ready" status of a disk drive. The drive number is found in the lower two bits of ST0. The DRIVES\$READY flag for this disk drive is set to the new drive "ready" status and the DRIVES\$STATUS\$CHANGE flag for the drive is also set. In addition, if a command is currently in progress, the ST0 data is transferred to the active control block, the OPERATION\$COMPLETE flag is set, and the OPERATION\$IN\$PROGRESS flag is reset.

After processing a type (b) interrupt, additional sense interrupt status commands must be issued and processed until an "invalid command" result is returned from the FDC. This action guarantees that all "hidden" interrupts are serviced.

In addition to the major driver procedures described above, a number of support procedures are required. These support routines are briefly described in the following paragraphs.

# APPLICATIONS

---

## **OUTPUT\$CONTROLS\$TO\$DMA**

This procedure outputs the DMA mode, the DMA address, and the DMA word count to the 8237 DMA controller. In addition, the upper four bits of the 20-bit DMA address are output to the address extension latch. Finally, the disk DMA channel is started.

## **OUTPUT\$COMMAND\$TO\$FDC**

This software module outputs a complete disk command to the 8272 FDC. The number of required command/parameter bytes is found in the COMMAND\$LENGTH table. The appropriate bytes are output one at a time (by calls to OUTPUT\$BYTE\$TO\$FDC) from the command portion of the disk operation control block.

## **INPUT\$RESULT\$FROM\$FDC**

This procedure is used to read result phase status information from the disk controller. At most, seven bytes are read. In order to read each byte, a call is made to INPUT\$BYTE\$FROM\$FDC. When the last byte has been read, a check is made to insure that the FDC is no longer busy.

## **OUTPUT\$BYTE\$TO\$FDC**

This software is used to output a single command/parameter byte to the FDC. This procedure waits until the FDC is ready for a command byte and then outputs the byte to the FDC data port.

## **INPUT\$BYTE\$FROM\$FDC**

This procedure inputs a single result byte from the FDC. The software waits until the FDC is ready to transfer a result byte and then reads the byte from the FDC data port.

## **FDC\$READY\$FOR\$COMMAND**

This procedure assures that the FDC is ready to accept a command/parameter byte by performing the following three steps. First, a small time interval (more than 20 microseconds) is inserted to assure that the RQM flag has time to become valid (after the last byte transfer). Second, the master request flag (RQM) is polled until it is activated by the FDC. Finally, the DIO flag is checked to ensure that it is properly set for FDC input (from the processor).

## **FDC\$READY\$FOR\$RESULT**

The operation of this procedure is similar to the FDC\$READY\$FOR\$COMMAND with the following exception. If the FDC BUSY flag (in the Main Status Register) is not set, the result phase is complete and no more data is available from the FDC. Otherwise, the procedure waits for the RQM flag and checks the DIO flag for FDC output (to the processor).

# APPLICATIONS

---

## OPERATION\$CLEAN\$UP

This procedure is called after the execution of a disk operation that has no result phase. OPERATION\$CLEAN\$UP resets the OPERATION\$IN\$PROGRESS flag and the GLOBAL\$DRIVE\$NO variable if appropriate. This procedure is also called to clean up after some disk operation errors.

## Modifications for Polling Operation

To operate in the polling mode, the following modifications should be made to the previous routines:

1. The OUTPUT\$CONTROLS\$TO\$DMA routine should be deleted.
2. In EXECUTE\$DOCB, immediately prior to WAIT\$FOR\$OP\$COMPLETE, a polling loop should be inserted into the code. The loop should test the RQM flag (in the Main Status Register). When RQM is set, a data byte should be written to, or read from, the 8272. The buffer address may be computed from the base address contained in DOCB.DMA\$ADDR and DOCB.DMA\$ADDR\$EXT. After the correct number of bytes have been transferred, an operation complete interrupt will be issued by the FDC. During data transfer in the non-DMA mode, the NON-DMA MODE flag (bit 5 of the Main Status Register) will be set. This flag will remain set for the complete execution phase. When the transfer is finished, the NON-DMA MODE flag is reset and the result phase interrupt is issued by the FDC.

# APPLICATIONS

---

## 5. 8272 Logical Interface Software

Appendix B of this Application Note contains a PL/M listing of an exerciser program for the 8272 drivers. This program illustrates the design of logical interface level procedures to specify disk parameters, recalibrate a drive, seek to a cylinder, format a disk, read data, and write data.

The exerciser program is written to operate a standard single-sided 8" floppy disk drive in either the single- or double-density recording mode. Only the eight parameters listed in Table 9 must be specified. All other parameters are derived from these 8 basic variables.

Each of these logical interface procedures is described in the following paragraphs (refer to the listing in Appendix B).

### **SPECIFY**

This procedure sets the FDC signal timing so that the FDC will interface correctly to the attached disk drive. The SPECIFY procedure requires four parameters, the step rate (SRT), head load time (HLT), head unload time (HUT), and the non-DMA mode flag (ND). This procedure builds a disk operation control block (SPECIFY\$DOCB) and passes the control block to the FDC driver module (EXECUTE\$DOCB) for execution. (Note carefully the computation required to transform the step rate (SRT) into the correct 8272 parameter byte.)

### **RECALIBRATE**

This procedure causes the floppy disk read/write head to retract to track 0. The RECALIBRATE procedure requires only one parameter - the drive number on which the recalibrate operation is to be performed. This procedure builds a disk operation control block (RECALIBRATE\$DOCB) and passes the control block to the FDC driver for execution.

### **SEEK**

This procedure causes the disk read/write head (on the selected drive) to move to the desired cylinder position. The SEEK procedure is called with three parameters: drive number (DRV), head/side number (HD), and cylinder number (CYL). This software module builds a disk operation control block (SEEK\$DOCB) that is executed by the FDC driver.

### **FORMAT**

The FORMAT procedure is designed to initialize a complete floppy disk so that sectors can subsequently be read and written by system and application programs. Three parameters must be supplied to this procedure: the drive number (DRV), the recording density (DENS), and the interleave factor (INTLVE). The FORMAT procedure generates a data block (FMTBLK) and a disk operation control block (FORMAT\$DOCB) for each track on the floppy disk (normally 77).

# APPLICATIONS

**Table 9: Basic Disk Parameters**

Name	Description
DENSITY	The recording mode (FM or MFM).
FILLER\$BYTE	The data byte to be written in all sectors during formatting.
TRACKS\$PER\$DISK	The number of cylinders on the floppy disk.
BYTES\$PER\$SECTOR	The number of bytes in each disk sector. The exerciser accepts 128, 256, and 512 in FM mode, and 256, 512, and 1024 in MFM mode.
INTERLEAVE	The sector interleave factor for each disk track.
STEP\$RATE	The disk drive step rate (1-16 milliseconds).
HEAD\$LOAD\$TIME	The disk drive head load time (2-254 milliseconds).
HEAD\$UNLOAD\$TIME	The head unload time (16-240 milliseconds).



# APPLICATIONS

---

The format data block specifies the four sector ID field parameters (cylinder, head, sector, and bytes per sector) for each sector on the track. The sector numbers need not be sequential; the interleave factor (INTLVE parameter) is used to compute the logical to physical sector mapping.

After both the format data block and the operation control block are generated for a given cylinder, control is passed to the 8272 drivers for execution. After the format operation is complete, a SEEK to the next cylinder is performed, a new format table is generated, and another track formatting operation is executed by the drivers. This track formatting continues until all tracks on the diskette are formatted.

In some systems, bad tracks must also be specified when a disk is formatted. For these systems, the existing FORMAT procedure should be modified to format bad tracks with a cylinder number of OFFH.

## WRITE

The WRITE procedure transfers a complete sector of data to the disk drive. Five parameters must be supplied to this software module: the drive number (DRV), the cylinder number (CYL), the head/side number (HD), the sector number (SEC) and the recording density (DENS). This procedure generates a disk operation control block (WRITE\$DOCB) from these parameters and passes the control block to the 8272 driver for execution. When control returns to the calling program, the data has been transferred to disk.

## READ

This procedure is identical to the WRITE procedure except the direction of data transfer is reversed. The READ procedure transfers a sector of data from the floppy disk to system memory.

## Coping With Errors

In actual practice all logical disk interface routines would contain error processing mechanisms. (Errors have been ignored for the sake of simplicity in the exerciser programs listed in Appendix B.) A typical error recovery technique consists of a two-stage procedure. First, when an error is detected, a recalibrate operation is performed followed by a retry of the failed operation. This procedure forces the drive to seek directly to the requested cylinder (lowering the probability of a seek error) and attempts to perform the requested operation an additional time. Soft (temporary) errors caused by mechanical or electrical interference do not normally recur during the retry operation; hard errors (caused by media or drive failures), on the other hand, will continue to occur during retry operations. If, after a number of retries (approximately 10), the operation continues to fail, an error message is displayed to the system operator. This error message lists the drive number, type of operation, and failure status (from the FDC). It is the operator's responsibility to take additional action as required.

# APPLICATIONS

---

## 6. File Systems

The file system provides the disk I/O interface level most familiar to users of interactive microcomputer and minicomputer systems. In a file system, all data is stored in named disk areas called files. The user and applications programs need not be concerned with the exact location of a file on the disk — the disk file system automatically determines the file location from the file name. Files may be created, read, written, modified, and finally deleted (destroyed) when they are no longer needed. Each floppy disk typically contains a directory that lists all the files existing on the disk. A directory entry for a file contains information such as file name, file size, and the disk address (track and sector) of the beginning of the file.

### File Allocation

File storage is actually allocated on the disk (by the file system) in fixed size areas called blocks. Normally a block is the same size as a disk sector. Files are created by finding and reserving enough unused blocks to contain the data in the file. Two file allocation methods are currently in widespread use. The first method allocates blocks (for a file) from a sequential pool of unused blocks. Thus, a file is always contained in a set of sequential blocks on the disk. Unfortunately, as files are created, updated, and deleted, these free-block pools become fragmented (separated from one another). When this fragmentation occurs, it often becomes impossible for the file system to create a file even though there is a sufficient number of free blocks on the disk. At this point, special programs must be run to "squeeze" or compact the disk, in order to re-create a single contiguous free-block pool.

The second file allocation method uses a more flexible technique in which individual data blocks may be located anywhere on the disk (with no restrictions). With this technique, a file directory entry contains the disk address of a file pointer block rather than the disk address of the first data block of the file. This file pointer block contains pointers (disk addresses) for each data block in the file. For example, the first pointer in the file pointer block contains the track and sector address of the first data block in the file, the second pointer contains the disk address of the second data block, etc.

In practice, pointer blocks are usually the same size as data blocks. Therefore, some files will require multiple pointer blocks. To accommodate this requirement without loss of flexibility, pointer blocks are linked together, that is, each pointer block contains the disk address of the following pointer block. The last pointer block of the file is signalled by an illegal disk address (e.g., track 0, sector 0 or track OFFH, sector OFFH).

# APPLICATIONS

---

## The Intel File System

The Intel file system (described in detail in the RMX-80 Users Guide) uses the second disk file allocation method (previously discussed). In order to lower the system overhead involved in finding free data blocks, the Intel file system incorporates a free space management data structure known as a bit map. Each disk sector is represented by a single bit in the bit map. If a bit in the bit map is set to 1, the corresponding disk sector has been allocated. A zero in the bit map indicates that the corresponding sector is free. With this technique, the process of allocating or freeing a sector is accomplished by simply altering the bit map.

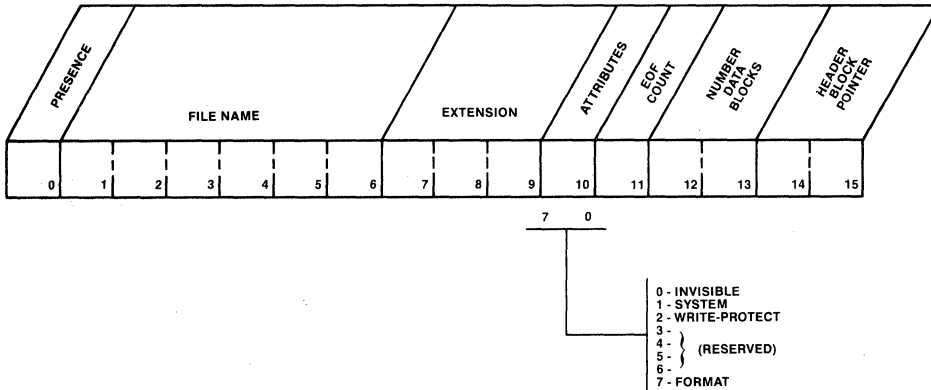
File names consist of a basic file name (up to six characters) and a file extension (up to three characters). The basic file name and the file extension are separated by a period (.). Examples of valid file names are: DRIV72.OBJ, XX.TMP, and FILE.CS. In addition, four file attributes are supported (see Figure 3 for attribute definitions).

The bit map and the file directory are placed on prespecified disk tracks (reserved for system use) beginning at track zero.

## Disk File System Functions

Table 2 illustrates the typical functions implemented by a disk file system. As an example, the disk directory function (DIR) lists disk file information on the console display terminal. Figure 3 details the contents of a display entry in the Intel file system. The PL/M procedure outlined in Figure 4 illustrates a disk directory algorithm that displays the file name, the file attributes, and the file size (in blocks) for each file in the directory.

# APPLICATIONS



AFN-01949A

## Directory Entry

**Presence** is a flag that can contain one of three values:

**000H** - The file associated with this entry is present on the disk.

**07FH** - No file is associated with this entry; the content of the rest of the entry is undefined. The first entry with its flag set to 07FH marks the current logical end of the directory and directory searches stop at this entry.

**OFFH** - The file named in this entry once existed on the disk but is currently deleted. The next file added to the directory will be placed in the first entry marked OFFH. This flag cannot, therefore, be used to (reliably) find a file that has been deleted. A value of OFFH should be thought of as simply marking an open directory entry.

**File Name** is a string of up to 6 non-blank ASCII characters specifying the name of the file associated with the directory entry. If the file name is shorter than six characters, the remaining bytes contain binary zeros. For example, the name ALPHA would be stored as: 414C50484100H.

**Extension** is a string of up to 3 non-blank ASCII characters that specifies an extension to the file name. Extensions often identify the type of data in the file such as OBJ (object module), or PLM (PL/M source module). As with the file name, unused positions in the extension field are filled with binary zeros.

Figure 3. Intel Directory Entry Format

## APPLICATIONS

**Attributes** are bits that identify certain characteristics of the file. A 1 bit indicates that the file has the attribute, while a 0 bit means that the file does not have the attribute. The bit positions and their corresponding attributes are listed below (bit 0 is the low-order or rightmost bit, bit 7 is the leftmost bit):

- 0: Invisible. Files with this attribute are not listed by the ISIS-II DIR command unless the I switch is used. All system files are invisible.
- 1: System. Files with this attribute are copied to the disk in drive 1 when the S switch is specified with the ISIS-II FORMAT command.
- 2: Write-Protect. Files with this attribute cannot be opened for output or update, nor can they be deleted or renamed.
- 3-6: These positions are reserved for future use.
- 7: Format. Files with this attribute are treated as though they are write-protected. In addition, these files are created on a new diskette when the ISIS-II FORMAT command is issued. The system files all have the FORMAT attribute and it should not be given to any other files.

**EOF Count** contains the number of the last byte in the last data block of the file. If the value of this field is 080H, for example, the last byte in the file is byte number 128 in the last data block (the last block is full).

**Number of Data Blocks** is an address variable that indicates the number of data blocks currently used by the file. ISIS-II and the RMX/80 Disk File system both maintain a counter called LENGTH that is the current number of bytes in the file. This is calculated as:

$$((\text{NUMBER OF DATA BLOCKS} - 1) \times 128 + \text{EOF COUNT}).$$

**Header Block Pointer** is the address of the file's header block. The high byte of the field is the sector number and the low byte is the track number. The system "finds" a disk file by searching the directory for the name and then using the header block pointer to seek to the beginning of the file.

Figure 3. Intel Directory Entry Format (Continued)

## APPLICATIONS

```
dir: procedure(drv,dens)    public;
  declare  drv              byte,
           dens            byte,
           sector          byte,
           i               byte,
           dir$ptr         byte,
           dir$entry       based rdbptr structure (presence byte,
           file$name(6)   byte,extension(3) byte,
           attribute byte,eof$count byte,
           data$blocks   address,header$ptr address),
           size (5)        byte,

           invisible$flag  literally '1',
           system$flag     literally '2',
           protected$flag  literally '4',
           format$flag     literally '80H';

/* The disk directory starts at cylinder 1, sector 2 */
call seek(drv,1,0);
do sector=2 to 26;
  call read(drv,1,0,sector,dens);
  do dir$ptr=0 to 112 by 4;
    if dir$entry.presence=7FH then return;
    if dir$entry.presence=0
      then do;
        do i=0 to 5; call co(dir$entry.file$name(i)); end;
        call co(period);
        do i=0 to 2; call co(dir$entry.extension(i)); end;
        do i=0 to 4; call co(space); end;
        call convert$to$decimal(@size,dir$entry.data$blocks);
        do i=0 to 4; call co(size(i)); end;
        If (dir$entry.attribute and invisible$flag) <> 0 then call co('I');
        If (dir$entry.attribute and system$flag) <> 0 then call co('S');
        If (dir$entry.attribute and protected$flag) <> 0 then call co('W');
        If (dir$entry.attribute and format$flag) <> 0 then call co('F');
      end;
  end;
end dir;
```

AFN-01949A

Figure 4. Sample PL/M Directory Procedure

# APPLICATIONS

---

## 7. Key 8272 Software Interfacing Considerations

This section contains a quick review of Key 8272 Software design features and issues. (Most items have been mentioned in other sections of this application note.) Before designing 8272 software drivers, it is advisable that the information in this section be thoroughly understood.

### 1. Non-DMA Data Transfers

In systems that operate without a DMA controller (in the polled or interrupt driven mode), the system software is responsible for counting data transfers to/from the 8272 and generating a TC signal to the FDC when the transfer is complete.

### 2. Processor Command/Result Phase Interface

In the command phase, the driver software must write the exact number of parameters in the exact order shown in Table 5. During the result phase, the driver must read the complete result status. For example, the Format Track command requires six command bytes and presents seven result bytes. The 8272 will not accept a new command until all result bytes are read. Note that the number of command and result bytes varies from command-to-command. **Command and result phases cannot be shortened.**

During both the command and result phases, the Main Status Register must be read by the driver before each byte of information is read from, or written to, the FDC Data Register. Before each command byte is written, DIO (bit 6) must be low (indicating a data transfer from the processor) and RQM (bit 7) must be high (indicating that the FDC is ready for data). During the result phase, DIO must be high (indicating a data transfer to the processor) and RQM must also be high (indicating that data is ready for the processor).

**Note:** After the 8272 receives a command byte, the RQM flag may remain set for approximately 16 microseconds (with an 8 MHz clock). The driver should not attempt to read the Main Status Register before this time interval has elapsed; otherwise, the driver may erroneously assume that the FDC is ready to accept the next byte.

### 3. Sector Sizes

The 8272 does not support 128 byte sectors in the MFM (double-density) mode.

### 4. Drive Status Changes

The 8272 constantly polls all drives for changes in the drive ready status. This polling begins immediately following RESET. An interrupt is generated every time the FDC senses a change in the drive ready status. After reset, the FDC assumes that all drives are "not ready". If a drive is ready immediately after reset, the 8272 generates a drive status change interrupt.

# APPLICATIONS

---

## 5. Seek Commands

The 8272 FDC does not perform implied seeks. Before issuing a data read or write command, the read/write head must be positioned over the correct cylinder by means of an explicit seek command. If the head is not positioned correctly, a cylinder address error is generated.

## 6. Interrupt Processing

When the processor receives an interrupt from the FDC, the FDC may be reporting one of two distinct events:

- a) The beginning of the result phase of a previously requested read, write, or scan command.
- b) An asynchronous event such as a seek/recalibrate completion, an attention, an abnormal command termination, or an invalid command.

These two cases are distinguished by the FDC BUSY flag (bit 4) in the Main Status Register. If the FDC BUSY flag is high, the interrupt is of type (a). If the FDC BUSY flag is low, the interrupt was caused by an asynchronous event (b).

A single interrupt from the FDC may signal more than one of the above events. After receiving an interrupt, the processor must continue to issue Sense Interrupt Status commands (and service the resulting conditions) until an invalid command code is received. In this manner, all "hidden" interrupts are ferreted out and serviced.

## 7. Skip Flag (SK)

The skip flag is used during the execution of Read Data, Read Deleted Data, Read Track, and various Scan commands. This flag permits the FDC to skip unwanted sectors on a disk track.

When performing a Read Data, Read Track, or Scan command, a high SK flag indicates that the FDC is to skip over (not transfer) any sector containing a deleted data address mark. A low SK flag indicates that the FDC is to terminate the command (after reading all the data in the sector) when a deleted data address mark is encountered.

When performing a Read Deleted Data command, a high SK flag indicates that sectors containing normal data address marks are to be skipped. Note that this is just the opposite situation from that described in the last paragraph. When a data address mark is encountered during a Read Deleted Data command (and the SK flag is low), the FDC terminates the command after reading all the data in the sector.



# APPLICATIONS

---

## 8. Bad Track Maintenance

The 8272 does not internally maintain bad track information. The maintenance of this information must be performed by system software. As an example of typical bad track operation, assume that a media test determines that track 31 and track 66 of a given floppy disk are bad. When the disk is formatted for use, the system software formats physical track 0 as logical cylinder 0 (C=0 in the command phase parameters), physical track 1 as logical track 1 (C=1), and so on, until physical track 30 is formatted as logical cylinder 30 (C=30). Physical track 31 is bad and should be formatted as logical cylinder FF (indicating a bad track). Next, physical track 32 is formatted as logical cylinder 31, and so on, until physical track 65 is formatted as logical cylinder 64. Next, bad physical track 66 is formatted as logical cylinder FF (another bad track marker), and physical track 67 is formatted as logical cylinder 65. This formatting continues until the last physical track (77) is formatted as logical cylinder 75. Normally, after this formatting is complete, the bad track information is stored in a prespecified area on the floppy disk (typically in a sector on track 0) so that the system will be able to recreate the bad track information when the disk is removed from the drive and reinserted at some later time.

To illustrate how the system software performs a transfer operation on a disk with bad tracks, assume that the disk drive head is positioned at track 0 and the disk described above is loaded into the drive. If a command to read track 36 is issued by an application program, the system software translates this read command into a seek to physical track 37 (since there is one bad track between 0 and 36, namely 31) followed by a read of logical cylinder 36. Thus, the cylinder parameter C is set to 37 for the Seek command and 36 for the Read Sector command.

# APPLICATIONS

---

## REFERENCES

1. Intel, "8272 Single/Double Density Floppy Disk Controller Data Sheet," Intel Corporation, 1980.
2. Intel, "An Intelligent Data Base System Using the 8272," Intel Application Note, AP-116, 1981.
3. Intel, iSBC 208 Hardware Reference Manual, Manual Order No. 143078, Intel Corporation, 1980.
4. Intel, RMX/80 User's Guide, Manual Order No. 9800522, Intel Corporation, 1978
5. Brinch Hansen, P., Operating System Principles, Prentice-Hall, Inc., New Jersey, 1973.
6. Flores, I., Computer Software: Programming Systems for Digital Computers, Prentice-Hall, Inc., New Jersey, 1965.
7. Knuth, D. E., Fundamental Algorithms, Addison-Wesley Publishing Company, Massachusetts, 1975.
8. Shaw, A. C., The Logical Design of Operating Systems, Prentice-Hall, Inc., New Jersey, 1974.
9. Watson, R. W., Time Sharing System Design Concepts, McGraw-Hill, Inc., New York, 1970.
10. Zarrella, J., Operating Systems: Concepts and Principles, Microcomputer Applications, California, 1979.

**APPENDIX A  
8272 FDC DEVICE DRIVER SOFTWARE**

# APPLICATIONS

PL/M-86 COMPILER 8272 FLOPPY DISK CONTROLLER DEVICE DRIVERS

ISIS-II PL/M-86 V1.2 COMPILATION OF MODULE DRIVERS  
OBJECT MODULE PLACED IN :Fl:driv72.OBJ  
COMPILER INVOKED BY: plm86 :Fl:driv72.p86 DEBUG

```
$title('8272 floppy disk controller device drivers')
$nointvector
$optimize(2)
$large

1 drivers: do;

2 1 declare
   /* floppy disk port definitions */
   fdc$status$port    literally `30H`,      /* 8272 status port */
   fdc$data$port      literally `31H`;      /* 8272 data port */

3 1 declare
   /* floppy disk commands */
   sense$int$status   literally `08H`;

4 1 declare
   /* interrupt definitions */
   fdc$int$level      literally `33`;      /* fdc interrupt level */

5 1 declare
   /* return status and error codes */
   error              literally `0`,
   ok                 literally `1`,
   complete           literally `3`,
   false             literally `0`,
   true              literally `1`,
   error$in          literally `not`,
   propagate$error    literally `return error`,

   stat$ok            literally `0`,      /* fdc operation completed without errors */
   stat$busy          literally `1`,      /* fdc is busy, operation cannot be started */
   stat$error         literally `2`,      /* fdc operation error */
   stat$command$error literally `3`,      /* fdc not ready for command phase */
   stat$result$error  literally `4`,      /* fdc not ready for result phase */
   stat$invalid       literally `5`;      /* invalid fdc command */

6 1 declare
   /* masks */
   busy$mask          literally `10H`,
   DIO$mask           literally `40H`,
   RQM$mask           literally `80H`,
   seek$mask          literally `0FH`,
   result$error$mask  literally `0COH`,
   result$drive$mask  literally `03H`,
   result$ready$mask  literally `08H`;

7 1 declare
   /* drive numbers */
   max$no$drives      literally `3`,
   fdc$general         literally `4`;

8 1 declare
   /* miscellaneous control */
   any$drive$seeking  literally `((input(fdc$status$port) and seek$mask) <> 0)`,
   command$code        literally `(docb.disk$command(0) and 1FH)`,
   DIO$set$for$in      literally `((input(fdc$status$port) and DIO$mask)=0)`,
   DIO$set$for$out     literally `((input(fdc$status$port) and DIO$mask)<>0)`,
   extract$drive$no    literally `(docb.disk$command(1) and 03H)`,
   fdc$busy            literally `((input(fdc$status$port) and busy$mask) <> 0)`,
   no$fdc$error        literally `possible$error(command$code) and ((docb.disk$result(0)
   and result$error$mask) = 0)`,

   wait$for$op$complete literally `do while not operation$complete(drive$no); end`,
   wait$for$RQM        literally `do while (input(fdc$status$port) and RQM$mask) = 0; end;`;

9 1 declare
   /* structures */
   docb$type           literally /* disk operation control block */
   `(dma$op byte,dma$addr word, dma$addr$ext byte,dma$count word,
   disk$command(9) byte,disk$result(7) byte,misc byte)`;

$sect
10 1 declare
   drive$status$change(4) byte public, /* when set - indicates that drive status changed */
   drive$ready(4) byte public; /* current status of drives */
```

# APPLICATIONS

```

11 1 declare
    operation$in$progress(5) byte, /* internal flags for operation with multiple drives */
    operation$complete(5) byte, /* fdc execution phase completed */
    operation$docb$ptr(5) pointer, /* pointers for operations in progress */
    interrupt$docb structure docb$type, /* temporary docb for interrupt processing */
    global$drive$no byte; /* drive number of non-overlapped operation
                           in progress - if any */

12 1 declare
    /* internal vectors that contain command operational information */
    no$result(32) byte /* no result phase to command */
    data(0,0,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0),
    immed$result(32) byte /* immediate result phase for command */
    data(0,0,0,0,1,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0),
    overlap$operation(32) byte /* command permits overlapped operation of drives */
    data(0,0,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0),
    drive$no$present(32) byte /* drive number present in command information */
    data(0,0,1,0,1,1,1,1,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0),
    possible$error(32) byte /* determines if command can return with an error */
    data(0,0,1,0,0,1,1,1,1,0,1,0,1,0,1,0,1,0,1,0,1,0,0,0,0,0,0,0,0,0,0,0),
    command$length(32) byte /* contains number of command bytes for each command */
    data(0,0,9,3,2,9,9,2,1,9,2,0,9,6,0,3,0,9,0,0,0,0,0,0,0,9,0,0,9,0,0),
    valid$command(32) byte /* flags invalid command codes */
    data(0,0,1,1,1,1,1,1,1,1,0,1,1,0,1,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0);

$seject

/**** initialization for the 8272 fdc driver software. This procedure must
    be called prior to execution of any driver software. ****/

13 1 initialize$drivers: procedure public;
    /* initialize 8272 drivers */
14 2 declare drv$no byte;

15 2 do drv$no=0 to max$no$drives;
16 3 drive$ready(drv$no)=false;
17 3 drive$status$change(drv$no)=false;
18 3 operation$in$progress(drv$no)=false;
19 3 operation$complete(drv$no)=false;
20 3 end;

21 2 operation$in$progress(fdc$general)=false;
22 2 operation$complete(fdc$general)=false;
23 2 global$drive$no=0;

24 2 end initialize$drivers;

/**** wait until the 8272 fdc is ready to receive command/parameter bytes
    in the command phase. The 8272 is ready to receive command bytes
    when the RQM flag is high and the DIO flag is low. ****/

25 1 fdc$ready$for$command: procedure byte;

26 2 /* wait for valid flag settings in status register */
    call time(1);

27 2 /* wait for "master request" flag */
    wait$for$RQM;

30 2 /* check data direction flag */
    if DIO$set$for$input
32 2 then return ok;
    else return error;

33 2 end fdc$ready$for$command;

/**** wait until the 8272 fdc is ready to return data bytes in the result
    phase. The 8272 is ready to return a result byte when the RQM and DIO
    flags are both high. The busy flag in the main status register will
    remain set until the last data byte of the result phase has been read
    by the processor. ****/

34 1 fdc$ready$for$result: procedure byte;

35 2 /* wait for valid settings in status register */
    call time(1);

36 2 /* result phase has ended when the 8272 busy flag is reset */
    if not fdc$busy
        then return complete;

```

# APPLICATIONS

```

    /* wait for "master request" flag */
38 2  wait$for$RQM;

    /* check data direction flag */
41 2  if DIO$set$for$output
43 2  then return ok;
    else return error;

44 2  end fdc$ready$for$result;

    /**** output a single command/parameter byte to the 8272 fdc. The "data$byte"
        parameter is the byte to be output to the fdc. ****/

45 1  output$byte$to$fdc: procedure (data$byte) byte;
46 2  declare data$byte byte;

    /* check to see if fdc is ready for command */
47 2  if not fdc$ready$for$command
    then propagate$error;

49 2  output (fdc$data$port)=data$byte;

50 2  return ok;
51 2  end output$byte$to$fdc;

    /**** input a single result byte from the 8272 fdc. The "data$byte$ptr"
        parameter is a pointer to the memory location that is to contain
        the input byte. ****/

52 1  input$byte$from$fdc: procedure (data$byte$ptr) byte;
53 2  declare data$byte$ptr pointer;
54 2  declare
    data$byte based data$byte$ptr byte,
    status byte;

    /* check to see if fdc is ready */
55 2  status=fdc$ready$for$result;
56 2  if error$in status
    then propagate$error;

58 2  /* check for result phase complete */
    if status=complete
    then return complete;

60 2  data$byte=input (fdc$data$port);
61 2  return ok;
62 2  end input$byte$from$fdc;

$reject

    /**** output the dma mode, the dma address, and the dma word count to the
        8237 dma controller. Also output the high order four bits of the
        address to the address extension latch. Finally, start the disk
        dma channel. The "docb$ptr" parameter is a pointer to the appropriate
        disk operation control block. ****/

63 1  output$controls$to$dma: procedure (docb$ptr);
64 2  declare docb$ptr pointer;
65 2  declare docb based docb$ptr structure docbtype;

66 2  declare
    /* dma port definitions */
    dma$upper$addr$port  literally '10H',      /* upper 4 bits of current address */
    dma$disk$addr$port   literally '00H',      /* current address port */
    dma$disk$word$count  literally '01H',      /* word count port */
    dma$command$port     literally '08H',      /* command port */
    dma$mode$port        literally '0BH',      /* mode port */
    dma$mask$sr$port     literally '0AH',      /* mask set/reset port */
    dma$clear$ff$port    literally '0CH',      /* clear first/last flip-flop port */
    dma$master$clear$port literally '0DH',      /* dma master clear port */
    dma$mask$port        literally '0FH',      /* parallel mask set port*/

    dma$disk$chan$start  literally '00H',      /* dma mask to start disk channel */
    dma$extended$write   literally 'shl(1,5)', /* extended write flag */
    dma$single$transfer  literally 'shl(1,6)'; /* single transfer flag */

67 2  if docb.dma$op < 3
    then do;
        /* set dma mode and clear first/last flip-flop */
69 3  output (dma$mode$port)=shl (docb.dma$op,2) or 40H;
70 3  output (dma$clear$ff$port)=0;

```

# APPLICATIONS

```

71 3      /* set dma address */
72 3      output(dma$disk$addr$port)=low(docb.dma$addr);
73 3      output(dma$disk$addr$port)=high(docb.dma$addr);
74 3      output(dma$upper$addr$port)=docb.dma$addr$ext;
75 3      /* output disk transfer word count to dma controller */
76 3      output(dma$disk$word$count)=low(docb.dma$count);
77 3      output(dma$disk$word$count)=high(docb.dma$count);
78 2      /* start dma channel 0 for fdc */
79 3      output(dma$mask$sr$port)=dma$disk$schan$start;
80 3      end;
81 2      end output$controls$to$dma;

/**** output a high-level disk command to the 8272 fdc. The number of bytes
required for each command is contained in the "command$length" table.
The "docb$ptr" parameter is a pointer to the appropriate disk operation
control block. ****/

82 1      output$command$to$fdc: procedure(docb$ptr) byte;
83 2      declare docb$ptr pointer;
84 2      declare
85 2      docb based docb$ptr structure docb$type,
86 2      cmd$byte$no byte;
87 2      disable;
88 2      /* output all command bytes to the fdc */
89 2      do cmd$byte$no=0 to command$length(command$code)-1;
90 3      if error$in output$byte$to$fdc(docb.disk$command(cmd$byte$no))
91 3      then do; enable; propagate$error; end;
92 3      end;
93 2      enable;
94 2      return ok;
95 2      end output$command$to$fdc;

/**** input the result data from the 8272 fdc during the result phase (after
command execution). The "docb$ptr" parameter is a pointer to the
appropriate disk operation control block. ****/

96 1      input$result$from$fdc: procedure(docb$ptr) byte;
97 2      declare docb$ptr pointer;
98 2      declare
99 2      docb based docb$ptr structure docb$type,
100 2      result$byte$no byte,
101 2      temp byte,
102 2      status byte;
103 2      disable;
104 2      do result$byte$no=0 to 7;
105 3      status=input$byte$from$fdc(@temp);
106 3      if error$in status
107 3      then do; enable; propagate$error; end;
108 3      if status=complete
109 3      then do; enable; return ok; end;
110 3      docb.disk$result(result$byte$no)=temp;
111 3      end;
112 2      enable;
113 2      if fdc$busy
114 2      then return error;
115 2      else return ok;
116 2      end input$result$from$fdc;

/**** cleans up after the execution of a disk operation that has no result
phase. The procedure is also used after some disk operation errors.
"drv" is the drive number, and "cc" is the command code for the
disk operation. ****/

117 1      operation$clean$up: procedure(drv,cc);
118 2      declare (drv,cc) byte;
119 2      disable;
120 2      operation$in$progress(drv)=false;
```

# APPLICATIONS

```
120 2      if not overlap$operation(cc)
122 2          then global$drive$no=0;
          enable;
123 2      end operation$clean$up;

$reject

/**** execute the disk operation control block specified by the pointer
parameter "docb$ptr". The "status$ptr" parameter is a pointer to
a byte variable that is to contain the status of the requested
operation when it has been completed. Six status conditions are
possible on return:

    0 The specified operation was completed without error.
    1 The fdc is busy and the requested operation cannot be started.
    2 Fdc error (further information is contained in the result
      storage portion of the disk operation control block - as
      described in the 8272 data sheet).
    3 Transfer error during output of the command bytes to the fdc.
    4 Transfer error during input of the result bytes from the fdc.
    5 Invalid fdc command. ****/

124 1      execute$docb: procedure(docb$ptr,status$ptr) public;
          /* execute a disk operation control block */

125 2      declare docb$ptr pointer, status$ptr pointer;
126 2      declare
          docb based docb$ptr structure docb$type,
          status based status$ptr byte,
          drive$no byte;

          /* check command validity */
127 2      if not valid$command(command$code)
          then do; status=stat$invalid; return; end;

          /* determine if command has a drive number field - if not, set the drive
          number for a general fdc command */
132 2      if drive$no$present(command$code)
134 2          then drive$no=extract$drive$no;
          else drive$no=fdc$general;

          /* an overlapped operation can not be performed if the fdc is busy */
135 2      if overlap$operation(command$code) and fdc$busy
          then do; status=stat$busy; return; end;

          /* for a non-overlapped operation, check fdc busy or any drive seeking */
140 2      if not overlap$operation(command$code) and (fdc$busy or any$drive$seeking)
          then do; status=stat$busy; return; end;

          /* check for drive operation in progress - if none, set flag and start operation */
145 2      disable;
146 2      if operation$in$progress(drive$no)
          then do; enable; status=stat$busy; return; end;
152 2      else operation$in$progress(drive$no)=true;

          /* at this point, an fdc operation is about to begin, so:
          1. reset the operation complete flag
          2. set the docb pointer for the current operation
          3. if this is not an overlapped operation, set the global drive
          number for the subsequent result phase interrupt. */
153 2      operation$complete(drive$no)=0;
154 2      operation$docb$ptr(drive$no)=docb$ptr;

155 2      if not overlap$operation(command$code)
          then global$drive$no=drive$no+1;
157 2      enable;

158 2      call output$controls$to$dma(docb$ptr);
159 2      if error$in output$command$to$fdc(docb$ptr)
          then do;
161 3          call operation$clean$up(drive$no,command$code);
162 3          status=stat$command$error;
163 3          return;
164 3      end;

          /* return immediately if the command has no result phase or completion interrupt - specify */
165 2      if no$result(command$code)
          then do;
167 3          call operation$clean$up(drive$no,command$code);
168 3          status=stat$ok;
169 3          return;
170 3      end;
```



# APPLICATIONS

```
171 2     if immed$result(command$code)
173 3         then do;
            if error$in input$result$from$fdc(docb$ptr)
            then do;
175 4                 call operation$clean$up(drive$no,command$code);
176 4                 status=stat$result$error;
177 4                 return;
178 4             end;
179 3         end;
180 2     else do;
181 3         wait$for$op$complete;
183 3         if docb.misc = error
            then do; status=stat$result$error; return; end;
188 3     end;

189 2     if no$fdc$error
            then status=stat$ok;
191 2     else status=stat$error;

192 2     end execute$docb;

$reject

/**** copy disk command results from the interrupt control block to the
        currently active disk operation control block if a disk operation is
        in progress. ****/

193 1     copy$int$result: procedure(drv);
194 2     declare drv byte;
195 2     declare
            i byte,
            docb$ptr pointer,
            docb based docb$ptr structure docb$type;

196 2     if operation$in$progress(drv)
            then do;
198 3         docb$ptr=operation$docb$ptr(drv);
199 3         do i=1 to 6; docb.disk$result(i)=interrupt$docb.disk$result(i); end;
202 3         docb.misc=ok;
203 3         operation$in$progress(drv)=false;
204 3         operation$complete(drv)=true;
205 3     end;

206 2     end copy$int$result;
```

/\*\*\*\* interrupt processing for 8272 fdc drivers. Basically, two types of interrupts are generated by the 8272: (a) when the execution phase of an operation has been completed, an interrupt is generated to signal the beginning of the result phase (the fdc busy flag is set when this interrupt is received), and (b) when an overlapped operation is completed or an unexpected interrupt is received (the fdc busy flag is not set when this interrupt is received).

When interrupt type (a) is received, the result bytes from the operation are read from the 8272 and the operation complete flag is set.

When an interrupt of type (b) is received, the interrupt result code is examined to determine which of the following four actions are indicated:

1. An overlapped option (recalibrate or seek) has been completed. The result data is read from the 8272 and placed in the currently active disk operation control block.
2. An abnormal termination of an operation has occurred. The result data is read and placed in the currently active disk operation control block.
3. The execution of an invalid command has been attempted. This signals the successful completion of all interrupt processing.
4. The ready status of a drive has changed. The "drive\$ready" and "drive\$ready\$status" change tables are updated. If an operation is currently in progress on the affected drive, the result data is placed in the currently active disk operation control block.

After an interrupt is processed, additional sense interrupt status commands must be issued and processed until an invalid command result is returned from the fdc. This action guarantees that all "hidden" interrupts are serviced. \*\*\*\*/

# APPLICATIONS

```

207 1      fdcint: procedure public interrupt fdc$int$level;
208 2      declare
          invalid byte,
          drive$no byte,
          docb$ptr pointer,
          docb based docb$ptr structure docb$type;

209 2      declare
          /* interrupt port definitions */
          ocw2          literally ^70H^,
          nsei          literally ^shl(1,5)^;

210 2      declare
          /* miscellaneous flags */
          result$code   literally ^shr(interrupt$docb.disk$result(0) and result$error$mask,6)^,
          result$drive$ready   literally ^((interrupt$docb.disk$result(0) and result$ready$mask) = 0)^,
          extract$result$drive$no   literally ^((interrupt$docb.disk$result(0) and result$drive$mask)^,
          end$of$interrupt   literally ^output(ocw2)=nsei^;

          /* if the fdc is busy when an interrupt is received, then the result
          phase of the previous non-overlapped operation has begun */
211 2      if fdc$busy
          then do;
          /* process interrupt if operation in progress */
213 3          if global$drive$no <> 0
          then do;
215 4              docb$ptr=operation$docb$ptr(global$drive$no-1);
216 4              if error$in input$result$from$fdc(docb$ptr)
                  then docb.misc=error;
218 4                  else docb.misc=ok;
219 4              operation$in$progress(global$drive$no-1)=false;
220 4              operation$complete(global$drive$no-1)=true;
221 4              global$drive$no=0;
222 4          end;
223 3      end;

          /* if the fdc is not busy, then either an overlapped operation has been
          completed or an unexpected interrupt has occurred (e.g., drive status
          change) */
224 2      else do;
225 3          invalid=false;
226 3          do while not invalid;

          /* perform a sense interrupt status operation - if errors are detected,
          in the actual fdc interface, interrupt processing is discontinued */
227 4      if error$in output$byte$to$fdc(sense$int$status) then go to ignore;
229 4      if error$in input$result$from$fdc(@interrupt$docb) then go to ignore;

231 4      do case result$code;

          /* case 0 - operation complete */
232 5          do;
233 6              drive$no=extract$result$drive$no;
234 6              call copy$int$result(drive$no);
235 6          end;

          /* case 1 - abnormal termination */
236 5          do;
237 6              drive$no=extract$result$drive$no;
238 6              call copy$int$result(drive$no);
239 6          end;

          /* case 2 - invalid command */
240 5          invalid=true;

          /* case 3 - drive ready change */
241 5          do;
242 6              drive$no=extract$result$drive$no;
243 6              call copy$int$result(drive$no);
244 6              drive$status$change(drive$no)=true;
245 6              if result$drive$ready
                  then drive$ready(drive$no)=true;
                  else drive$ready(drive$no)=false;
247 6              end;
248 6          end;
249 5          end;
250 4          end;
251 3          end;

252 2      ignore: end$of$interrupt;
253 2      end fdcint;

254 1      end drivers;

```

# APPLICATIONS

---

MODULE INFORMATION:

CODE AREA SIZE = 0615H 1557D  
CONSTANT AREA SIZE = 0000H 0D  
VARIABLE AREA SIZE = 0050H 80D  
MAXIMUM STACK SIZE = 0032H 50D  
564 LINES READ  
0 PROGRAM ERROR(S)

END OF PL/M-86 COMPILATION

**APPENDIX B  
8272 FDC EXERCISER PROGRAM**

# APPLICATIONS

PL/M-86 COMPILER 8272 FLOPPY DISK DRIVER EXERCISE PROGRAM

ISIS-II PL/M-86 V1.2 COMPILATION OF MODULE RUN72  
OBJECT MODULE PLACED IN :F1:run72.OBJ  
COMPILER INVOKED BY: plm86 :F1:run72.p86 DEBUG

```
$title ("8272 floppy disk driver exercise program")
$nointvector
$optimize(2)
$large
run72: do;
1
2 1 declare
    docb$type           literally          /* disk operation control block */
    (dma$op byte,dma$addr word,dma$addr$ext byte,dma$count word,
    disk$command(9) byte,disk$result(7) byte,misc byte);
3 1 declare
    /* 8272 fdc commands */
    fm                  literally "0";
    mfm                 literally "1";
    dma$mode            literally "0";
    non$dma$mode        literally "1";
    recalibrate$command literally "7";
    specify$command     literally "3";
    read$command        literally "6";
    write$command       literally "5";
    format$command      literally "0FH";
    seek$command        literally "0FH";
4 1 declare
    dma$verify          literally "0";
    dma$read            literally "1";
    dma$write           literally "2";
    dma$noop            literally "3";
5 1 declare
    /* disk operation control blocks */
    format$docb         structure docb$type,
    seek$docb           structure docb$type,
    recalibrate$docb    structure docb$type,
    specify$docb        structure docb$type,
    read$docb           structure docb$type,
    write$docb          structure docb$type;
6 1 declare
    step$rate           byte,
    head$load$time      byte,
    head$unload$time    byte,
    filler$byte         byte,
    operation$status    byte,
    interleave          byte,
    format$gap          byte,
    read$write$gap      byte,
    index               byte,
    drive               byte,
    density             byte,
    multitrack          byte,
    sector              byte,
    cylinder            byte,
    head                byte,
    tracks$per$disk     byte,
    sectors$per$track   byte,
    bytes$per$sector$code byte,
    bytes$per$sector    word;
    /* disk drive head */
    /* number of bytes in a sector on the disk */
7 1 declare
    /* read and write buffers */
    fmtblk(104)         byte public,
    wrbuf(1024)         byte public,
    rdbuf(1024)         byte public;
8 1 declare
    /* disk format initialization tables */
    sec$trk$table(3)    byte data(26,15,8),
    fmt$gap$table(8)    byte data(1BH,2AH,3AH,0,0,36H,54H,74H),
    rd$wr$gap$table(8) byte data(07H,0EH,1BH,0,0,0EH,1BH,35H);
```

# APPLICATIONS

```
9  1      declare
      /* external pointer tables and interrupt vector */
      rdbptr(2)          word external,
      wrbptr(2)          word external,
      fbp(2)            word external,
      intptr(2)          word external,
      intvec(80H)        word external;

10 1      execute$docb: procedure(docb$ptr,status$ptr) external;
11 2          declare docb$ptr pointer, status$ptr pointer;
12 2      end execute$docb;

13 1      initialize$drivers: procedure external;
14 2      end initialize$drivers;

$seject

/**** specify step rate ("srt"), head load time ("hlt"), head unload time ("hut"),
and dma or non-dma operation ("nd"). ****/

15 1      specify: procedure(srt, hlt, hut, nd);
16 2          declare (srt, hlt, hut, nd) byte;

17 2          specify$docb.dma$op=dma$noop;
18 2          specify$docb.disk$command(0)=specify$command;
19 2          specify$docb.disk$command(1)=shl((not srt)+1,4) or shr(hut,4);
20 2          specify$docb.disk$command(2)=(hlt and 0FEH) or (nd and 1);
21 2          call execute$docb(@specify$docb,@operation$status);

22 2      end specify;

/**** recalibrate disk drive
8272 automatically steps out until the track 0 signal is activated
by the disk drive. ****/

23 1      recalibrate: procedure(drv);
24 2          declare drv byte;

25 2          recalibrate$docb.dma$op=dma$noop;
26 2          recalibrate$docb.disk$command(0)=recalibrate$command;
27 2          recalibrate$docb.disk$command(1)=drv;
28 2          call execute$docb(@recalibrate$docb,@operation$status);

29 2      end recalibrate;

/**** seek drive "drv", head (side) "hd" to cylinder "cyl". ****/

30 1      seek: procedure(drv,cyl,hd);
31 2          declare (drv,cyl,hd) byte;

32 2          seek$docb.dma$op=dma$noop;
33 2          seek$docb.disk$command(0)=seek$command;
34 2          seek$docb.disk$command(1)=drv or shl(hd,2);
35 2          seek$docb.disk$command(2)=cyl;
36 2          call execute$docb(@seek$docb,@operation$status);

37 2      end seek;

/**** format a complete side ("head") of a single floppy disk in drive "drv". The density,
(single or double) is specified by flag "dens". ****/

38 1      format: procedure(drv,dens,intlve);
39 2          /* format disk */
40 2          declare (drv,dens,intlve) byte;
41 2          declare physical$sector byte;

42 2          call recalibrate(drv);
43 2          do cylinder=0 to tracks$per$disk-1;
44 2              /* set sector numbers in format block to zero before computing interleave */
45 2              do physical$sector=1 to sectors$per$track; fmbt((physical$sector-1)*4+2)=0; end;
46 2              /* physical sector 1 equals logical sector 1 */
47 2              physical$sector=1;

48 2              /* assign interleaved sectors */
49 2              do sector=1 to sectors$per$track;
50 2                  index=(physical$sector-1)*4;
```

# APPLICATIONS

```

49  4      /* change sector and index if sector has already been assigned */
        do while fmtblk(index+2) <> 0; index=index+4; physical$sector=physical$sector+1; end;

53  4      /* set cylinder, head, sector, and size code for current sector into table */
54  4      fmtblk(index)=cylinder;
55  4      fmtblk(index+1)=head;
56  4      fmtblk(index+2)=sector;
        fmtblk(index+3)=bytes$per$sector$code;

57  4      /* update physical sector number by interleave */
58  4      physical$sector=physical$sector+intlve;
        if physical$sector > sectors$per$track
60  4      then physical$sector=physical$sector-sectors$per$track;
        end;

61  3      /* seek to next cylinder */
        call seek(drv,cylinder,head);

        /* set up format control block */
62  3      format$dco.b.dma$op=dma$write;
63  3      format$dco.b.dma$addr=fbptr(0)+shl(fbptr(1),4);
64  3      format$dco.b.dma$addr$ext=0;
65  3      format$dco.b.dma$count=sectors$per$track*4-1;
66  3      format$dco.b.disk$command(0)=format$command or shl(dens,6);
67  3      format$dco.b.disk$command(1)=drv or shl(head,2);
68  3      format$dco.b.disk$command(2)=bytes$per$sector$code;
69  3      format$dco.b.disk$command(3)=sectors$per$track;
70  3      format$dco.b.disk$command(4)=format$gap;
71  3      format$dco.b.disk$command(5)=filler$byte;
72  3      call execute$dco.b(@format$dco.b,@operation$status);
73  3      end;

74  2      end format;

/**** write sector "sec" on drive "drv" at head "hd" and cylinder "cyl". The
disk recording density is specified by the "dens" flag. Data is expected to be
in the global write buffer ("wrbuf"). ****/

75  1      write: procedure(drv,cyl,hd,sec,dens);
76  2      declare (drv,cyl,hd,sec,dens) byte;

77  2      write$dco.b.dma$op=dma$write;
78  2      write$dco.b.dma$addr=wrbufptr(0)+shl(wrbufptr(1),4);
79  2      write$dco.b.dma$addr$ext=0;
80  2      write$dco.b.dma$count=bytes$per$sector-1;
81  2      write$dco.b.disk$command(0)=write$command or shl(dens,6) or shl(multitrack,7);
82  2      write$dco.b.disk$command(1)=drv or shl(hd,2);
83  2      write$dco.b.disk$command(2)=cyl;
84  2      write$dco.b.disk$command(3)=hd;
85  2      write$dco.b.disk$command(4)=sec;
86  2      write$dco.b.disk$command(5)=bytes$per$sector$code;
87  2      write$dco.b.disk$command(6)=sectors$per$track;
88  2      write$dco.b.disk$command(7)=read$write$gap;
89  2      if bytes$per$sector$code = 0
        then write$dco.b.disk$command(8)=bytes$per$sector;
        else write$dco.b.disk$command(8)=0FFH;
91  2      call execute$dco.b(@write$dco.b,@operation$status);
92  2

93  2      end write;

/**** read sector "sec" on drive "drv" at head "hd" and cylinder "cyl". The
disk recording density is defined by the "dens" flag. Data is read into
the global read buffer ("rdbuf"). ****/

94  1      read: procedure(drv,cyl,hd,sec,dens);
95  2      declare (drv,cyl,hd,sec,dens) byte;

96  2      read$dco.b.dma$op=dma$read;
97  2      read$dco.b.dma$addr=rdbufptr(0)+shl(rdbufptr(1),4);
98  2      read$dco.b.dma$addr$ext=0;
99  2      read$dco.b.dma$count=bytes$per$sector-1;
100 2      read$dco.b.disk$command(0)=read$command or shl(dens,6) or shl(multitrack,7);
101 2      read$dco.b.disk$command(1)=drv or shl(hd,2);
102 2      read$dco.b.disk$command(2)=cyl;
103 2      read$dco.b.disk$command(3)=hd;
104 2      read$dco.b.disk$command(4)=sec;
105 2      read$dco.b.disk$command(5)=bytes$per$sector$code;
106 2      read$dco.b.disk$command(6)=sectors$per$track;
107 2      read$dco.b.disk$command(7)=read$write$gap;

```

# APPLICATIONS

```

108 2      if bytes$per$sector$code = 0
110 2          then read$docb.disk$command(8)=bytes$per$sector;
111 2          else read$docb.disk$command(8)=0FFH;
112 2          call execute$docb(@read$docb,@operation$status);

112 2      end read;

$seject

/**** initialize system by setting up 8237 dma controller and 8259A interrupt
controller. ****/

113 1      initialize$system: procedure;
114 2      declare
          /* I/O ports */
          dma$disk$addr$port      literally '00H',      /* current address port */
          dma$disk$word$count$port literally '01H',      /* word count port */
          dma$command$port        literally '08H',      /* command port */
          dma$mode$port            literally '0BH',      /* mode port */
          dma$mask$sr$port         literally '0AH',      /* mask set/reset port */
          dma$clear$ff$port        literally '0CH',      /* clear first/last flip-flop port */
          dma$master$clear$port    literally '0DH',      /* dma master clear port */
          dma$mask$port            literally '0FH',      /* parallel mask set port*/
          dma$c1$addr$port         literally '02H',
          dma$c1$word$count$port   literally '03H',
          dma$c2$addr$port         literally '04H',
          dma$c2$word$count$port   literally '05H',
          dma$c3$addr$port         literally '06H',
          dma$c3$word$count$port   literally '07H',
          icw1                      literally '70H',
          icw2                      literally '71H',
          icw4                      literally '71H',
          ocw1                      literally '71H',
          ocw2                      literally '70H',
          ocw3                      literally '70H';

115 2      declare
          /* misc masks and literals */
          dma$extended$write       literally 'shl(1,5)', /* extended write flag */
          dma$single$transfer       literally 'shl(1,6)', /* single transfer flag */
          dma$disk$mode             literally '40H',
          dma$c1$mode               literally '41H',
          dma$c2$mode               literally '42H',
          dma$c3$mode               literally '43H',
          mode$8088                 literally '1',
          interrupt$base            literally '20H',
          single$controller         literally 'shl(1,1)',
          level$ssensitive           literally 'shl(1,3)',
          control$word$4$required   literally '1',
          base$icw1                 literally '10H',
          mask$all                  literally '0FFH',
          disk$interrupt$mask       literally '1';

116 2      output(dma$master$clear$port)=0;      /* master reset */
117 2      output(dma$mode$port)=dma$extended$write; /* set dma command mode */

          /* set all dma registers to valid values */
118 2      output(dma$mask$port)=mask$all;      /* mask all channels */

          /* set all addresses to zero */
119 2      output(dma$clear$ff$port)=0;          /* reset first/last flip-flop */
120 2      output(dma$disk$addr$port)=0;
121 2      output(dma$disk$word$count$port)=0;
122 2      output(dma$c1$addr$port)=0;
123 2      output(dma$c1$word$count$port)=0;
124 2      output(dma$c2$addr$port)=0;
125 2      output(dma$c2$word$count$port)=0;
126 2      output(dma$c3$addr$port)=0;
127 2      output(dma$c3$word$count$port)=0;

          /* set all word counts to valid values */
128 2      output(dma$clear$ff$port)=0;          /* reset first/last flip-flop */
129 2      output(dma$disk$word$count$port)=1;
130 2      output(dma$disk$word$count$port)=1;
131 2      output(dma$c1$word$count$port)=1;
132 2      output(dma$c1$word$count$port)=1;
133 2      output(dma$c2$word$count$port)=1;
134 2      output(dma$c2$word$count$port)=1;
135 2      output(dma$c3$word$count$port)=1;
136 2      output(dma$c3$word$count$port)=1;

```



# APPLICATIONS

```

137 2      /* initialize all dma channel modes */
138 2      output(dma$mode$port)=dma$disk$mode;
139 2      output(dma$mode$port)=dma$c1$mode;
140 2      output(dma$mode$port)=dma$c2$mode;
141 2      output(dma$mode$port)=dma$c3$mode;

141 2      /* initialize 8259A interrupt controller */
142 2      output(icw1)=single$controller or level$sensitive or control$word4$required or base$icw1;
143 2      output(icw2)=interrupt$bbase;
144 2      output(icw4)=mode$8088;          /* set 8088 interrupt mode */
144 2      output(ocw1)=not disk$interrupt$mask; /* mask all interrupts except disk */

145 2      /* initialize interrupt vector for fdc */
146 2      intvec(40H)=intptr(0);
146 2      intvec(41H)=intptr(1);

147 2      end initialize$system;

$seject

/**** main program: first format disk (all tracks on side (head) 0. Then
read each sector on every track of the disk forever. ****/

148 1      declare drive$ready(4) byte external;

149 1      /* disable until interrupt vector setup and initialization complete */
149 1      disable;

150 1      /* set initial floppy disk parameters */
151 1      density=mfm;          /* double-density */
152 1      head=0;              /* single sided */
153 1      multitrack=0;        /* no multitrack operation */
154 1      filler$byte=55H;    /* for format */
155 1      tracks$per$disk=77; /* normal floppy disk drive */
156 1      bytes$per$sector=1024; /* 1024 bytes in each sector */
157 1      interleave=6;      /* set track interleave factor */
158 1      step$rate=11;      /* 10ms for SA800 plus 1 for uncertainty */
159 1      head$load$time=40; /* 40ms head load for SA800 */
159 1      head$unload$time=240; /* keep head loaded as long as possible */

160 1      /* derive dependent parameters from those above */
161 1      bytes$per$sector$code=shr(bytes$per$sector,7);
162 2      do index=0 to 3;
162 2          if (bytes$per$sector$code and 1) <> 0
167 2              then do; bytes$per$sector$code=index; go to donebc; end;
168 2          else bytes$per$sector$code=shr(bytes$per$sector$code,1);
169 1      donebc;
170 1      sectors$per$track=sec$trk$table(bytes$per$sector$code-density);
171 1      format$gap=fmt$gap$table(shl(density,2)+bytes$per$sector$code);
171 1      read$write$gap=rd$wr$gap$table(shl(density,2)+bytes$per$sector$code);

172 1      /* initialize system and drivers */
173 1      call initialize$system;
173 1      call initialize$drivers;

174 1      /* reenale interrupts and give 8272 a chance to report on drive status
175 1      before proceeding */
174 1      enable;
175 1      call time(10);

176 1      /* specify disk drive parameters */
176 1      call specify(step$rate,head$load$time,head$unload$time,dma$mode);

177 1      drive=0;          /* run single disk drive #0 */

178 1      /* wait until drive ready */
179 2      do while 1;
179 2          if drive$ready(drive)
181 2              then go to start;
181 2      end;

182 1      start;
182 1      call format(drive,density,interleave);

183 1      do while 1;
184 2          do cylinder=0 to tracks$per$disk-1;
185 3              call seek(drive,cylinder,head);
186 3              do sector=1 to sectors$per$track;

187 4          /* set up write buffer */
187 4          do index=0 to bytes$per$sector-1; wrbuf(index)=index+sector+cylinder; end;

```

# APPLICATIONS

---

```
190 4      call write(drive,cylinder,head,sector,density);
191 4      call read(drive,cylinder,head,sector,density);

        /* check read buffer against write buffer */
192 4      if cmpw(@wrbuf,@rdbuf,shr(bytes$per$sector,1)) <> 0FFFFH
        then halt;
194 4      end;
195 3      end;
196 2      end;

197 1      end run72;
```

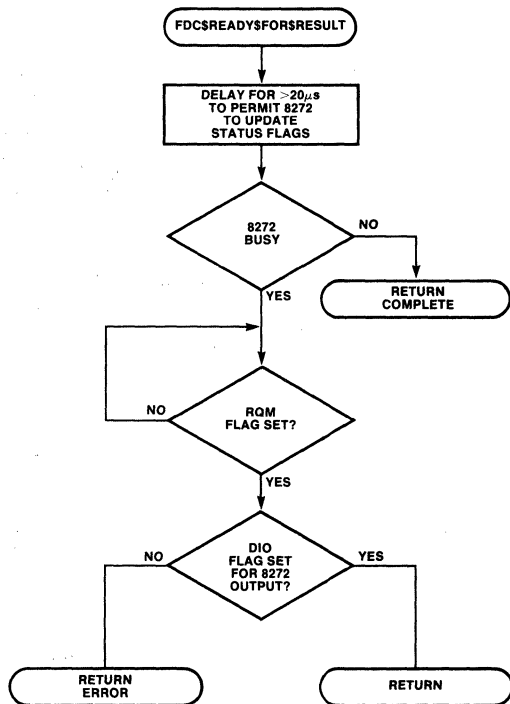
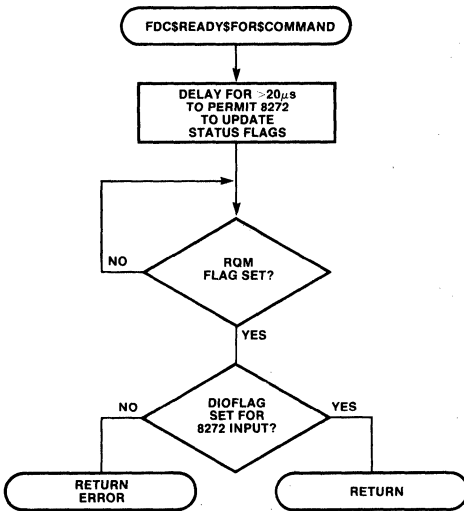
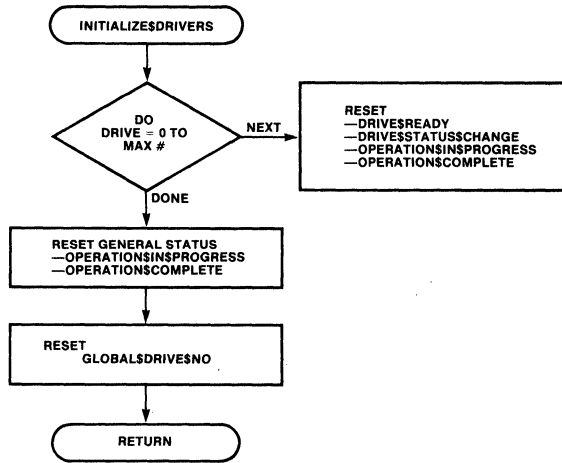
## MODULE INFORMATION:

```
CODE AREA SIZE      = 0570H  1392D
CONSTANT AREA SIZE  = 0000H   0D
VARIABLE AREA SIZE  = 0907H  2311D
MAXIMUM STACK SIZE  = 0022H   34D
412 LINES READ
0 PROGRAM ERROR(S)
```

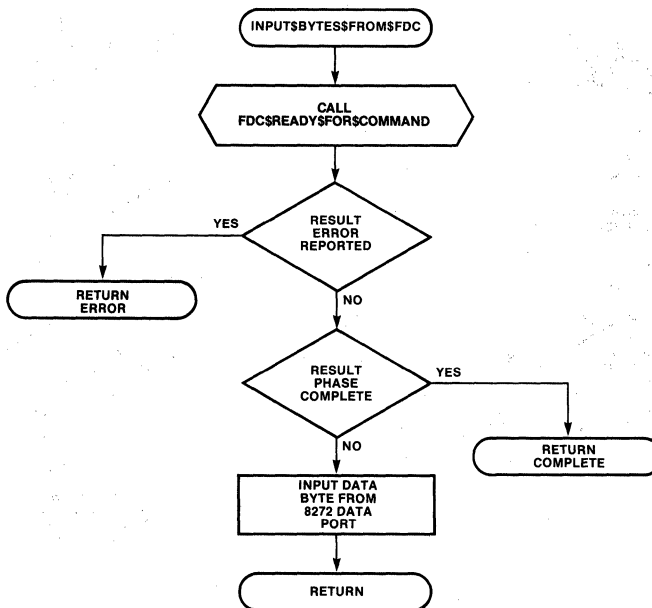
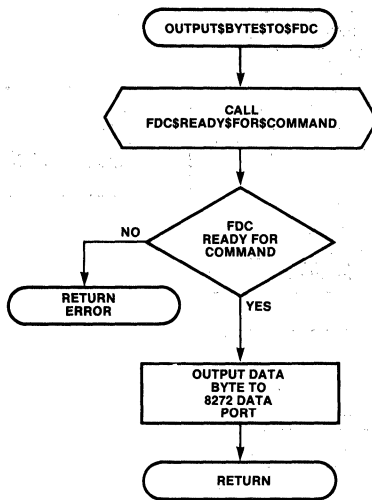
END OF PL/M-86 COMPILATION

**APPENDIX C  
8272 DRIVER FLOWCHARTS**

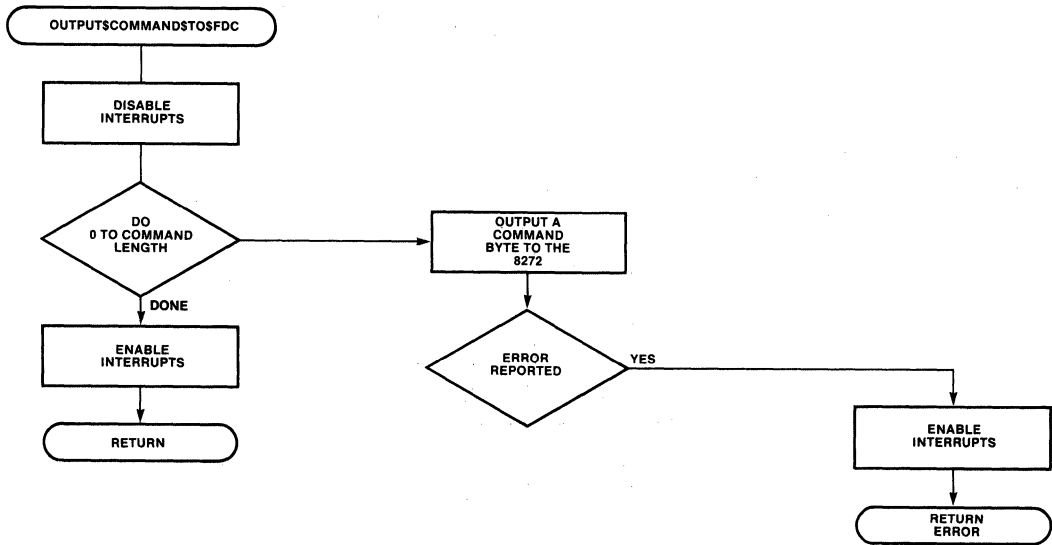
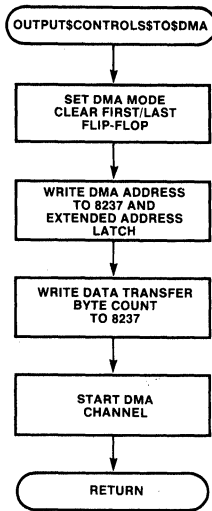
# APPLICATIONS



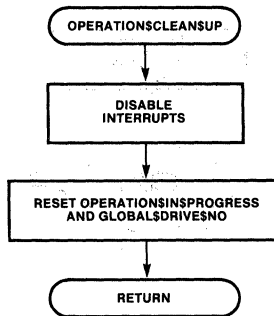
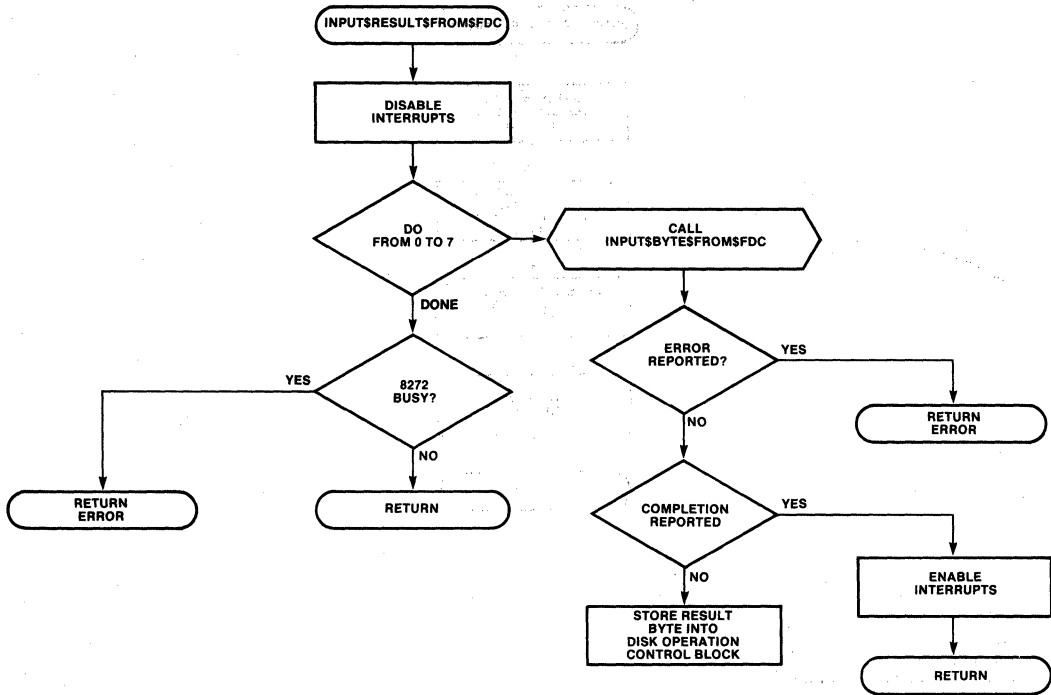
# APPLICATIONS



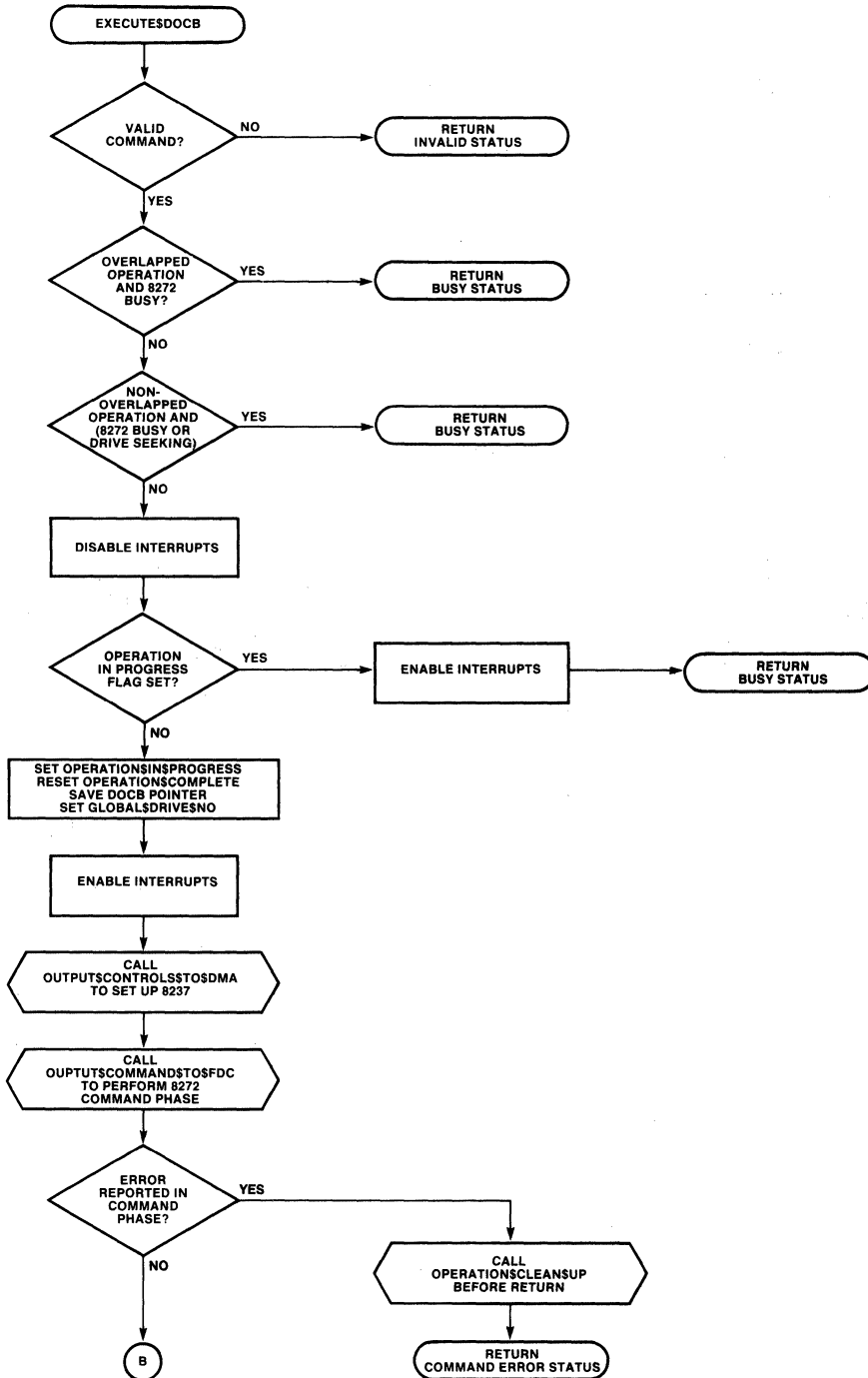
# APPLICATIONS



# APPLICATIONS

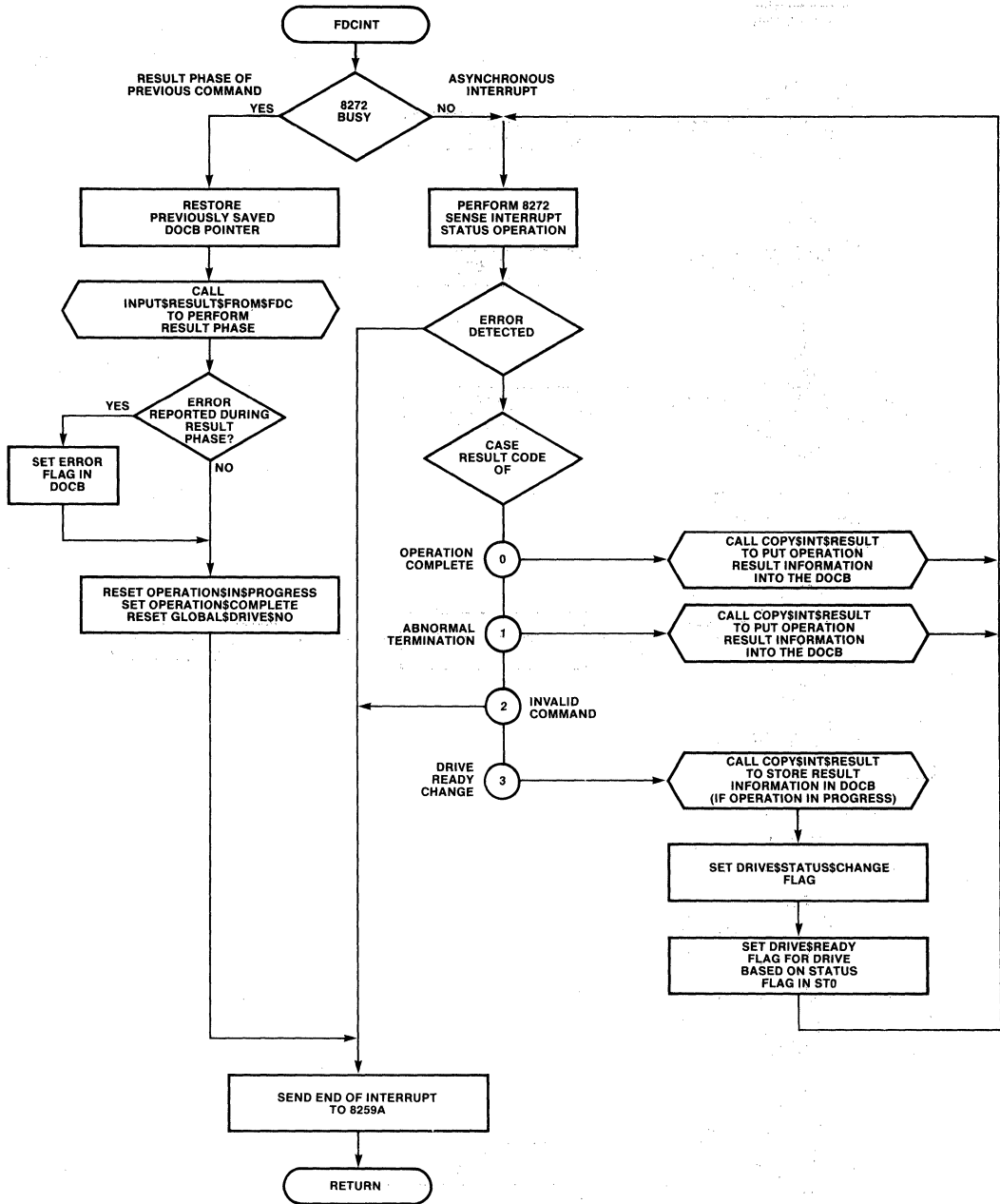


# APPLICATIONS



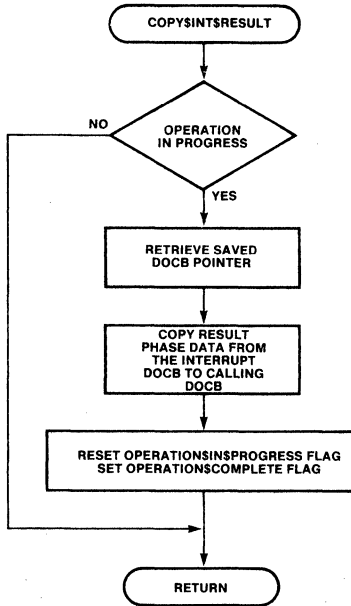


# APPLICATIONS

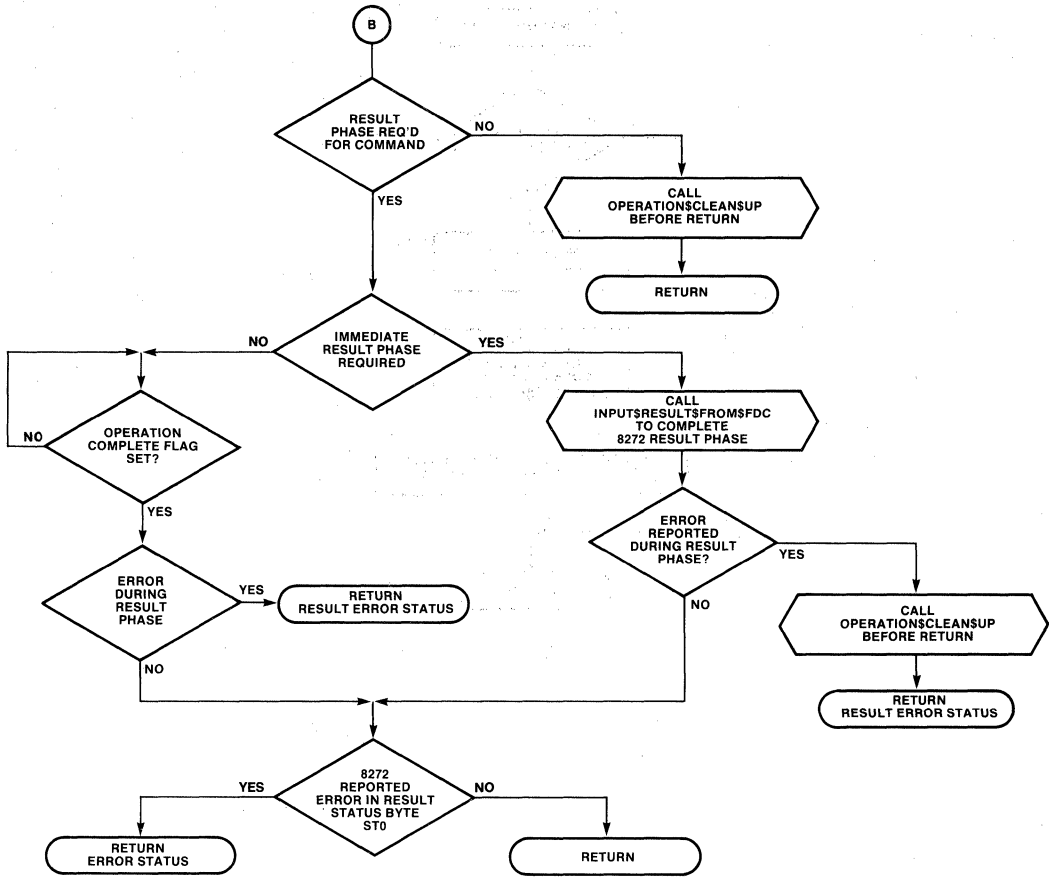


# APPLICATIONS

---



# APPLICATIONS



# Data Communications

1. The data communication system is a system that enables the exchange of data between two or more devices. It consists of a sender, a receiver, and a transmission medium. The sender transmits data to the receiver through the transmission medium. The receiver then processes the data and sends it back to the sender. This process is repeated until the data is successfully transmitted.

2. The data communication system is a system that enables the exchange of data between two or more devices. It consists of a sender, a receiver, and a transmission medium. The sender transmits data to the receiver through the transmission medium. The receiver then processes the data and sends it back to the sender. This process is repeated until the data is successfully transmitted.

---

# Using the 8251 Universal Synchronous/ Asynchronous Receiver/Transmitter

## Contents

<b>INTRODUCTION</b>	2-242
<b>COMMUNICATION FORMATS</b>	2-242
<b>BLOCK DIAGRAM</b>	2-243
Receiver	
Transmitter	
Modem Control	
I/O Control	
<b>INTERFACE SIGNALS</b>	2-245
CPU-Related Signals	
Device-Related Signals	
<b>MODE SELECTION</b>	2-248
<b>PROCESSOR DATA LINK</b>	2-251
<b>CONCLUSION</b>	2-257
<b>APPENDIX</b>	2-269
8251 Design Hints	

## INTRODUCTION

The Intel 8251 is a Universal Synchronous/Asynchronous Receiver/Transmitter (USART) which is capable of operating with a wide variety of serial communication formats. Since many peripheral devices are available with serial interfaces, the 8251 can be used to interface a microcomputer to a broad spectrum of peripherals, as well as to a serial communications channel. The 8251 is part of the MCS-80™ Microprocessor Family, and as such it is capable of interfacing to the 8080 system with a minimum of external hardware.

This application note describes the 8251 as a component and then explains its use in sample applications via several examples. A specific use of the 8251 to facilitate communication between two MCS-80 systems is discussed in detail from both the hardware and software viewpoints. The first two sections of this application note describe the 8251 first from a functional standpoint and then on a detailed level. The function of each input and output pin is fully defined. The next section describes the various operating modes and how they can be selected, and finally, a sample design is discussed using the 8251 as a data link between the MCS-80 systems.

## COMMUNICATION FORMATS

Serial communications, either on a data link or with a local peripheral, occurs in one of two basic formats; asynchronous or synchronous. These formats are similar in that they both require framing information to be added to the data to enable proper detection of the character at the receiving end. The major difference between the two formats is that the asynchronous format requires framing information to be added to each character, while the synchronous format adds framing information to blocks of data, or messages. Since the synchronous format is more efficient than the asynchronous format but requires more complex decoding, it is typically found on high-speed data links, while the asynchronous format is used on lower speed lines.

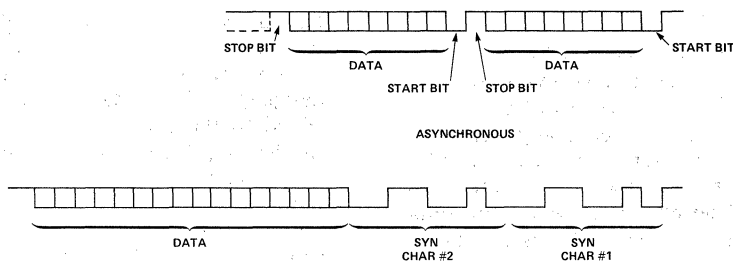
The asynchronous format starts with the basic data bits to be transmitted and adds a "START" bit to the front of them and one or more "STOP" bits behind them as they are transmitted. The START bit is a logical zero, or SPACE, and is defined as the positive voltage level by RS-232-C. The STOP bit is a logical one, or MARK, and is defined as the negative voltage level by RS-232-C. In current loop applications current flow normally indicates a MARK and lack of current a SPACE. The START bit tells the receiver to start assembling a character and allows the receiver to synchronize itself with the transmitter. Since this synchronization only

has to last for the duration of the character (the next character will contain a new START bit), this method works quite well assuming a properly designed receiver. One or more STOP bits are added to the end of the character to ensure that the START bit of the next character will cause a transition on the communication line and to give the receiver time to "catch up" with the transmitter if its basic clock happens to be running slightly slower than that of the transmitter. If, on the other hand, the receiver clock happens to be running slightly faster than the transmitter clock, the receiver will perceive gaps between characters but will still correctly decode the data. Because of this tolerance to minor frequency deviations, it is not necessary that the transmitter and receiver clocks be locked to the identical frequency for successful asynchronous communication.

The synchronous format, instead of adding bits to each character, groups characters into records and adds framing characters to the record. The framing characters are generally known as SYN characters and are used by the receiver to determine where the character boundaries are in a string of bits. Since synchronization must be held over a fairly long stream of data, bit synchronization is normally either extracted from the communication channel by the modem or supplied from an external source.

An example of the synchronous and asynchronous formats is shown in Figure 1. The synchronous format shown is fairly typical in that it requires two SYN characters at the start of the message. The asynchronous format, also typical, requires a START bit preceding each character and a single STOP bit following it. In both cases, two 8-bit characters are to be transmitted. In the asynchronous mode  $10 \cdot n$  bits are used to transmit  $n$  characters and in the synchronous mode  $8N + 16$  bits are used. For the example shown the asynchronous mode is actually more efficient, using 20 bits versus 32. To transmit a thousand characters in the asynchronous mode, however, takes 10,000 bits versus 8,016 for the synchronous format mode. For long messages the synchronous format becomes much more efficient than the asynchronous format; the crossover point for the examples shown in Figure 1 is eight characters, for which both formats require 80 bits.

In addition to the differences in format between synchronous and asynchronous communication, there are differences with regards to the type of modems that can be used. Asynchronous modems typically employ FSK (Frequency Shift Keying) techniques which simply generate one audio tone for a MARK and another for a SPACE. The receiving modem detects these tones on the telephone



**Figure 1. Transmission Formats**

line, converts them to logical signals, and presents them to the receiving terminal. Since the modem itself is not concerned with the transmission speed, it can handle baud rates from zero to its maximum speed. Synchronous modems, in contrast to asynchronous modems, supply timing information to the terminal and require data to be presented to them in synchronism with this timing information. Synchronous modems, because of this extra clocking, are only capable of operating at certain preset baud rates. The receiving modem, which has an oscillator running at the same frequency as the transmitting modem, phase locks its clock to that of the transmitter and interprets changes of phase as data.

In some cases it is desirable to operate in a hybrid mode which involves transmitting data with the asynchronous format using a synchronous modem. This occurs when an increase in operating speed is required without a change in the basic protocol of the system. This hybrid technique is known as isosynchronous and involves the generation of the start and stop bits associated with the asynchronous format, while still using the modem clock for bit synchronization.

The 8251 USART has been designed to meet a broad spectrum of requirements in the synchronous, asynchronous, and isosynchronous modes. In the synchronous mode the 8251 operates with 5, 6, 7, or 8-bit characters. Even or odd parity can be optionally appended and checked. Synchronization can be achieved either externally via added hardware or internally via SYN character detection. SYN detection can be based on one or two characters which may or may not be the same. The single or double SYN characters are inserted into the data stream automatically if the software fails to supply data in time. The automatic generation of SYN characters is required to prevent the loss of synchronization. In the asynchronous mode the 8251 operates with the same data and parity structures as it does in the synchronous mode. In addition to appending a START bit to this data, the

8251 appends 1, 1½, or 2 STOP bits. Proper framing is checked by the receiver and a status flag set if an error occurs. In the asynchronous mode the USART can be programmed to accept clock rates of 16 or 64 times the required baud rate. Isosynchronous operation is a special case of asynchronous with the multiplier rate programmed as one instead of 16 or 64. Note that X1 operation is only valid if the clocks of the receiver and transmitter are synchronized.

The 8251 USART can transmit the three formats in half or full duplex mode and is double-buffered internally (i.e., the software has a complete character time to respond to a service request). Although the 8251 supports basic data set control signals (e.g., DTR and RTS), it does not fully support the signaling described in EIA-RS-232-C. Examples of unsupported signals are Carrier Detect (CF), Ring Indicator (CE), and the secondary channel signals. In some cases an additional port will be required to implement these signals. The 8251 also does not interface to the voltage levels required by EIA-RS-232-C; drivers and receivers must be added to accomplish this interface.

## BLOCK DIAGRAM

A block diagram of the 8251 is shown in Figure 2. As can be seen in the figure, the 8251 consists of five major sections which communicate with each other on an internal data bus. The five sections are the receiver, transmitter, modem control, read/write control, and I/O Buffer. In order to facilitate discussion, the I/O Buffer has been shown broken down into its three major subsections: the status buffer, the transmit data/command buffer, and the receive data buffer.

### Receiver

The receiver accepts serial data on the RxD pin and converts it to parallel data according to the appropriate format. When the 8251 is in the asynchronous mode and it is ready to accept a character

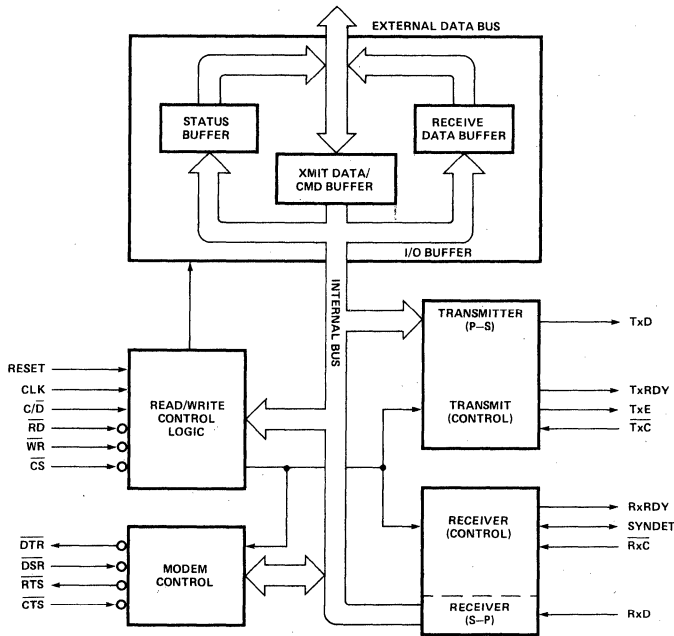


Figure 2. 8251 Block Diagram

(i.e., it is not in the process of receiving a character), it looks for a low level on the RxD line. When it sees the low level, it assumes that it is a START bit and enables an internal counter. At a count equivalent to one-half of a bit time, the RxD line is sampled again. If the line is still low, a valid START bit has probably been received and the 8251 proceeds to assemble the character. If the RxD line is high when it is sampled, then either a noise pulse has occurred on the line or the receiver has become enabled in the middle of the transmission of a character. In either case the receiver aborts its operation and prepares itself to accept a new character. After the successful reception of a START bit the 8251 clocks in the data, parity, and STOP bits, and then transfers the data on the internal data bus to the receive data register. When operating with less than 8 bits, the characters are right-justified. The RxRDY signal is asserted to indicate that a character is available.

In the synchronous mode the receiver simply clocks in the specified number of data bits and transfers them to the receiver buffer register, setting RxRDY. Since the receiver blindly groups data bits into characters, there must be a means of synchronizing the receiver to the transmitter so that the proper character boundaries are maintained in the serial data stream. This synchronization is achieved in the HUNT mode.

In the HUNT mode the 8251 shifts in data on the

RxD line one bit at a time. After each bit is received, the receiver register is compared to a register holding the SYN character (program loaded). If the two registers are not equal, the 8251 shifts in another bit and repeats the comparison. When the registers compare as equal, the 8251 ends the HUNT mode and raises the SYNDET line to indicate that it has achieved synchronization. If the USART has been programmed to operate with two SYN characters the process is as described above, except that two contiguous characters from the line must compare to the two stored SYN characters before synchronization is declared. Parity is not checked. If the USART has been programmed to accept external synchronization, the SYNDET pin is used as an input to synchronize the receiver. The timing necessary to do this is discussed in the SIGNALS section of this note. The USART enters the HUNT mode when it is initialized into the synchronous mode or when it is commanded to do so by the command instruction. Before the receiver is operated, it must be enabled by the RxE bit ( $D_2$ ) of the command instructions. If this bit is not set the receiver will not assert the RxRDY bit.

### Transmitter

The transmitter accepts parallel data from the processor, adds the appropriate framing information, serializes it, and transmits it on the Tx pin. In the asynchronous mode the transmitter always



adds a START bit; depending on how the unit is programmed, it also adds an optional even or odd parity bit, and either 1, 1½, or 2 STOP bits. In the synchronous mode no extra bits (other than parity, if enable) are generated by the transmitter unless the computer fails to send a character to the USART. If the USART is ready to transmit a character and a new character has not been supplied by the computer, the USART will transmit a SYN character. This is necessary since synchronous communications, unlike asynchronous communications, does not allow gaps between characters. If the USART is operating in the dual SYN mode, both SYN characters will be transmitted before the message can be resumed. The USART will not generate SYN characters until the software has supplied at least one character; i.e., the USART will fill 'holes' in the transmission but will not initiate transmission itself. The SYN characters which are to be transmitted by the USART are specified by the software during the initialization procedure. In either the synchronous or asynchronous modes, transmission is inhibited until TxEnable and the CTS input are asserted.

An additional feature of the transmitter is the ability to transmit a BREAK. A BREAK is a period of continuous SPACE on the communication line and is used in full duplex communication to interrupt the transmitting terminal. The 8251 USART will transmit a BREAK condition as long as bit 3 (SBRK) of the command register is set.

### Modem Control

The modem control section provides for the generation of  $\overline{\text{RTS}}$  and the reception of  $\overline{\text{CTS}}$ . In addition, a general purpose output and a general purpose input are provided. The output is labeled  $\overline{\text{DTR}}$  and the input is labeled  $\overline{\text{DSR}}$ .  $\overline{\text{DTR}}$  can be asserted by setting bit 2 of the command instruction;  $\overline{\text{DSR}}$  can be sensed as bit 7 of the status register. Although the USART itself attaches no special significance to these signals, DTR (Data Terminal Ready) is normally assigned to the modem, indicating that the terminal is ready to communicate and DSR (Data Set Ready) is a signal from the modem indicating that it is ready for communications.

### I/O Control

The Read/Write Control Logic decodes control signals on the 8080 control bus into signals which gate data on and off the USART's internal bus and controls the external I/O bus (DB<sub>0</sub>-DB<sub>7</sub>). The truth table for these operations is as follows:

If neither  $\overline{\text{READ}}$  or  $\overline{\text{WRITE}}$  is a zero, then the USART will not perform an I/O function.  $\overline{\text{READ}}$

CE	C/D	READ	WRITE	Function
0	0	0	1	CPU Reads Data from USART
0	1	0	1	CPU Reads Status from USART
0	0	1	0	CPU Writes Data to USART
0	1	1	0	CPU Writes Command to USART
1	X	X	X	USART Bus Floating (NO-OP)

and  $\overline{\text{WRITE}}$  being a zero at the same time is an illegal state with undefined results. The Read/Write Control Logic contains synchronization circuits so that the  $\overline{\text{READ}}$  and  $\overline{\text{WRITE}}$  pulses can occur at any time with respect to the clock inputs to the USART.

The I/O buffer contains the STATUS buffer, the RECEIVE DATA buffer and the XMIT DATA/CMD buffer as shown in Figure 2. Note that although there are two registers which store data for transfer to the CPU (STATUS and RECEIVE DATA), there is only one register which stores data being transferred to the USART. The sharing of the input register for both transmit data and commands makes it important to ensure that the USART does not have data stored in this register before sending a command to the device. The TxRDY signal can be monitored to accomplish this. Neither data nor commands should be transferred to the USART if TxRDY is low. Failure to perform this check can result in erroneous data being transmitted.

### INTERFACE SIGNALS

The interface signals of the 8251 USART can be broken down into two groups — a CPU-related group and a device-related group. The CPU-related signals have been designed to optimize the attachment of the 8251 to a MCS-80™ system. The device-related signals are intended to interface a modem or like device. Since many peripherals (TTY, CRT, etc.) can be obtained with a modem-like interface, the USART has a broad range of applications which do not include a modem. Note that although the USART provides a logical interface to an EIA-RS-232 device, it does not provide EIA compatible drive, and this must be added via circuitry external to the 8251. As an example of a peripheral interface application and to aid in understanding the signal descriptions which follow, Figure 3 shows a system configured to interface with a TTY or CRT.

# APPLICATIONS

CONNECT APPROPRIATE JUMPER PADS FOR APPROPRIATE USE:

CRT	TTY	TTL	SIGNAL RATE	BAND RATE
21 IO 24	21 OC 28	21 O 25	31	4000
18 IO 13	18 IO 13	18 IO 12	32	2400
14 IO 13	14 IO 13	14 IO 3	33	1200
6 IO 5	6 IO 2	6 IO 25	34	600
2 IO 5	2 IO 2	2 IO 25	35	150
27 IO 28	10 IO 18		36	75100IF 4 TO 5 IS CONNECTED

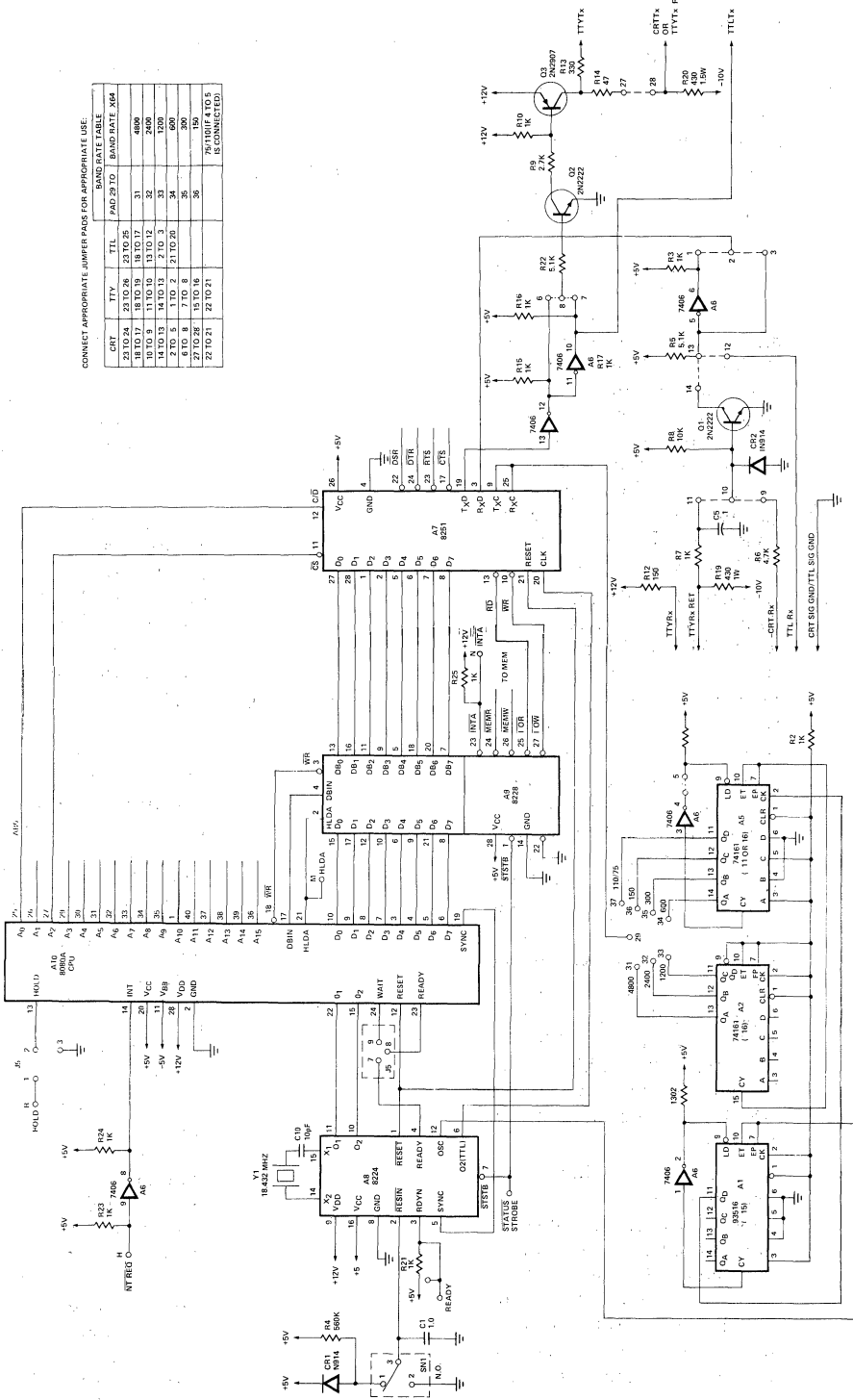


Figure 3. Terminal Interface

# APPLICATIONS

## CPU-Related Signals

V <sub>CC</sub> (26)	I	+5 Volt Supply			
GND (4)	I	+5 Volt Common	$\overline{WR}$ (10)	I	A low on this input causes the USART to accept data on the data bus as either a command or as a data character.
CLK (20)	I	The CLK input generates internal device timing. No external inputs or outputs are referenced to CLK, but the frequency of CLK must be greater than 30 times the Receiver or Transmitter clock inputs for synchronous mode or 4.5 times the clock inputs for an asynchronous mode. An additional constraint is imposed by the electrical specifications (ref. Appendix B) which require the period of CLK be between 0.42 $\mu$ sec and 1.35 $\mu$ sec. The CLK input can generally be connected to the Phase 2 (TTL) output of the 8224 clock generator.	TxRDY (15)	O	<i>Transmitter Ready.</i> This output signals the CPU that the USART is ready to accept a data character or command. It can be used as an interrupt to the system or, for polled operation, the CPU can check TxRDY using the status read operation. Note, however, that while the TxRDY status bit will be asserted whenever the XMIT DATA/CMD buffer is empty, the TxRDY output will be asserted only if the buffer is empty and the USART is enabled to transmit (i.e., CTS is low and TxEN is high). TxRDY will be reset when the USART receives a character from the program.
RESET (21)	I	A high on this input performs a master reset on the 8251. The device returns to the idle mode and will remain there until reinitialized with the appropriate control words.	TxE (18)	O	<i>Transmitter Empty.</i> A high output on this line indicates that the parallel to serial converter in the transmitter is empty. In the synchronous mode, if the CPU has failed to load a new character in time, TxE will go high momentarily as SYN characters are loaded into the transmitter to fill the gap in transmission.
DB <sub>7</sub> –DB <sub>0</sub> (8,7,6,5,2,1, 28,27)	I/O	The DB signals form a three-state bus which can be connected to the CPU data bus. Control, status, and data are transferred on this bus. Note that the CPU always remains in control of the bus and all transfers are initiated by it.	RxRDY (14)	O	<i>Transmitter Ready.</i> This output goes high to indicate that the 8251 has received a character on its serial input and is ready to transfer it to the CPU. Although the receiver runs continuously, RxRDY will only be asserted if the RxE (Receive Enable) bit in the command register has been set. RxRDY can be connected to the interrupt structure or, for polled operation, the CPU can check the condition of RxRDY using a status read operation. RxRDY will be reset when the character is read by the CPU.
$\overline{CS}$ (11)	I	<i>Chip Select.</i> A low on this input enables communication between the USART and the CPU. Chip Select should go low when the USART is being addressed by the CPU.			
$\overline{C/D}$ (12)	I	<i>Control/Data.</i> During a read operation this pin selects either status or data to be input to the CPU (high=status, low=data). During a write operation this pin causes the USART to interpret the data on the bus as a command if it is high or as data if it is low.			
$\overline{RD}$ (13)	I	A low on this input causes the USART to gate either			

## APPLICATIONS

<p>SYNDET (16) I/O</p>	<p><i>Synch Detect.</i> This line is used in the synchronous mode only. It can be either an input or output, depending on whether the initialization program sets the USART for external or internal synchronization. SYNDET is reset to a zero by RESET. When in the internal synchronization mode, the USART uses SYNDET as an output to indicate that the device has detected the required SYN character(s). A high output indicates synchronization has been achieved. If the USART is programmed to operate with double SYN characters, SYNDET will go high in the middle of the last bit of the second SYN character. SYNDET will be reset by a status read operation. When in the external synchronization mode a positive-going input on the SYNDET line will cause the 8251 to start assembling characters on the next falling edge of RxC. The high input should be maintained at least for one RxC cycle following this edge.</p>	<p>(brought low) by setting bit 5 in the command instruction.</p>
<p><math>\overline{\text{CTS}}</math> (17)</p>	<p>I</p>	<p><i>Clear to Send.</i> A low on this input enables the USART to transmit data. CTS is normally generated by the modem in response to a RTS.</p>
<p><math>\overline{\text{RxC}}</math> (25)</p>	<p>I</p>	<p><i>Receiver Clock.</i> This clock controls the data rate of characters to be received by the USART. In the synchronous mode RxC is equivalent to the baud rate, and is supplied by the modem. In asynchronous mode <math>\overline{\text{RxC}}</math> is 1, 16, or 64 times the baud rate. The clock division is preselected by the mode control instruction. Data is sampled by the USART on the rising edge of <math>\overline{\text{RxC}}</math>.</p>
<p>RxD (3)</p>	<p>I</p>	<p><i>Receiver Data.</i> Characters are received serially on this pin and assembled into parallel characters. RxD is high true (i.e., High = MARK or ONE).</p>
<p><math>\overline{\text{TxC}}</math> (9)</p>	<p>I</p>	<p><i>Transmitter Clock.</i> This clock controls the rate at which characters are transmitted by the USART. The relationship between clock rate and baud rate is the same as for RxC. Data is shifted out of the USART on the falling edge of TxC.</p>

### Device-Related Signals

<p><math>\overline{\text{DTR}}</math> (24)</p>	<p>O</p>	<p><i>Data Terminal Ready.</i> This is a general purpose output signal which can be set low by programming a '1' in command instruction bit 1. This signal allows additional device control.</p>	<p>TxD (19)</p>	<p>O</p>	<p><i>Transmit Data.</i> Parallel characters sent by the CPU are transmitted serially by the USART on this line. TxD is high true (i.e., High = MARK or ONE).</p>
<p><math>\overline{\text{DSR}}</math> (22)</p>	<p>I</p>	<p><i>Data Set Ready.</i> This is a general purpose input signal. The status of this signal can be tested by the CPU through a status read. This pin can be used to test device status and is read as bit 7 of the status register.</p>			
<p><math>\overline{\text{RTS}}</math> (23)</p>	<p>O</p>	<p><i>Request to Send.</i> This is a general purpose output signal equivalent to <math>\overline{\text{DTR}}</math>. RTS is normally used to request that the modem prepare itself to transmit (i.e., establish carrier). RTS can be asserted</p>			

### MODE SELECTION

The 8251 USART is capable of operating in a number of modes (e.g., synchronous or asynchronous). In order to keep the hardware as flexible as possible (both at the chip and end product level), these operating modes are selected via a series of control outputs to the USART. These mode control outputs must occur between the time the USART is reset and the time it is utilized for data transfer. Since the USART needs this information to structure its internal logic it is essential to complete the initialization before any attempts are made at data transfer (including reading status).

A flowchart of the initialization process appears in Figure 4. The first operation which must occur following a reset is the loading of the mode control

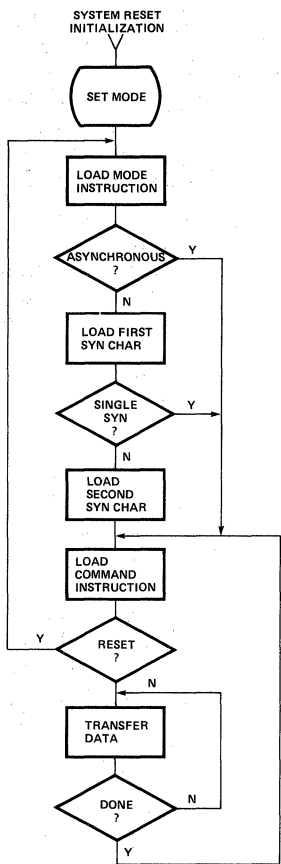


Figure 4. Initialization Flowchart

register. The mode control register is loaded by the first control output ( $C/\bar{D}=1$ ,  $\bar{RD}=1$ ,  $\bar{WR}=0$ ,  $\bar{CS}=0$ ) following a reset. The format of the mode control instruction is shown in Figure 5. The instruction can be considered as four 2-bit fields. The first 2-bit field ( $D_1 D_0$ ) determines whether the USART is to operate in the synchronous (00) or asynchronous mode. In the asynchronous mode this field also controls the clock scaling factor. As an example, if  $D_1$  and  $D_0$  are both ones, the  $\bar{RxC}$  and  $\bar{TxC}$  will be divided by 64 to establish the baud rate. The second field,  $D_3-D_2$ , determines the number of data bits in the character and the third,  $D_5-D_4$ , controls parity generation. Note that the parity bit (if enabled) is added to the data bits and is not considered as part of them when setting up the character length. As an example, standard ASCII transmission, which is seven data bits plus even parity, would be specified as:

X X 1 1 1 0 X X

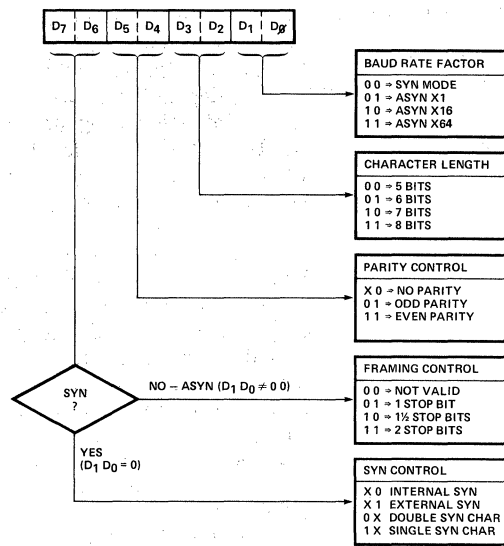


Figure 5. Mode Instruction Format

The last field,  $D_7-D_6$ , has two meanings, depending on whether operation is to be in the synchronous or asynchronous mode. For the asynchronous mode (i.e.,  $D_1 D_0 \neq 00$ ), it controls the number of STOP bits to be transmitted with the character. Since the receiver will always operate with only one STOP bit,  $D_7$  and  $D_6$  only control the transmitter. In the synchronous mode ( $D_1 D_0 = 00$ ), this field controls the synchronizing process. Note that the choice of single or double SYN characters is independent of the choice of internal or external synchronization. This is because even though the receiver may operate with external synchronization logic, the transmitter must still know whether to send one or two SYN characters should the CPU fail to supply a character in time.

Following the loading of the mode instruction the appropriate SYN character (or characters) must be loaded if synchronous mode has been specified. The SYN character(s) are loaded by the same control output instruction used to load the mode instruction. The USART determines from the mode instruction whether no, one, or two SYN characters are required and uses the control output to load SYN characters until the required number are loaded.

At completion of the load of SYN characters (or after the mode instruction in the asynchronous mode), a command character is issued to the USART. The command instruction controls the operation of the USART within the basic framework established by the mode instruction. The format of the command instruction is shown in

# APPLICATIONS

Figure 6. Note that if, as an example, the USART is waiting for a SYN character load and instead is issued an internal reset command, it will accept the command as a SYN character instead of resetting. This situation, which should only occur if two independent programs control the USART, can be avoided by outputting three all zero characters as commands before issuing the internal reset command. The USART indicates its state in a status register which can be read under program control. The format of the status register read is shown in Figure 7.

When operating the receiver it is important to realize that RxE (bit 2 of the command instruction) only inhibits the assertion of RxRDY; it does not inhibit the actual reception of characters. Because the receiver is constantly running, it is possible for it to contain extraneous data when it is enabled. To avoid problems this data should be read from the USART and discarded. The read should be done immediately following the setting of Receive Enable in the asynchronous mode, and following the setting of Enter Hunt in the synchronous mode. It is not necessary to wait for RxRDY before executing the dummy read.

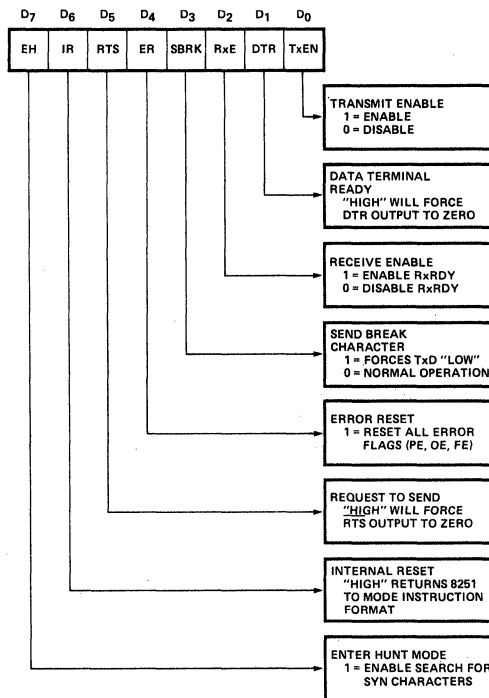


Figure 6. Command Instruction Format

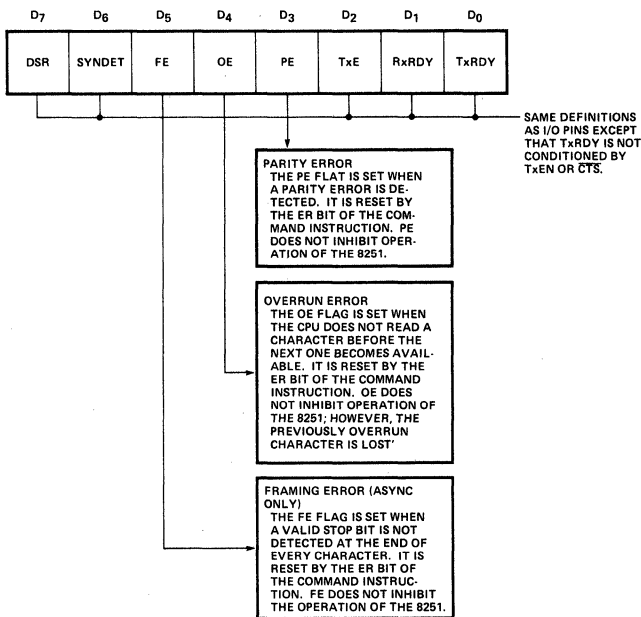


Figure 7. Status Register Format

## PROCESSOR DATA LINK

The ability to change the operating mode of the USART by software makes the 8251 an ideal device to use to implement a serial communication link. A terminal initially configured with a simple asynchronous protocol can be upgraded to a synchronous protocol such as IBM Binary Synchronous Communication by a software only upgrade. In order to demonstrate the use of the 8251 USART, the remainder of this document will describe the implementation of an interrupt-driven, full duplex communication link on the Intel MDS™ system. With minor modifications, the program developed could be used on the Intel SBC-80/10™ OEM card, thus implementing a data link between the two systems. Such a facility can be used to down-load programs, run diagnostics, and maintain common data bases in multiprocessor systems.

The factors which must be considered in the design of such a link include the desired transmission rate and format, the error checking requirements, the desirability of full duplex operation, and the physical implementation of the link. The basic requirement of the system described here is that it allow an Intel SBC-80/10 OEM card to be loaded from an MDS development system, either locally or on the switched telephone network. An additional constraint is that the modem used on the switched network be readily available and inexpensive. These requirements led to the choice of a modem such as the Bell 103A to implement the link. These modems, which support full duplex communication at up to 300 baud, are readily available from a number of sources at reasonable cost. These modems are also available in acoustically coupled versions which do not require permanent installation on the telephone network. Interface to the 103A modem is accomplished with nine wires: Protective Ground, Signal Ground, Transmitted Data, Received Data, Clear to Send, Data Set Ready, Data Terminal Ready, Carrier Detector, and Ringing Indicator.

The utilization of the interface signals to the modem is as follows:

Protective Ground	Protective Ground is used to bond the chassis ground of the modem to that of the terminal.
Signal Ground	Signal Ground provides a common ground reference between the modem and the terminal.
Transmitted Data	Transmitted Data is used to transfer serial data from the terminal to the modem.

Received Data	Received Data is used to transfer serial data from the modem to the terminal.
Clear to Send	Clear to Send indicates that the modem has established a connection with a remote modem and is ready to transmit data.
Data Set Ready	Data Set Ready indicates that the modem is connected to the telephone line and is in the data mode.
Data Terminal Ready	Data Terminal Ready is a signal from the terminal which permits the modem to enter the data mode.
Carrier Detector	Carrier Detector is identical to Clear to Send in the 103 modem and will not be used in this interface.
Ringing Indicator	Ringing Indicator indicates that the modem is receiving a ringing signal from the telephone system. This signal will not be used in the interface, since it is possible for the terminal to assert Data Terminal Ready whenever it is ready for the modem to "answer the telephone". The modem uses Data Set Ready to indicate that it has answered the call.

A block diagram showing the connections between the MDS and the SBC-80/10 through the modems is shown in Figure 8. Figure 9 shows the portion of the MDS monitor board devoted to the USARTs and Figure 10 shows the equivalent section of the SBC-80/10 board. Note that several signals on the MDS do not have the proper EIA defined voltage levels, and for this reason the adapter shown in Figure 11 was added to the MDS. The 390 pF capacitor was added to the 1488 driver to bring the rise time within EIA imposed limits of 30 volts/μsec. In Figure 7 the signal labels within the MDS and SBC-80/10 blocks correspond to the labels on the schematics, the signal labels within the modem blocks correspond to EIA conventions, and the signal labels on the wires between the blocks are abbreviations for the English language names of the signals.

As an example of how the USART clocks can be generated, circuits A27, A16, and A15 of Figure 9 form a divider of the OSC signal. The OSC signal has a frequency of 18.432 MHz and is generated by the 8224 which generates system timing for the 8080A. The 18.432 MHz signal results in a state time of 488 ns versus the normal 500 ns for the 8080A. (This does not violate 8080A specifications.) The 18.432 MHz signal can be divided by

# APPLICATIONS

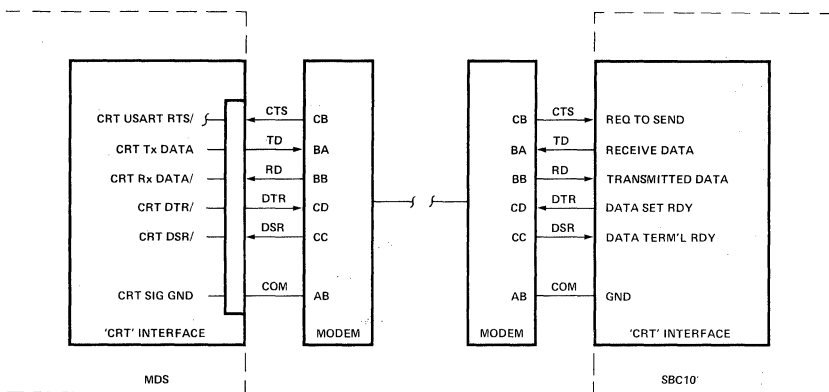


Figure 8. System Block Diagram

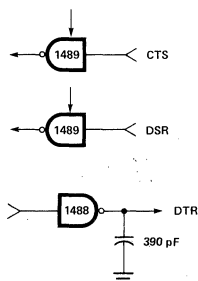


Figure 9. EIA Adapter

30 and then 64 to give a 9600 baud communication standard. The 9600 baud signal can be further divided to give 4800, 2400, 1200, 600, and 300 baud signals. The 1200 baud signal can be divided by 11 to give a 109.1 baud signal which is within 1% of the 110 baud standard signal rate. Note that because of constraints on the CLK input 9600 baud operation is not possible in the X64 mode. The divide by 64 can be accomplished by dividing by 4 with a counter and then 16 within the USART.

In order to keep the system as general purpose as possible, it was decided to transmit 8-bit data characters with an appended odd parity bit. Having a full 8-bit byte available for data enables the transmission of codes such as ASCII (which is 7-level with an additional parity bit) to be transmitted and received transparently in the system. Also, of course, it allows 8-bit bytes from the 8080A memory to be transferred in one transmission character. If error checking beyond the parity check is required, it could be added to the data record to be transmitted in the form of redundant check characters.

Before the software design of the system could be undertaken, it was necessary to decide whether service requests from the USART would be handled on a polled or interrupt driven mode. Polled operation normally results in more compact code but it requires that whatever programs are running concurrently with a transmission or reception must periodically either check the status of the USART or call a routine that does. Since it was not possible to determine what program might be running during a receive or transmit operation, it was decided to operate in an interrupt driven mode.

The program which operates the 8251 must be instructed as to what data it should transmit or receive from some other program resident in the 8080 system. To facilitate the discussion of the operation of the software, the following definitions will be made:

USRUN is the program which controls the operation of the 8251.

USER is a program which utilizes USRUN in order to effect a data transmission.

USER passes commands and parameters to USRUN by means of the control block shown in Figure 12. The first byte of the block contains the command which USER wants USRUN to execute. Valid contents of this byte are "C" which causes USRUN to initialize itself and the 8251, "R" which causes the execution of the data input (or READ) operation, and "W" which causes a data output (WRITE) operation. The second byte of the control block is used by USRUN to inform USER of the status of the requested operation. The third and fourth bytes specify the starting address of a buffer set up by USER which contains the data for a transmit operation or which will be used by USRUN to store received data. The fifth and sixth bytes are concatenated to form a positive binary



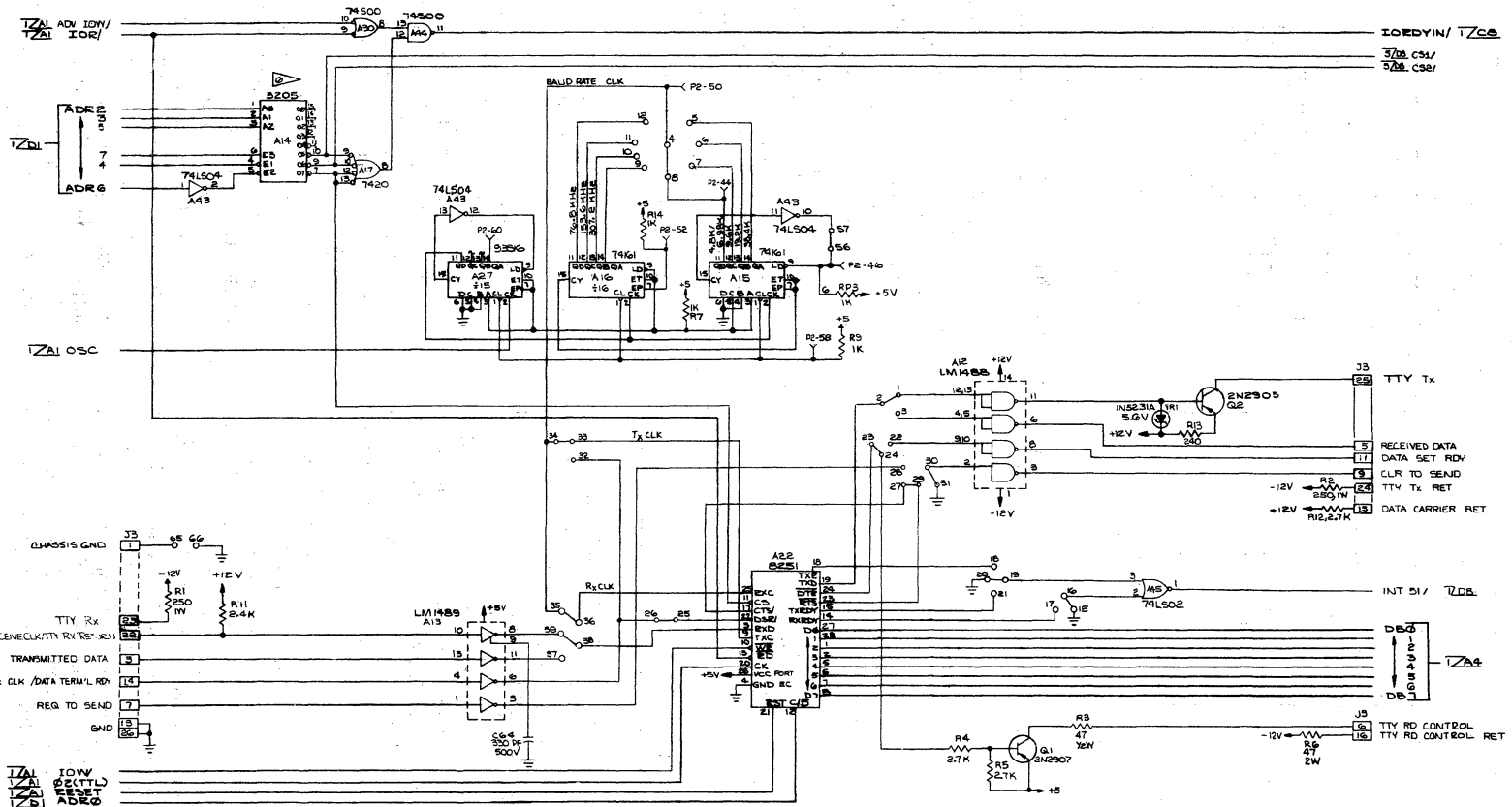


Figure 10. SBC 80/10 Serial I/O

JUMPER CONFIGURATIONS  
FOR (3) 9600 X 16 H<sub>z</sub>  
CRT (4) 1200 X 16 H<sub>z</sub>  
USART (5) 110 X 16 H<sub>z</sub>

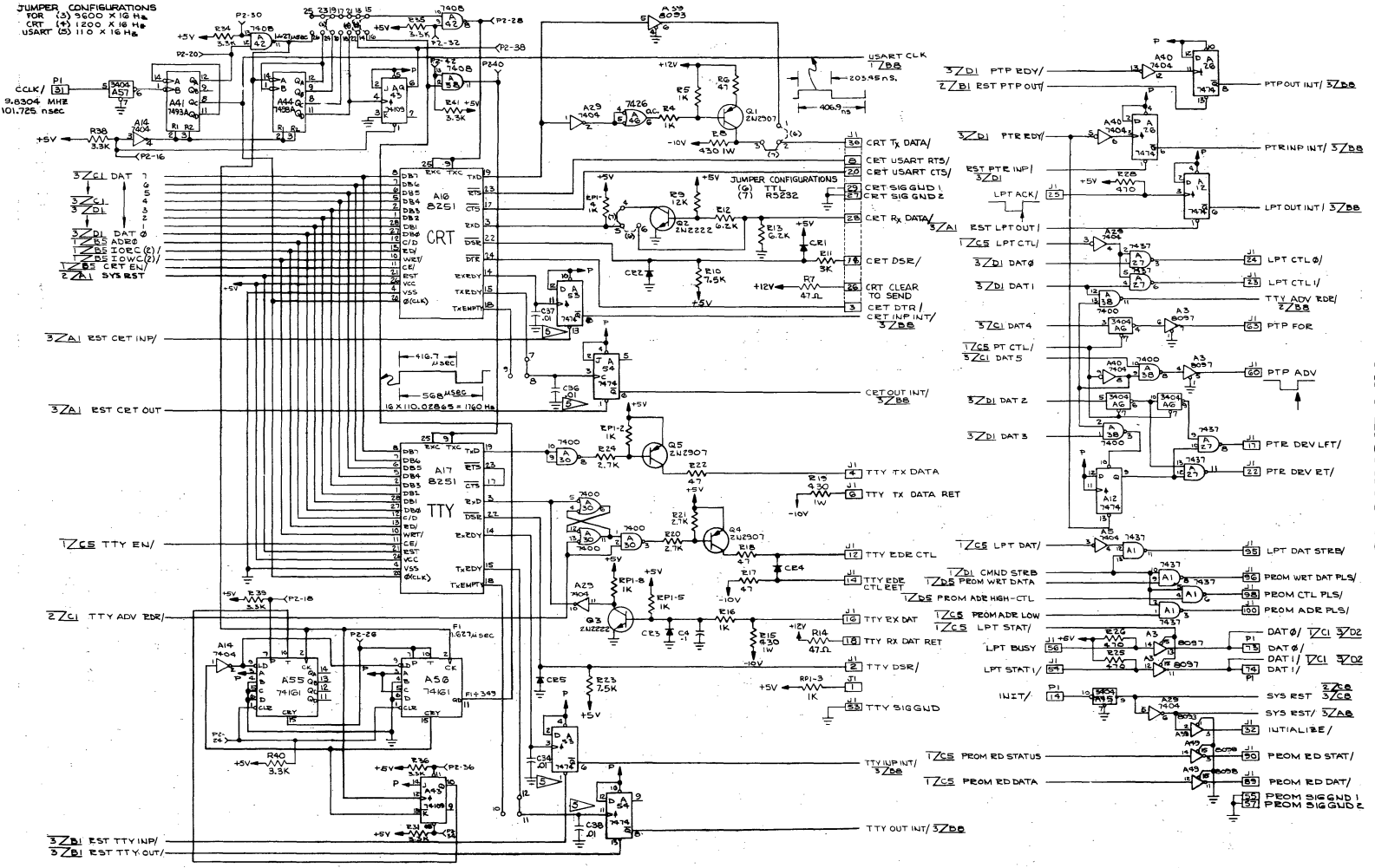


Figure 11. MDS Monitor Module

APPLICATIONS

2-254

AFN-0800A

number which specifies how many bytes of data USER wants transferred. The seventh and eighth bytes are concatenated and used by USRUN to count the number of bytes that have been transferred. When the required number of characters have been transferred, or if USRUN terminates a READ or WRITE due to an abnormal condition, then USRUN calls a subroutine at an address defined by the ninth and tenth bytes of the command block. This subroutine, which is provided by USER, must determine the state of the process and then take appropriate action.

Since USRUN must be capable of operation in a full duplex mode (i.e., be able to receive and transmit simultaneously), it keeps the address of two control blocks; one for a READ operation and one for a WRITE. The address of the controlling command block is kept in RAM locations labeled RCBA for the READ operation and TCBA for the WRITE operation. If RCBA (Receive Control Block Address) or TCBA (Transmit Control Block Address) is zero, it indicates that the corresponding operation is in an idle status.

Flowcharts of USRUN appear in Figure 13 and the listings appear in Figure 14. The first section of the flowcharts (Figures 13.1 and 13.2) consists of two subroutines which are used as convenient tools for operating on the control blocks. These routines are labeled LOADA and CLEAN. LOADA is entered with the address of a control block in registers H and L. Upon return registers D and E have been set equal to the address in the buffer which is the target of the next data transfer (i.e.,  $D,E = BAD + CCT$ ); and CCT (transferred byte count) has then been incremented. In addition, the B register is set to zero if the number of bytes that have been transferred is equal to the number requested (i.e.,  $CCT = RCT$ ). CLEAN, the second routine, is also entered with the address of a command block in the H and L registers. In addition, the Accumulator holds the status which will be placed in the STATUS byte of the command block. On exit the STATUS byte has been updated and the address of the completion routine has been placed in H and L.

Upon interrupt, control of the MCS-80 system is transferred to VECTOR (Figure 13.3). Vector is a program which saves the state of the system, gets the status of the USART and jumps to the RISR (Receive Interrupt Service Routine) or the TISR (Transmit Interrupt Service Routine), depending on which of the two ready flags is active. If neither ready flag is active, VECTOR restores the status of the running program, enables interrupts, and returns. (Interrupts are automatically disabled by the hardware upon an interrupt.) This exit from VECTOR, which is labeled VOUT, is used from other

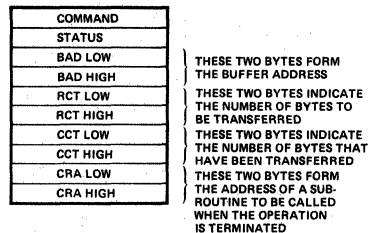


Figure 12. Control Block

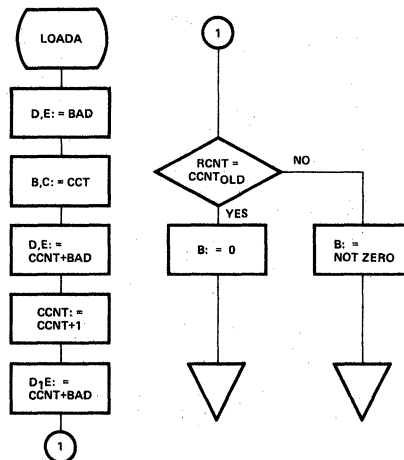


Figure 13.1. LOADA Subroutine

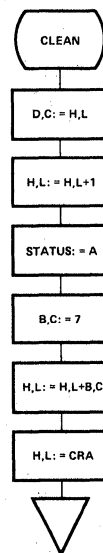


Figure 13.2. CLEAN Subroutine

# APPLICATIONS

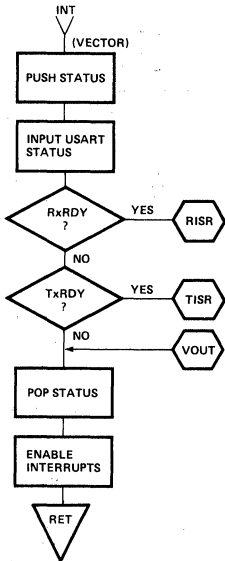


Figure 13.3. Interrupt Entry

portions of USRUN if return from the interrupt mode is required.

In addition to handling normal data transfers, TISR (Figure 13.4) checks a location in memory named TCMD in order to determine if the receive program wishes to send a command to the USART. Since the transmit data and command must share a buffer within the USART, any command output must occur when TxRDY is asserted. If TCMD is zero, TISR proceeds with the data transfer. If TCMD is non-zero, TISR calls TUTE (Transmit Utility, Figure 13.5) which, depending on the value

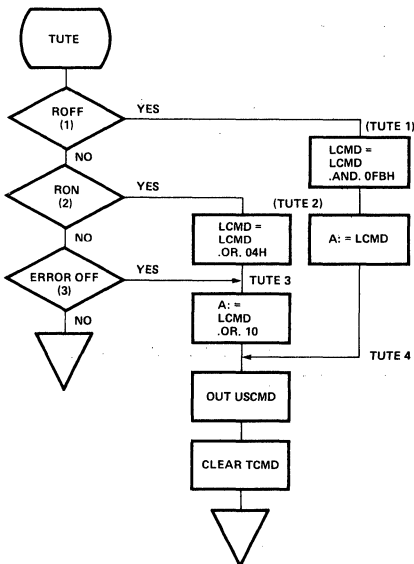


Figure 13.4. Transmit Interrupt Service Routine

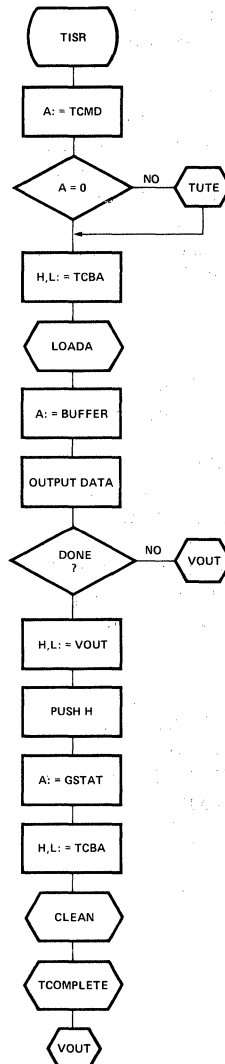


Figure 13.5. Transmit Utility Routine

in TCMD, turns off the receiver, turns on the receiver, or clears error conditions. Note that the error flags (parity, framing, and overrun) are always cleared by the software when the receiver is first enabled.

The flowchart of the RISR is shown in Figure 13.6. Note that in addition to terminating whenever the required number of characters have been received, the RISR also terminates if one of the error flags becomes set or if the received character matches a character found in a table pointed to by the label ETAB. This table, which starts at ETAB and continues until an all "ones" entry is found, can be used by USER to define special characters, such as EOT (End Of Transmission), which will terminate a READ operation. The remainder of Figure 13 (13.7) shows the decoding of the commands to USRUN. The listings also include a test USER which exercises USRUN. This program sets up a 256-byte transmit buffer and transfers it to a similar input buffer by means of a local loop. When both the READ and WRITE operations are complete, the test USER checks to insure that the two buffers are identical. If the buffers differ, the MDS monitor is called; if the data is correct, the test is repeated.

### CONCLUSION

The 8251 USART has been described both as a device and as a component in a system. Since not only modems but also many peripheral devices have a serial interface, the 8251 is an extremely useful component in a microcomputer system. A particular advantage of the device is that it is capable of operating in various modes without requiring hardware modifications to the system of which it is a part. As with any complex subsystem, however, the 8251 USART must be carefully applied so that it can be utilized to full advantage in the overall system. It is hoped that this application note will aid in the designer in the application of the 8251 USART. As a further aid to the application of the 8251, the appendix of this document includes a list of design hints based on past experience with the 8251.

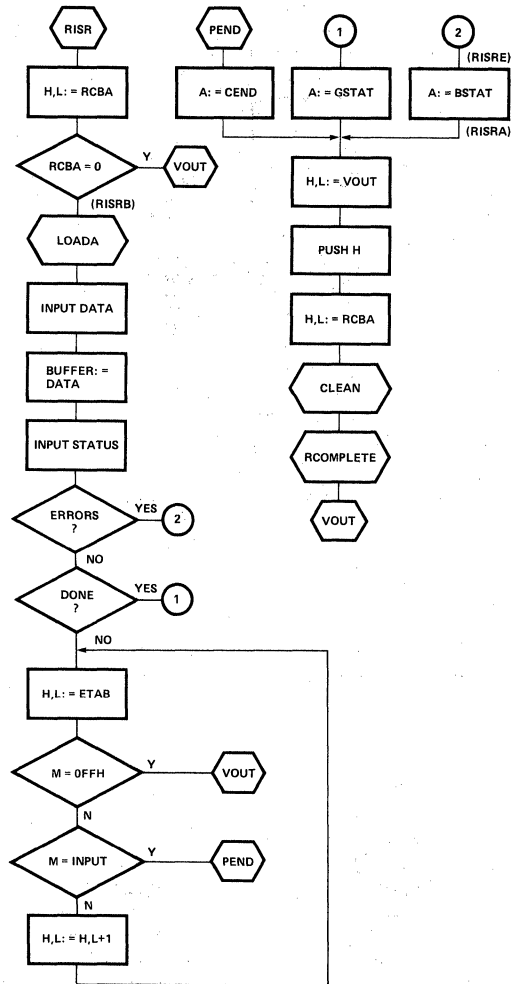


Figure 13.6. Receive Interrupt Service Routine

# APPLICATIONS

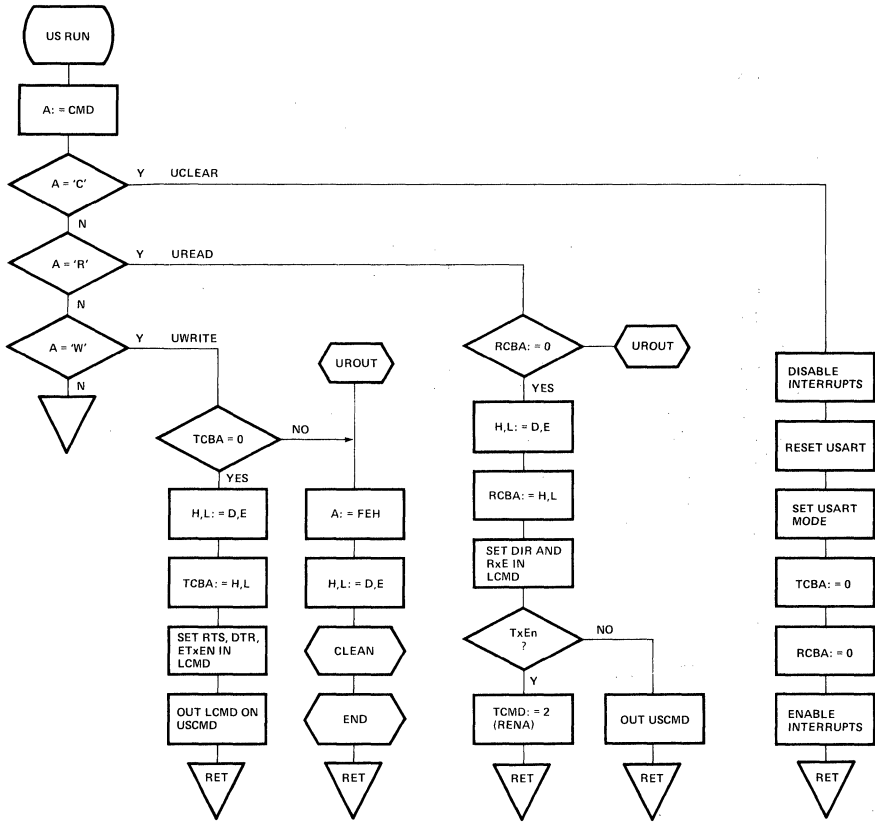


Figure 13.7. URUN Command Decode

# APPLICATIONS

Figure 14. Program Listing

```
;*****  
;  
; SYSTEM ORIGIN STATEMENT  
;  
;*****  
4000          ORG      4000H  
  
;*****  
;  
; DATA STORAGE FOR TEST USER  
;  
;*****  
4000          BUFIN:  DS      100H      ;INPUT BUFFER  
4100          BUFOUT: DS      100H      ;OUTPUT BUFFER  
4200 5200      RBLOCK: DB      'R',00H  ;RECEIVE CONTROL BLOCK  
4202 0040      RBAD:   DW      BUFIN  
4204 FF00      RRCT:   DW      OFFH  
4206 0000      RCCT:   DW      00H  
4208 1742      RCRA:   DW      RCR  
420A 5700      TBLOCK: DB      'W',00H  ;TRANSMIT CONTROL BLOCK  
420C 0041      TBAD:   DW      BUFOUT  
420E FF00      TRCT:   DW      OFFH  
4210 0000      TCCT:   DW      00H  
4212 2742      TCRA:   DW      TCR  
4214 4300      GBLOCK: DB      'C',00H  
4216 00        FLAG:   DB      00H  
  
;*****  
;  
; COMPLETION ROUTINES  
;  
;*****  
4217 AF        RCR:    XRA      A          ;CLEAR A  
4218 323B42    STA      RCBA      ;TURN OFF RECEIVE  
421B 323C42    STA      RCBA+1  
421E 3A1642    LDA      FLAG      ;GET FLAG  
4221 E60F      ANI      OFH      ;CLEAR UPPER FOUR BITS  
4223 321642    STA      FLAG      ;RESTORE FLAG  
4226 C9        RET  
4227 AF        TCR:    XRA      A          ;CLEAR A  
4228 323942    STA      TCBA      ;TURN OFF TRANSMIT  
422B 323A42    STA      TCBA+1  
422E 3A1642    LDA      FLAG      ;GET FLAG  
4231 E6F0      ANI      OFOH     ;CLEAR LOWER FOUR BITS  
4233 321642    STA      FLAG      ;RESTORE FLAG  
4236 C9        RET  
;THEN RETURN
```

# APPLICATIONS

---

```
;*****  
;  
;  
;  
;*****
```

SYSTEM EQUATES

```
00F5      USTAT  EQU    0F5H    ;USART STATUS ADDRESS  
00F5      USCMD  EQU    0F5H    ;USART CMD ADDRESS  
00F4      USDAI  EQU    0F4H    ;USART DATA INPUT ADDRESS  
00F4      USDAO  EQU    0F4H    ;USART DATA OUTPUT ADDRESS  
0000      GSTAT  EQU    00H     ;GOOD STATUS  
00FF      BSTAT  EQU    0FFH    ;BAD STATUS  
0001      CEND   EQU    01H
```

```
;*****  
;  
;  
;  
;*****
```

SYSTEM DATA TABLE

```
4237 00    LCMD:  DB      00H     ;CURRENT OPERATING COMMAND  
4238 00    TCMD:  DB      00H     ;IF NON ZERO A COMMAND TO BE SENT  
4239 0000  TCBA:  DW      00H     ;ADDRESS OF XMIT CBLOCK  
423B 0000  RCBA:  DW      00H     ;ADDRESS OF RECEIVE CBLOCK  
423D FF    MTAB:  DB      0FFH    ;END CHARACTER TABLE
```



# APPLICATIONS

```

;*****
;
;
;   LOAD ADDRESS ROUTINE
;   LOADA IS ENTERED WITH THE ADDRESS OF A CONTROL
;   BLOCK IN H,L. ON EXIT D,E CONTAINS THE ADDRESS
;   WHICH IS THE TARGET OF THE NEXT DATA TRANSFER (BAD+CCNT)
;   AND B HAS BEEN SET TO ZERO IF THE REQUESTED NUMBER OF
;   TRANSFERS HAS BEEN ACCOMPLISHED. CCNT IS INCREMENTED
;   AFTER THE TARGET ADDRESS HAS BEEN CALCULATED.
;
;*****

```

```

423E 23   LOADA: INX      H           ;D,E GETS BUFFER ADDRESS
423F 23           INX      H
4240 5E           MOV     E,M
4241 23           INX      H
4242 56           MOV     D,M           ;DONE
4243 23           INX      H           ;B,C GETS COMPLETED COUNT (CCNT)
4244 23           INX      H
4245 23           INX      H
4246 4E           MOV     C,M
4247 23           INX      H
4248 46           MOV     B,M           ;DONE
4249 EB           XCHG           ;D,E GETS BAD+CCNT
424A 09           DAD     B
424B EB           XCHG           ;DONE
424C 03           INX      B           ;CCNT GETS INCREMENTED
424D 70           MOV     M,B
424E 2B           DCX     H
424F 71           MOV     M,C           ;DONE
4250 0B           DCX     B           ;DOES OLD CCNT=RCNT?
4251 2B           DCX     H
4252 7E           MOV     A,M
4253 90           SUB     B
4254 47           MOV     B,A
4255 C0           RNZ           ;NO-RETURN WITH B NOT ZERO
4256 2B           DCX     H
4257 7E           MOV     A,M
4258 91           SUB     C
4259 47           MOV     B,A
425A C9           RET           ;RETURN WITH B=0 IF RCNT=CCNT

```

# APPLICATIONS

```

;*****
;
;      CLEAN-UP ROUTINE
;      CLEAN IS ENTERED WITH THE ADDRESS OF A CONTROL
;      BLOCK IN H,L AND A NEW STATUS TO BE
;      ENTERED INTO IT IN A. ON EXIT THE ADDRESS OF THE
;      CONTROL BLOCK IS IN D,E; THE STATUS OF THE BLOCK
;      HAS BEEN UPDATED; AND THE ADDRESS OF THE COMPLETION
;      ROUTINE IS IN H,L.
;
;*****

```

```

425B 5D      CLEAN:  MOV     E,L      ;SAVE THE ADRESS OF THE COMMAND BLOCK
425C 54      MOV     D,H
425D 23      INX     H      ;POINT AT STATUS
425E 77      MOV     M,A      ;SET STATUS EQUAL TO A
425F 010700  LXI     B,7      ;SET INDEX TO SEVEN
4262 09      DAD     B      ;POINT AT COMPLETION ADDRESS
4263 7E      MOV     A,M      ;GET LOWER ADDRESS
4264 23      INX     H      ;POINT AT UPPER ADDRESS
4265 66      MOV     H,M      ;H GETS HIGH ADDRESS BYTE
4266 6F      MOV     L,A      ;L GETS LOW ADDRESS BYTE
4267 C9      RET

```

```

;*****
;
;      INTERRUPT VECTOR ROUTINE
;      VECTOR SAVES THE STATUS OF THE RUNNING PROGRAM
;      THEN READS THE STATUS OF THE USART TO DETERMINE
;      IF A RECEIVE OR TRANSMIT INTERRUPT OCCURRED.
;      VECTOR THEN CALLS THE APPROPRIATE SERVICE ROUTINE.
;      IF NEITHER INTERRUPTS OCCURRED THEN VECTOR RESTORES
;      THE STATUS OF THE RUNNING PROGRAM. THE SERVICE
;      ROUTINES USE THE EXIT CODE, LABELED VOUT, TO EFFECT
;      THEIR EXIT FROM INTERRUPT MODE.
;
;*****

```

```

4268 F5      VECTOR:  PUSH    PSW      ;PUSH STATUS INTO THE STACK
4269 C5      PUSH    B
426A D5      PUSH    D
426B E5      PUSH    H
426C DBF5    IN      USTAT   ;GET USART ADDRESS
426E DBFA    IN      OFAH   ;MDS-GET MONITOR CARD INT. STATUS
4270 0F      RRC      ;ROTATE TWO PLACES
4271 0F      RRC      ;SO THAT CARRY=RXRDY
4272 DA8842  JC      RISR    ;IF RXRDY GO TO SERVICE ROUTINE
4275 07      RLC      ;IF NOT ROTATE BACK
4276 07      RLC      ;LEAVING TXRDY IN CARRY
4277 DAD442  JC      TISR    ;IF TXRDY THEN GO TO SERVICE ROUTINE
427A 3EFC    MVI     A,OFCH  ;MDS-CLEAR OTHER LEVEL THREE INTERRUPTS
427C D3F3    OUT     OF3H   ;MDS
427E E1      VOUT:   POP     H      ;ELSE EXIT FROM INTERRUPT MODE
427F D1      POP     D
4280 C1      POP     B
4281 3E20    MVI     A,20H   ;MDS-RESTORE CURRENT LEVEL
4283 D3FD    OUT     OFDH   ;MDS
4286 FB      EI      ;ENABLE INTERRUPTS
4287 C9      RET

```

# APPLICATIONS

```

;*****
;
; RECEIVE INTERRUPT SERVICE ROUTINE;
; RISR PROCESSES A RECEIVE INTERRUPT
; AT THE END OF RECEIVE THE USER SUPPLIED
; COMPLETION ROUTINE IS CALLED AND THEN AN
; EXIT IS TAKEN THROUGH VOUT OF THE
; VECTOR
;*****

```

```

4288 2A3B42  RISR:  LHLD  RCBA
428B 3E82      MVI  A,82H  ;MDS-CLEAR RECEIVE INTERRUPT
428D D3F3      OUT  OF3H  ;MDS
428F 2C        INR  L
4290 2D        DCR  L
4291 C29942    JNZ  RISRB
4294 24        INR  H
4295 25        DCR  H
4296 CA7E42    JZ   VOUT
4299 CD3E42    RISRB: CALL  LOADA  ;READY-SET UP ADDRESS
429C DBF4      IN   USDAI  ;GET INPUT DATA
429E 12        STAX  D      ;AND PUT IN THE BUFFER
429F 4F        MOV  C,A   ;SAVE INPUT DATA IN C
42A0 DBF5      IN   USTAT  ;GET STATUS AGAIN
42A2 E638      ANI  38H   ;MASK FOR ERROR FIELD
42A4 C2B942    JNZ  RISRE  ;NOT ZERO-TAKE ERROR EXIT
42A7 04        INR  B      ;B WAS 00 IF DONE
42A8 05        DCR  B
42A9 C2BE42    JNZ  EXCHAR ;NOT DONE-EXIT
42AC 3E00      MVI  A,GSTAT ;A GETS GOOD STATUS
42AE 217E42    RISRA: LXI  H,VOUT ;GET RETURN ADDRESS
42B1 E5        PUSH H      ;AND PUSH IT INTO THE STACK
42B2 2A3B42    LHLD  RCBA   ;POINT H,L AT THE CMD BLOCK
42B5 CD5B42    CALL  CLEAN  ;CALL CLEANUP ROUTINE
42B8 E9        PCHL  ;EFFECTIVELY CALLS COMPLETION ROUTINE
;RETURN IS TO VOUT BECAUSE OF PUSH H
42B9 3EFF      RISRE: MVI  A,BSTAT ;A GETS BAD STATUS
42BB C3AE42    JMP  RISRA  ;OTHERWISE EXIT IS NORMAL
42BE 213D42    EXCHAR: LXI  H,MTAB ;TEST CHARACTER AGAINST EXIT TABLE
42C1 7E        EXA:  MOV  A,M
42C2 FEFF      CPI  OFFH  ;END OF TABLE
42C4 CA7E42    JZ   VOUT
42C7 B9        CMP  C
42C8 CACF42    JZ   PEND  ;MATCH-TERMINATE READ
42CB 23        INX  H
42CC C3C142    JMP  EXA
42CF 3E01      PEND:  MVI  A,CEND
42D1 C3AE42    JMP  RISRA

```

# APPLICATIONS

```

;*****
;
; TRANSMIT INTERUPT SERVICE ROUTINE
; TISR PROCESSES TRANSMITTER INTERUPTS
; WHEN THE END OF A TRANSMISSION IS
; DETECTED THE USER SUPPLIED COMPLETION
; ROUTINE IS CALLED AND THEN AN EXIT IS
; TAKEN THROUGH VOUT OF VECTOR
;
;*****

```

```

42D4 3A3842 TISR: LDA TCMD ;GET POTENTIAL COMMAND
42D7 B7 ORA A ;DESIGNATE ON IT
42D8 C40443 CNZ TUTE ;DO UTILITY COMMAND
42DB 3E81 MVI A,081H ;MDS-CLEAR XMIT INTERUPTS
42DD D3F3 OUT OF3H ;MDS
42DF 2A3942 LHLD TCBA
42E2 2C INR L ;MAKE SURE HAVE VALID CONTROL BLOCK
42E3 2D DCR L
42E4 C2EC42 JNZ TISRA ;GOOD
42E7 24 INR H
42E8 25 DCR H
42E9 CA7E42 JZ VOUT ;NON VALID BLOCK (H,L=0)
42EC CD3E42 TISRA: CALL LOADA ;SET UP ADDRESS
42EF 1A LDAX D ;GET DATA FROM BUFFER
42F0 D3F4 OUT USDAO ;AND OUTPUT IT
42F2 04 INR B ;B WAS 00 IF DONE
42F3 05 DCR B
42F4 C27E42 JNZ VOUT ;NOT DONE-EXIT FROM SERVICE ROUTINE
42F7 217E42 LXI H,VOUT ;SET UP RETURN ADDRESS
42FA E5 PUSH H ;AND PUSH IT INTO THE STACK
42FB 3E00 MVI A,GSTAT ;A GETS GOOD STATUS
42FD 2A3942 LHLD TCBA ;POINT H,L AT COMMAND BLOCK
4300 CD5B42 CALL CLEAN ;CALL CLEANUP ROUTINE
4303 E9 PCHL ;CALL COMPLETION ROUTINE
;RETURN WILL BE TO VOUT
;RECEIVER OFF
4304 FE01 TUTE: CPI 01
4306 CA2443 JZ TUTE1
4309 FE02 CPI 02 ;RECEIVER ON
430B CA1443 JZ TUTE2
430E FE03 CPI 03 ;CLEAR ERRORS
4310 CA1C43 JZ TUTE3
4313 C9 RET
4314 3A3742 TUTE2: LDA LCMD
4317 F604 ORI 04
4319 323742 STA LCMD
431C 3A3742 TUTE3: LDA LCMD
431F F610 ORI 10H
4321 D3F5 TUTE4: OUT USCMD
4323 C9 RET
4324 3A3742 TUTE1: LDA LCMD
4327 E6FB ANI OFBH
4329 323742 STA LCMD
432C C32143 JMP TUTE4

```

# APPLICATIONS

;\*\*\*\*\*

;

;

;

;

;

;

;\*\*\*\*\*

USART COMMAND BLOCK INTERPRETER  
 USRUN IS CALLED BY USER WITH THE ADDRESS  
 OF THE COMMAND BLOCK IN H,L. USRUN EXAMINES  
 THE BLOCK AND INITIALIZES THE REQUESTED OPERATION

```

432F 1A      USRUN:  LDAX   D      ;GET THE CMD FROM THE BLOCK
4330 FE43    CPI     'C'    ;IS IT A CLEAR COMMAND?
4332 CA4043  JZ      UCLEAR ;YES GO TO CLEAR ROUTINE
4335 FE52    CPI     'R'    ;IS IT A READ COMMAND?
4337 CA5D43  JZ      UREAD  ;YES-GO TO READ ROUTINE
433A FE57    CPI     'W'    ;IS IT A WRITE COMMAND?
433C CA9D43  JZ      UWRITE ;GO TO WRITE ROUTINE
433F C9      RET      ;NOT A GOOD COMMAND-RETURN
4340 F3      UCLEAR: DI     ;DISABLE INTERRUPTS
4341 AF      XRA     A      ;CLEAR A
4342 D3F5    OUT     USCMD  ;OUTPUT THREE TIMES TO ENSURE
4344 D3F5    OUT     USCMD  ;THAT THE USART IS IN A KNOWN STATE
4346 D3F5    OUT     USCMD
4348 3E40    MVI     A,40H   ;CODE TO RESET USART
434A D3F5    OUT     USCMD  ;OUTPUT ON CMD CHANNEL
434C 3E5E    MVI     A,05EH  ;CE IMPLIES ASYN MODE (X16)
                    ;          8 DATA BITS
                    ;          ODD PARITY
                    ;          1 STOP BIT
434E D3F5    OUT     USCMD  ;OUTPUT ON CMD CHANNEL
4350 AF      XRA     A      ;CLEAR A, SET ZERO
4351 213942  LXI     H,TCBA  ;CLEAR TCBA AND RCBA
4354 77      MOV     M,A
4355 23      INX     H
4356 77      MOV     M,A
4357 23      INX     H
4358 77      MOV     M,A
4359 23      INX     H
435A 77      MOV     M,A
435B FB      EI         ;ENABLE INTERRUPTS
435C C9      RET      ;AND RETURN TO USER
                    ;
                    ;
435D 213B42  UREAD:  LXI     H,RCBA  ;CHECK READ IDLE
4360 7E      MOV     A,M
4361 B7      ORA     A
4362 C26B43  JNZ     UROUT
4365 23      INX     H
4366 7E      MOV     A,M
4367 B7      ORA     A
4368 CA7743  JZ      URDA     ;READ IS IDLE-PROCEDE
436B 3EFE    UROUT:  MVI     A,0FEH ;ALREADY RUNNING-ERROR STATUS
436D 217643  LXI     H,URDB  ;SET UP RETURN ADDRESS
4370 E5      PUSH    H      ;PUSH IT INTO STACK
4371 EB      XCHG   ;H GETS COMMAND BLOCK ADDRESS
4372 CD5B42  CALL    CLEAN   ;CALL CLEANUP ROUTINE
4375 E9      PCHL   ;EFFECTIVELY CALLS END ROUTINE
4376 C9      URDB:  RET      ;RETURN TO USER

4377 EB      URDA:  XCHG   ;H GETS COMMAND BLOCK ADDRESS
4378 223B42  SHLD   RCBA    ;RCBA GETS COMMAND BLOCK ADDRESS
437B 3A3742  LDA     LCMD   ;GET LAST COMMAND
437E F616    ORI     16H   ;SET RXE AND DTR AND RESET ERRORS
4380 323742  STA     LCMD   ;AND RETURN TO MEMORY
4383 0F      RRC     ;SET CARRY EQUAL TO TXE
  
```

# APPLICATIONS

```

4384 D28C43      JNC      URDC
4387 3E02        MVI      A,2
4389 323842      STA      TCMD
438C 07          URDC:   RLC
438D D3F5        OUT      USCMD ;OUTPUT CMD
438F DBF4        IN       USDAI ;CLEAR USART OF LEFT OVER CHARACTERS
4391 DBF4        IN       USDAI
4393 3E82        MVI      A,82H ;MDS-CLEAR RECEIVE INTERRUPT
4395 D3F3        OUT      OF3H ;MDS
4397 3EF6        MVI      A,OF6H ;MDS-ENABLE LEVEL THREE
4399 D3FC        OUT      OFCH ;MDS
439B FB         EI       ;ENABLE INTERRUPTS
439C C9         RET      ;RETURN TO USER

```

```

439D 213942      UWRITE: LXI     H,TCBA ;CHECK WRITE IDLE
43A0 7E         MOV     A,M
43A1 B7         ORA     A
43A2 C26B43      JNZ     UROUT ;BUSY-EXIT
43A5 23         INX     H
43A6 7E         MOV     A,M
43A7 C26B43      JNZ     UROUT ;BUSY-EXIT
43AA EB         XCHG   ;OK-H GETS COMMAND BLOCK ADDRESS
43AB 223942      SHLD   TCBA ;TCBA GETS COMMAND BLOCK ADDRESS
43AE 3A3742      LDA     LCMD ;GET LAST COMMAND
43B1 F623        ORI     023H ;SET RTS,DTR, AND TXEN
43B3 323742      STA     LCMD
43B6 D3F5        OUT     USCMD
43B8 3EF6        MVI     A,OF6H ;MDS-ENABLE LEVEL THREE INTERRUPTS
43BA D3FC        OUT     OFCH ;MDS
43BC FB         EI     ;ENABLE SYSTEM INTERRUPTS
43BD C9         RET     ;AND RETURN

```

# APPLICATIONS

```

;*****
;
;      USER IS A TEST PROGRAM WHICH EXERCISES USRUN
;
;*****

43BE 3EC3      USER:   MVI      A,0C3H ;MDS-SET INTERRUPT VECTOR
43C0 321800    STA      018H
43C3 216842    LXI      H,VECTOR
43C6 221900    SHLD     019H
43C9 3E43      MVI      A,'C' ;SET GENERAL BLOCK TO A 'C'
43CB 111442    LXI      D,GBLOCK
43CE 12        STAX     D
43CF CD2F43    CALL    USRUN
43D2 210040    LXI      H,BUFIN ;CLEAR INPUT BUFFER
43D5 AF        XRA      A
43D6 77        MOV      M,A
43D7 2C        INR      L
43D8 C2D643    JNZ      $-2
43DB 210041    LXI      H,BUFOUT ;INITIALIZE OUTPUT BUFFER
43DE 75        MOV      M,L
43DF 2C        INR      L
43E0 C2DE43    JNZ      $-2
43E3 65        MOV      H,L ;REINTIALIZE CONTROL BLOCKS
43E4 2E52      MVI      L,'R'
43E6 220042    SHLD     RBLOCK
43E9 2E57      MVI      L,'W'
43EB 220A42    SHLD     TBLOCK
43EE 6C        MOV      L,H
43EF 220642    SHLD     RCCT
43F2 221042    SHLD     TCCT
43F5 110042    LXI      D,RBLOCK ;START READ
43F8 CD2F43    CALL    USRUN
43FB 110A42    LXI      D,TBLOCK ;START WRITE
43FE CD2F43    CALL    USRUN
4401 3EFF      MVI      A,OFFH ;LOOP WAITING COMPLETION
4403 321642    STA      FLAG ;FLAG WILL BE SET BY COMPLETION ROUTINES
4406 3A1642    LDA      FLAG
4409 B7        ORA      A
440A C20644    JNZ      $-4
440D 210040    LXI      H,BUFIN ;TEST INPUT BUFFER=OUTPUT BUFFER
4410 7E        COMLP:  MOV     A,M
4411 24        INR      H
4412 BE        CMP      M
4413 C21E44    JNZ      COMER
4416 25        DCR      H
4417 2C        INR      L
4418 C21044    JNZ      COMLP
441B C3BE43    JMP      USER ;GOOD COMPARE-REPEAT TEST
441E C7        COMER:  RST      0 ;ERROR-RETURN TO MONITOR

0000          END

```

# APPLICATIONS

---

BSTAT 00FF	BUFIN 4000	BUFOU 4100	CEND 0001
CLEAN 425B	COMER 441E	COMLP 4410	EXA 42C1
EXCHA 42BE	FLAG 4216	GBLOC 4214	GSTAT 0000
LCMD 4237	LOADA 423E	MTAB 423D	PEND 42CF
RBAD 4202	RBLOC 4200	RCBA 423B	RCCT 4206
RCR 4217	RCRA 4208	RISR 4288	RISRA 42AE
RISRB 4299	RISRE 42B9	RRCT 4204	TBAD 420C
TBLOC 420A	TCBA 4239	TCCT 4210	TCMD 4238
TCR 4227	TCRA 4212	TISR 42D4	TISRA 42EC
TRCT 420E	TUTE 4304	TUTE1 4324	TUTE2 4314
TUTE3 431C	TUTE4 4321	UCLEA 4340	URDA 4377
URDB 4376	URDC 438C	UREAD 435D	UROUT 436B
USCMD 00F5	USDAI 00F4	USDAO 00F4	USER 43BE
USRUN 432F	USTAT 00F5	UWRIT 439D	VECTO 4268
VOUT 427E			



## APPENDIX A

### 8251 DESIGN HINTS

1. Output of a command to the USART destroys the integrity of a transmission in progress if timed incorrectly.

Sending a command into the USART will overwrite any character which is stored in the buffer waiting for transfer to the parallel-to-serial converter in the device. This can be avoided by waiting for TxRDY to be asserted before sending a command if transmission is taking place. Due to the internal structure of the USART, it is also possible to disturb the transmission if a command is sent while a SYN character is being generated by the device. (The USART generates a SYN if the software fails to respond to TxRDY.) If this occurrence is possible in a system, commands should be transferred only when a positive-going edge is detected on the TxRDY line.

2. RxE only acts as a mask to RxRDY; it does not control the operation of the receiver.

When the receiver is enabled, it is possible for it to already contain one or two characters. These characters should be read and discarded when the RxE bit is first set. Because of these extraneous characters the proper sequence for gaining synchronization is as follows:

1. Disable interrupts
2. Issue a command to enter hunt mode, clear errors, and enable the receiver (EH,ER,RxE=1)
3. Read USART data (it is not necessary to check status)
4. Enable interrupts

The first RxRDY that occurs after the above sequence will indicate that the SYN character or

characters have been detected and the next character has been assembled and is ready to be read.

3. Loss of CTS or dropping TxEnable will immediately clamp the serial output line.

TxEnable and RTS should remain asserted until the transmission is complete. Note that this implies that not only has the USART completed the transfer of all bits of the last character, but also that they have cleared the modem. A delay of 1 msec following a proper occurrence of TxEmpty is usually sufficient (see item 4). An additional problem can occur in the synchronous mode because the loss of TxEnable clamps the data in at a SPACE instead of the normal MARK. This problem, which does not occur in the asynchronous mode, can be corrected by an external gate combining RTS and the serial output data.

4. Extraneous transitions can occur on TxEmpty while data (including USART generated SYNs) is transferred to the parallel-to-serial converter.

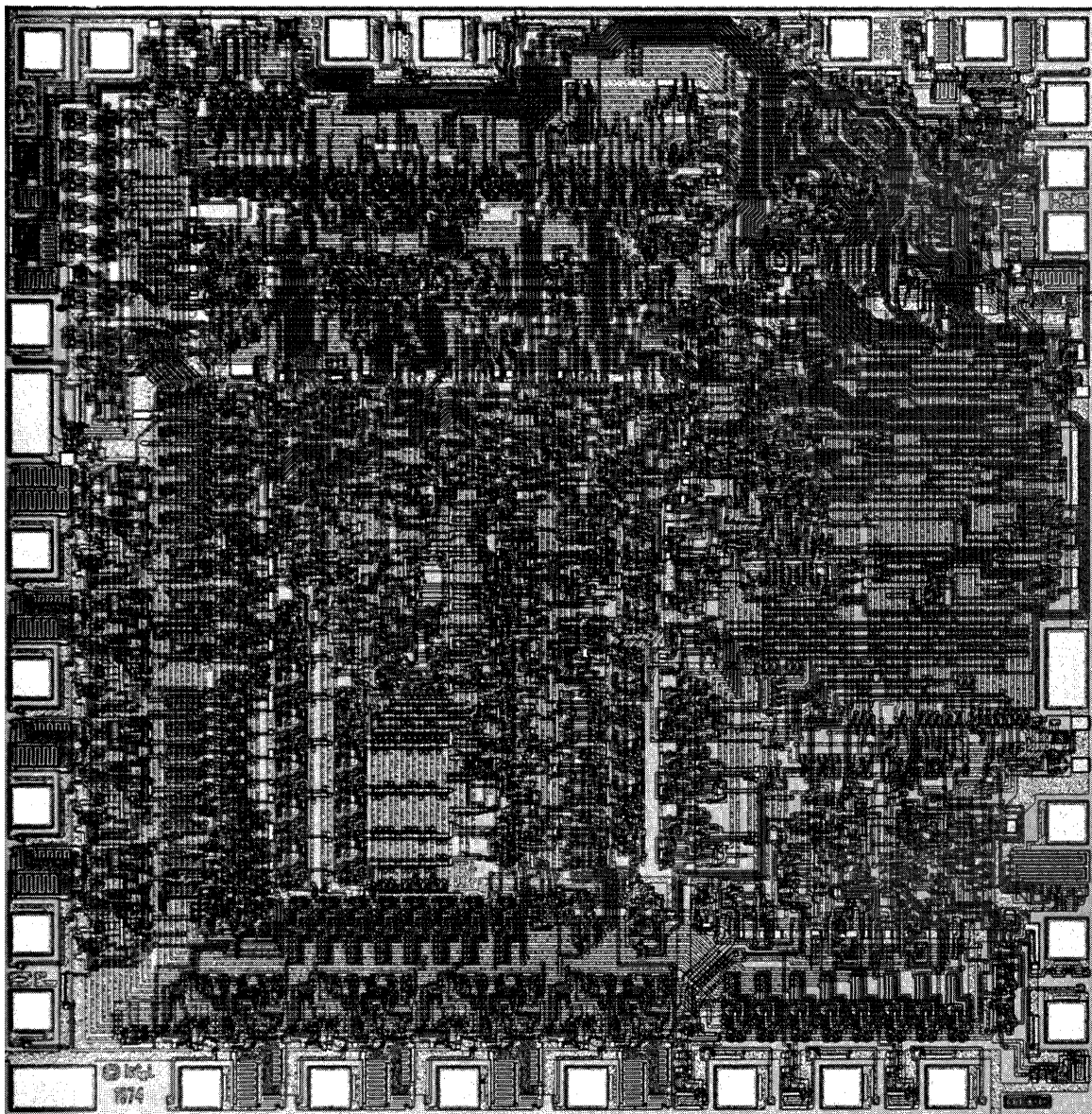
This situation can be avoided by ensuring that TxEmpty occurs during several consecutive status reads before assuming that the transmitter is truly in the empty state.

5. A BREAK (i.e., long space) detected by the receiver results in a string of characters which have framing errors.

If reception is to be continued after a BREAK, care must be taken to ensure that valid data is being received; special care must be taken with the last character perceived during a BREAK, since its value, including any framing error associated with it, is indeterminate.

# 8251 PROGRAMMABLE COMMUNICATION INTERFACE

---



---

# Using the 8273 SDLC/HDLC Protocol Controller

## Contents

<b>INTRODUCTION</b>	<b>2-272</b>
<b>SDLC/HDLC OVERVIEW</b>	<b>2-272</b>
<b>BASIC 8273 OPERATION</b>	<b>2-275</b>
<b>HARDWARE ASPECTS OF THE 8273</b>	<b>2-275</b>
CPU Interface	
Modem Interface	
<b>SOFTWARE ASPECTS OF THE 8273</b>	<b>2-281</b>
Command Phase Software	
Execution Phase Software	
Result Phase Software	
<b>8273 COMMAND DESCRIPTION</b>	<b>2-284</b>
Initialization/Configuration Commands	
Operating Mode Register	
Serial I/O Mode Register	
Data Transfer Mode Register	
One Bit Delay Register	
Receive Commands	
General Receive	
Selective Receive	
Selective Loop Receive	
Receive Disable	
Transmit Commands	
Transmit Frame	
Loop Transmit	
Transmit Transparent	
Abort Commands	
Reset Commands	
Modem Control Commands	
<b>HDLC CONSIDERATIONS</b>	<b>2-289</b>
<b>LOOP CONFIGURATION</b>	<b>2-290</b>
<b>APPLICATION EXAMPLE</b>	<b>2-294</b>
<b>CONCLUSION</b>	<b>2-299</b>
<b>APPENDIX</b>	<b>2-300</b>

# APPLICATIONS

## INTRODUCTION

The Intel 8273 is a Data Communications Protocol Controller designed for use in systems utilizing either SDLC or HDLC (Synchronous or High-Level Data Link Control) protocols. In addition to the usual features such as full duplex operation, automatic Frame Check Sequence generation and checking, automatic zero bit insertion and deletion, and TTL compatibility found on other single component SDLC controllers; the 8273 features a frame level command structure, a digital phase locked loop, SDLC loop operation, and diagnostics.

The frame level command structure is made possible by the 8273's unique internal dual processor architecture. A high-speed bit processor handles the serial data manipulations and character recognition. A byte processor implements the frame level commands. These dual processors allow the 8273 to control the necessary byte-by-byte operation of the data channel with a minimum of CPU (Central Processing Unit) intervention. For the user this means the CPU has time to take on additional tasks. The digital phase locked loop (DPLL) provides a means of clock recovery from the received data stream on-chip. This feature, along with the frame level commands, makes SDLC loop operation extremely simple and flexible. Diagnostics in the form of both data and clock loopback are available to simplify board debug and link testing. The 8273 is a dedicated function peripheral in the MCS-80/85 Microcomputer family and as such, it interfaces to the 8080/8085 system with a minimum of external hardware.

This application note explains the 8273 as a component and shows its use in a generalized loop configuration and a typical 8085 system. The 8085 system was used to verify the SDLC operation of the 8273 on an actual IBM SDLC data communications link.

The first section of this application note presents an overview of the SDLC/HDLC protocols. It is fairly tutorial in nature and may be skipped by the more knowledgeable reader. The second section describes the 8273 from a functional standpoint with explanation of the block diagram. The software aspects of the 8273, including command examples, are discussed in the third section. The fourth and fifth sections discuss a loop SDLC configuration and the 8085 system respectively.

## SDLC/HDLC OVERVIEW

SDLC is a protocol for managing the flow of information on a data communications link. In other words, SDLC can be thought of as an envelope — addressed, stamped, and containing an s.a.s.e. — in which information is transferred from location to location on a data communications link. (Please note that while SDLC is discussed specifically, all comments also apply to HDLC except where noted.) The link may be either point-to-point or multi-point, with the point-to-point configuration being either switched or nonswitched. The information flow may use either full or half duplex exchanges. With this many configurations supported, it is difficult to find a synchronous data communications application where SDLC would not be appropriate.

Aside from supporting a large number of configurations, SDLC offers the potential of a 2x increase in throughput over the presently most prevalent protocol: Bi-Sync. This performance increase is primarily due to two characteristics of SDLC: full duplex operation and the implied acknowledgement of transferred information. The performance increase due to full duplex operation is fairly obvious since, in SDLC, both stations can communicate simultaneously. Bi-Sync supports only half-duplex (two-way alternate) communication. The increase from implied acknowledgement arises from the fact that a station using SDLC may acknowledge previously received information while transmitting different information. Up to 7 messages may be outstanding before an acknowledgement is required. These messages may be acknowledged as a block rather than singly. In Bi-Sync, acknowledgements are unique messages that may not be included with messages containing information and each information message requires a separate acknowledgement. Thus the line efficiency of SDLC is superior to Bi-Sync. On a higher level, the potential of a 2x increase in performance means lower cost per unit of information transferred. Notice that the increase is not due to higher data link speeds (SDLC is actually speed independent), but simply through better line utilization.

Getting down to the more salient characteristics of SDLC; the basic unit of information on an SDLC link is that of the frame. The frame format is shown in Figure 1. Five fields comprise each frame: flag, address, control, information, and frame check sequence. The flag fields (F) form the boundary of the frame and all other fields are positionally related to one of the two flags. All frames start with an opening flag and end with a closing flag. Flags are used for frame synchronization. They also may serve as time-fill characters between frames. (There are no intraframe time-fill characters in SDLC as there are in Bi-Sync.) The opening flag serves as a reference point for the address (A) and control (C) fields. The frame check sequence (FCS) is referenced from the closing flag. All flags have the binary configuration 01111110 (7EH).

SDLC is a bit-oriented protocol, that is, the receiving station must be able to recognize a flag (or any other special character) at any time, not just on an 8-bit boundary. This, of course, implies that a frame may be N-bits in length. (The vast majority of applications tend to use frames which are multiples of 8 bits long, however.)

OPENING FLAG	ADDRESS FIELD (A)	CONTROL FIELD (C)	INFORMATION FIELD (I)	FRAME CHECK SEQUENCE (FCS)	CLOSING FLAG
0 1 1 1 1 1 1 0	8 BITS	8 BITS	ANY LENGTH 0 TO N BITS	16 BITS	0 1 1 1 1 1 1 0

Figure 1. SDLC Frame Format

The fact that the flag has a unique binary pattern would seem to limit the contents of the frame since a flag pattern might inadvertently occur within the frame. This would cause the receiver to think the closing flag was received, invalidating the frame. SDLC handles this situation through a technique called zero bit insertion. This technique specifies that within a frame a binary 0 be inserted by the transmitter after any succession of five contiguous binary 1s. Thus, no pattern of 01111110 is ever transmitted by chance. On the receiving end, after the opening flag is detected, the receiver removes any 0 following 5 consecutive 1s. The inserted and deleted 0s are not counted for error determination.

Before discussing the address field, an explanation of the roles of an SDLC station is in order. SDLC specifies two types of stations: primary and secondary. The primary is the control station for the data link and thus has responsibility of the overall network. There is only one predetermined primary station, all other stations on the link assume the secondary station role. In general, a secondary station speaks only when spoken to. In other words, the primary polls the secondaries for responses. In order to specify a specific secondary, each secondary is assigned a unique 8-bit address. It is this address that is used in the frame's address field.

When the primary transmits a frame to a specific secondary, the address field contains the secondary's address. When responding, the secondary uses its own address in the address field. The primary is never identified. This ensures that the primary knows which of many secondaries is responding since the primary may have many messages outstanding at various secondary stations. In addition to the specific secondary address, an address common to all secondaries may be used for various purposes. (An all 1s address field is usually used for this "All Parties" address.) Even though the primary may use this common address, the secondaries are expected to respond with their unique address. The address field is always the first 8 bits following the opening flag.

The 8 bits following the address field form the control field. The control field embodies the link-level control of SDLC. A detailed explanation of the commands and responses contained in this field is beyond the scope of this application note. Suffice it to say that it is in the control field that the implied acknowledgement is carried out through the use of frame sequence numbers. None of the currently available SDLC single chip controllers utilize the control field. They simply pass it to the processor for analysis. Readers wishing a more detailed explanation of the control field, or of SDLC in general, should consult the IBM documents referenced on the front page overleaf.

In some types of frames, an information field follows the control field. Frames used strictly for link management may or may not contain one. When an information field is used, it is unrestricted in both content and length. This code transparency is made possible because of the zero bit insertion mentioned earlier and the bit-oriented nature of SDLC. Even main memory core dumps may be transmitted because of this capability. This feature is unique to bit-oriented protocols. Like the

control field, the information field is not interpreted by the SDLC device; it is merely transferred to and from memory to be operated on and interpreted by the processor.

The final field is the frame check sequence (FCS). The FCS is the 16 bits immediately preceding the closing flag. This 16-bit field is used for error detection through a Cyclic Redundancy Checkword (CRC). The 16-bit transmitted CRC is the complement of the remainder obtained when the A, C, and I fields are "divided" by a generating polynomial. The receiver accumulates the A, C, and I fields and also the FCS into its internal CRC register. At the closing flag, this register contains one particular number for an error-free reception. If this number is not obtained, the frame was received in error and should be discarded. Discarding the frame causes the station to not update its frame sequence numbering. This results in a retransmission after the station sends an acknowledgement from previous frames. [Unlike all other fields, the FCS is transmitted MSB (Most Significant Bit) first. The A, C, and I fields are transmitted LSB (Least Significant Bit) first.] The details of how the FCS is generated and checked is beyond the scope of this application note and since all single component SDLC controllers handle this function automatically, it is usually sufficient to know only that an error has or has not occurred. The IBM documents contain more detailed information for those readers desiring it.

The closing flag terminates the frame. When the closing flag is received, the receiver knows that the preceding 16 bits constitute the FCS and that any bits between the control field and the FCS constitute the information field.

SDLC does not support an interframe time-fill character such as the SYN character in Bi-Sync. If an unusual condition occurs while transmitting, such as data is not available in time from memory or CTS (Clear-to-Send) is lost from the modem, the transmitter aborts the frame by sending an Abort character to notify the receiver to invalidate the frame. The Abort character consists of eight contiguous 1s sent without zero bit insertion. Intraframe time-fill consists of either flags, Abort characters, or any combination of the two.

While the Abort character protects the receiver from transmitted errors, errors introduced by the transmission medium are discovered at the receiver through the FCS check and a check for invalid frames. Invalid frames are those which are not bounded by flags or are too short, that is, less than 32 bits between flags. All invalid frames are ignored by the receiver.

Although SDLC is a synchronous protocol, it provides an optional feature that allows its use on basically asynchronous data links — NRZI (Non-Return-to-Zero-Inverted) coding. NRZI coding specifies that the signal condition does not change for transmitting a binary 1, while a binary 0 causes a change of state. Figure 2 illustrates NRZI coding compared to the normal NRZ. NRZI coding guarantees that an active line will have a transition at least every 5-bit times; long strings of zeroes cause a transition every bit time, while long strings of 1s are broken up by zero bit insertion. Since asynchronous

# APPLICATIONS

operation requires that the receiver sampling clock be derived from the received data, NRZI encoding plus zero bit insertion make the design of clock recovery circuitry easier.

All of the previous discussion has applied to SDLC on either point-to-point or multi-point data networks, SDLC (but not HDLC) also includes specification for a loop configuration. Figure 3 compares these three configurations. IBM uses this loop configuration in its 3650 Retail Store System. It consists of a single loop controller station with one or more down-loop secondary stations. Communications on a loop rely on the secondary stations repeating a received message down loop with a delay of one bit time. The reason for the one bit delay will be evident shortly.

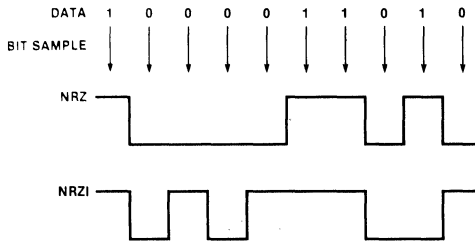


Figure 2. NRZI vs NRZ Encoding

Loop operation defines a new special character: the EOP (End-of-Poll) character which consists of a 0 followed by 7 contiguous, non-zero bit inserted, ones. After the loop controller transmits a message, it idles the line (sends all 1s). The final zero of the closing flag plus the first 7 1s of the idle form an EOP character. While repeating, the secondaries monitor their incoming line for an EOP character. When an EOP is detected, the secondary checks to see if it has a message to transmit. If it does, it changes the seventh 1 to a 0 (the one bit delay allows time for this) and repeats the modified EOP (now alias flag). After this flag is transmitted, the secondary terminates its repeater function and inserts its message (with multiple preceding flags if necessary). After the closing flag, the secondary resumes its one bit delay repeater function. Notice that the final zero of the secondary's closing flag plus the repeated 1s from the controller form an EOP for the next down-loop secondary, allowing it to insert a message if it desires.

One might wonder if the secondary missed any messages from the controller while it was inserting its own message. It does not. Loop operation is basically half-duplex. The controller waits until it receives an EOP before it transmits its next message. The controller's reception of the EOP signifies that the original message has propagated around the loop followed by any messages inserted by the secondaries. Notice that secondaries cannot communicate with one another directly, all secondary-to-secondary communication takes place by way of the controller.

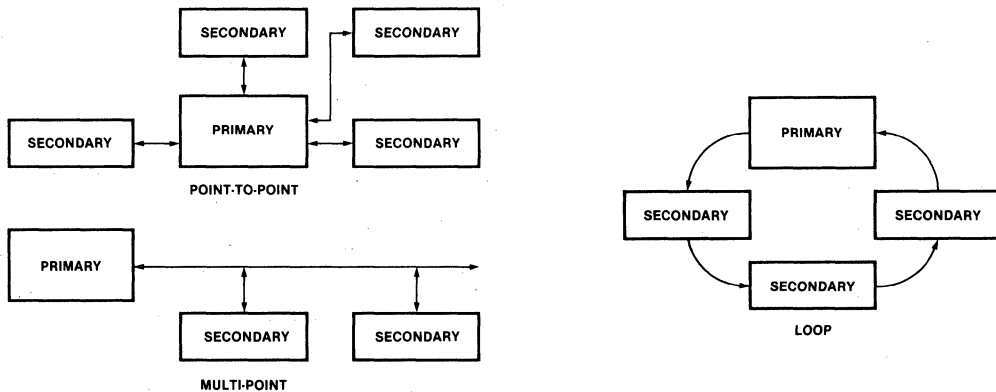


Figure 3. Network Configurations

Loop protocol does not utilize the normal Abort character. Instead, an abort is accomplished by simply transmitting a flag character. Down loop, the receiver sees the abort as a frame which is either too short (if the abort occurred early in the frame) or one with an FCS error. Either results in a discarded frame. For more details on loop operation, please refer to the IBM documents referenced earlier.

Another protocol very similar to SDLC which the 8273 supports is HDLC (High-Level Data Link Control). There are only three basic differences between the two: HDLC offers extended address and control fields, and the HDLC Abort character is 7 contiguous 1s as opposed to SDLC's 8 contiguous 1s.

Extended addressing, beyond the 256 unique addresses possible with SDLC, is provided by using the address field's least significant bit as the extended address modifier. The receiver examines this bit to determine if the octet should be interpreted as the final address octet. As long as the bit is 0, the octet that contains it is considered an extended address. The first time the bit is a 1, the receiver interprets that octet as the final address octet. Thus the address field may be extended to any number of octets. Extended addressing is illustrated in Figure 4a.

A similar technique is used to extend the control field although the extension is limited to only one extra control octet. Figure 4b illustrates control field extension.

Those readers not yet asleep may have noticed the similarity between the SDLC loop EOP character (a 0 followed by 7 1s) and the HDLC Abort (7 1s). This possible incompatibility is neatly handled by the HDLC protocol not specifying a loop configuration.

This completes our brief discussion of the SDLC/HDLC protocols. Now let us turn to the 8273 in particular and discuss its hardware aspects through an explanation of the block diagram and generalized system schematics.

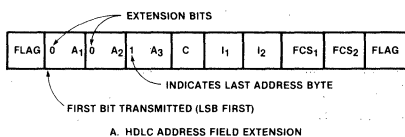


Figure 4a

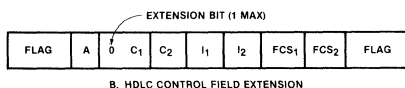


Figure 4b

## BASIC 8273 OPERATION

It will be helpful for the following discussions to have some idea of the basic operation of the 8273. Each operation, whether it is a frame transmission, reception or port read, etc., is comprised of three phases: the Command, Execution, and Result phases. Figure 5 shows the sequence of these phases. As an illustration of this sequence, let us look at the transmit operation.

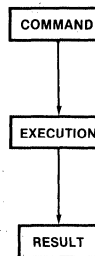


Figure 5. 8273 Operational Phases

When the CPU decides it is time to transmit a frame, the Command phase is entered by the CPU issuing a Transmit Frame command to the 8273. It is not sufficient to just instruct the 8273 to transmit. The frame level command structure sometimes requires more information such as frame length and address and control field content. Once this additional information is supplied, the Command phase is complete and the Execution phase is entered. It is during the Execution phase that the actual operation, in this case a frame transmission, takes place. The 8273 transmits the opening flag, A and C fields, the specified number of I field bytes, inserts the FCS, and closes with the closing flag. Once the closing flag is transmitted, the 8273 leaves the Execution phase and begins the Result phase. During the Result phase the 8273 notifies the CPU of the outcome of the command by supplying interrupt results. In this case, the results would be either that the frame is complete or that some error condition causes the transmission to be aborted. Once the CPU reads all of the results (there is only one for the Transmit Frame command), the Result phase and consequently the operation, is complete. Now that we have a general feeling for the operation of the 8273, let us discuss the 8273 in detail.

## HARDWARE ASPECTS OF THE 8273

The 8273 block diagram is shown in Figure 6. It consists of two major interfaces: the CPU module interface and the modem interface. Let's discuss each interface separately.

# APPLICATIONS

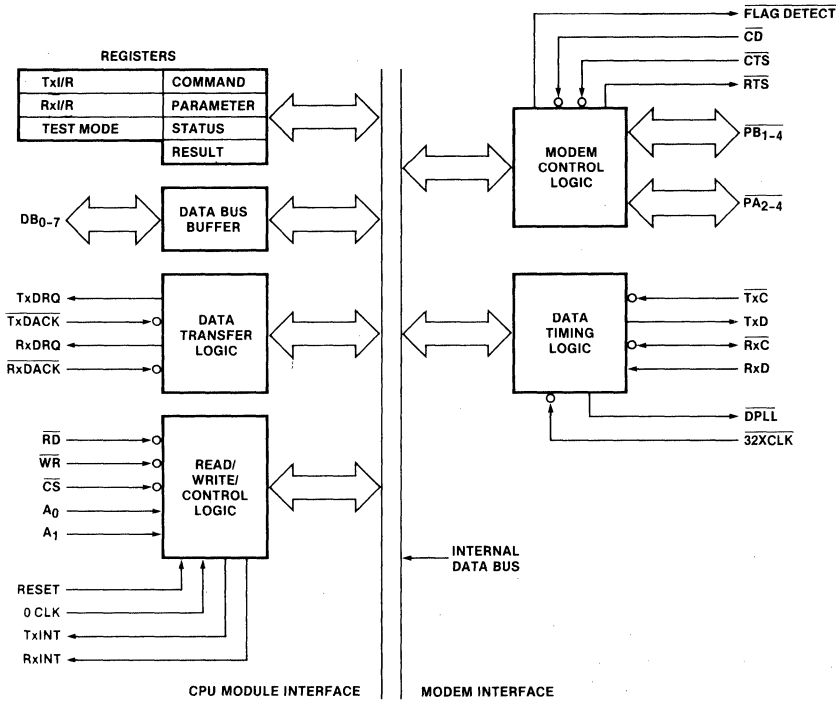


Figure 6. 8273 Block Diagram

## CPU Interface

The CPU interface consists of four major blocks: Control/Read/Write logic (C/R/W), internal registers, data transfer logic, and data bus buffers.

The CPU module utilizes the C/R/W logic to issue commands to the 8273. Once the 8273 receives a command and executes it, it returns the results (good/bad completion) of the command by way of the C/R/W logic. The C/R/W logic is supported by seven registers which are addressed via the  $A_0$ ,  $A_1$ ,  $\overline{RD}$ , and  $\overline{WR}$  signals, in addition to  $\overline{CS}$ . The  $A_0$  and  $A_1$  signals are generally derived from the two low order bits of the CPU module address bus while  $\overline{RD}$  and  $\overline{WR}$  are the normal I/O Read and Write signals found on the system control bus. Figure 7 shows the address of each register using the C/R/W logic. The function of each register is defined as follows:

ADDRESS INPUTS		CONTROL INPUTS	
$A_1$	$A_0$	$\overline{CS} \cdot \overline{RD}$	$\overline{CS} \cdot \overline{WR}$
0	0	STATUS	COMMAND
0	1	RESULT	PARAMETER
1	0	TxI/R	TEST MODE
1	1	RxI/R	—

Figure 7. 8273 Register Selection

**Command** — 8273 operations are initiated by writing the appropriate command byte into this register.

**Parameter** — Many commands require more information than found in the command itself. This additional information is provided by way of the parameter register.

**Immediate Result (Result)** — The completion information (results) for commands which execute immediately are provided in this register.

**Transmit Interrupt Result (TxI/R)** — Results of transmit operations are passed to the CPU in this register.

**Receiver Interrupt Result (RxI/R)** — Receive operation results are passed to the CPU via this register.

**Status** — The general status of the 8273 is provided in this register. The Status register supplies the handshaking necessary during various phases of the 8273 operation.

**Test Mode** — This register provides a software reset function for the 8273.

The commands, parameters, and bit definition of these registers are discussed in the following software section. Notice that there are not specific transmit or receive data registers. This feature is explained in the data transfer logic discussion.



The final elements of the C/R/W logic are the interrupt lines (RxINT and TxINT). These lines notify the CPU module that either the transmitter or the receiver requires service; i.e., results should be read from the appropriate interrupt result register or a data transfer is required. The interrupt request remains active until all the associated interrupt results have been read or the data transfer is performed. Though using the interrupt lines relieves the CPU module of the task of polling the 8273 to check if service is needed, the state of each interrupt line is reflected by a bit in the Status register and non-interrupt driven operation is possible by examining the contents of these bits periodically.

The 8273 supports two independent data interfaces through the data transfer logic; receive data and transmit data. These interfaces are programmable for either DMA or non-DMA data transfers. While the choice of the configuration is up to the system designer, it is based on the intended maximum data rate of the communications channel. Figure 8 illustrates the transfer rate of data bytes that are acquired by the 8273 based on link data rate. Full-duplex data rates above 9600 baud usually require DMA. Slower speeds may or may not require DMA depending on the task load and interrupt response time of the processor.

Figure 9 shows the 8273 in a typical DMA environment. Notice that a separate DMA controller, in this case the Intel 8257, is required. The DMA controller supplies the timing and addresses for the data transfers while the 8273 manages the requesting of transfers and the actual counting of the data block lengths. In this case, elements of the data transfer interface are:

**TxDQ:** *Transmit DMA Request* — Asserted by the 8273, this line requests a DMA transfer from memory to the 8273 for transmit.

**TxDACK:** *Transmit DMA Acknowledge* — Returned by the 8257 in response to TxDQ, this line notifies the 8273 that a request has been granted, and provides access to the transmitter data register.

**RxDQ:** *Receiver DMA Request* — Asserted by the 8273, it requests a DMA transfer from the 8273 to memory for a receive operation.

**RxDACK:** *Receiver DMA Acknowledge* — Returned by the 8257, it notifies the 8273 that a receive DMA cycle has been granted, and provides access to the receiver data register.

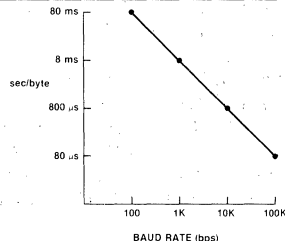
**RD:** *Read* — Supplied by the 8257 to indicate data is to be read from the 8273 and placed in memory.

**WR:** *Write* — Supplied by the 8257 to indicate data is to be written to the 8273 from memory.

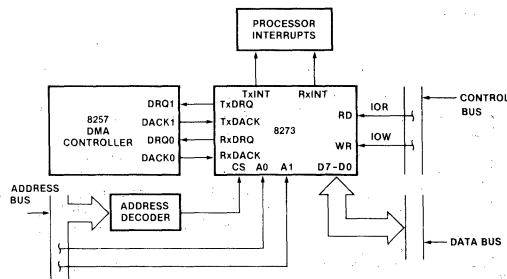
To request a DMA transfer the 8273 raises the appropriate DMA request line; let us assume it is a transmitter request (TxDQ). Once the 8257 obtains control of the system bus by way of its HOLD and HLDA (hold acknowledge) lines, it notifies the 8273 that TxDQ has been granted by returning TxDACK and WR. The TxDACK and WR signals transfer data to the 8273 for a transmit, independent of the 8273 chip select pin (CS). A similar sequence of events occurs for receiver requests. This "hard select" of data into the transmitter or out of

the receiver alleviates the need for the normal transmit and receive data registers addressed by a combination of address lines, CS, and WR or RD. Competitive devices that do not have this "hard select" feature require the use of an external multiplexer to supply the correct inputs for register selection during DMA. (Do not forget that the SDLC controller sees both the addresses and control signals supplied by the DMA controller during DMA cycles.) Let us look at typical frame transmit and frame receive sequences to better see how the 8273 truly manages the DMA data transfer.

Before a frame can be transmitted, the DMA controller is supplied, by the CPU, the starting address for the desired information field. The 8273 is then commanded to transmit a frame. (Just how this is done is covered later during our software discussion.) After the command, but before transmission begins, the 8273 needs a little more information (parameters). Four parameters are required for the transmit frame command: the address field byte, the control field byte, and two bytes which are the least significant and most significant bytes of the information field byte length. Once all four parameters are loaded, the 8273 makes RTS (Request-to-Send) active and waits for CTS (Clear-to-Send) to go active. Once CTS is active, the 8273 starts the frame transmission. While the 8273 is transmitting the opening flag, address field, and control field; it starts making transmitter DMA requests. These requests continue at character (byte) boundaries until the pre-loaded number of bytes of information field have been transmitted. At this point the requests stop, the FCS and closing flag are transmitted, and the TxINT line is raised, signaling the CPU that the frame transmission is complete. Notice that after the initial command and parameter loading, absolutely no CPU intervention was required (since DMA is used for data transfers) until the entire frame was transmitted. Now let's look at a frame reception.



**Figure 8. Byte Transfer Rate vs Baud Rate**



**Figure 9. DMA, Interrupt-Driven System**

# APPLICATIONS

The receiver operation is very similar. Like the initial transmit sequence, the DMA controller is loaded with a starting address for a receiver data buffer and the 8273 is commanded to receive. Unlike the transmitter, there are two different receive commands: General Receive, where all received frames are transferred to memory, and Selective Receive, where only frames having an address field matching one of two preprogrammed 8273 address fields are transferred to memory. Let's assume for now that we want to general receive. After the receive command, two parameters are required before the receiver becomes active: the least significant and most significant bytes of the receiver buffer length. Once these bytes are loaded, the receiver is active and the CPU may return to other tasks. The next frame appearing at the receiver input is transferred to memory using receiver DMA requests. When the closing flag is received, the 8273 checks the FCS and raises its RxINT line. The CPU can then read the results which indicate if the frame was error-free or not. (If the received frame had been longer than the pre-loaded buffer length, the CPU would have been notified of that occurrence earlier with a receiver error interrupt. The command description section contains a complete list of error conditions.) Like the transmit example, after the initial command, the CPU is free for other tasks until a frame is completely received. These examples have illustrated the 8273's management of both the receiver and transmitter DMA channels.

It is possible to use the DMA data transfer interface in a non-DMA interrupt-driven environment. In this case, 4 interrupt levels are used: one each for TxINT and RxINT, and one each for TxDRQ and RxDRQ. This configuration is shown in Figure 10. This configuration offers the advantages that no DMA controller is required and data requests are still separated from result (completion) requests. The disadvantages of the configuration are that 4 interrupt levels are required and that the CPU must actually supply the data transfers. This, of course, reduces the maximum data rate compared to the configuration based strictly on DMA. This system could use an Intel 8259 8-level Priority Interrupt Controller to supply a vectored CALL (subroutine) address based on requests on its inputs. The 8273 transmitter and receiver make data requests by raising the respective DRQ line. The CPU is interrupted by the 8259 and vectored to a data transfer routine. This routine either writes (for transmit) or reads (for receive) the 8273 using the respective TxDACK or RxDACK line. As in the case above, the DACK lines serve as "hard" chip selects into and out of the 8273. (TxDACK + WR writes data into the 8273 for transmit. RxDACK + RD reads data from the 8273 for receive.) The CPU is notified of operation completion and results by way of TxINT and RxINT lines. Using the 8273, and the 8259, in this way, provides a very effective, yet simple, interrupt-driven interface.

Figure 11 illustrates a system very similar to that described above. This system utilizes the 8273 in a non-DMA data transfer mode as opposed to the two DMA approaches shown in Figures 9 and 10. In the non-DMA case, data transfer requests are made on the TxINT and RxINT lines. The DRQ lines are not used. Data transfer requests are separated from result requests by a bit in

the Status register. Thus, in response to an interrupt, the CPU reads the Status register and branches to either a result or a data transfer routine based on the status of one bit. As before, data transfers are made via using the DACK lines as chip selects to the transmitter and receiver data registers.

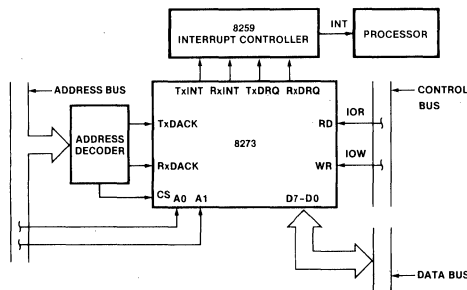


Figure 10. Interrupt-Based DMA System

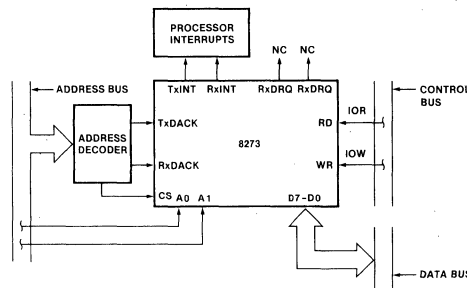


Figure 11. Non-DMA Interrupt-Driven System

Figure 12 illustrates the simplest system of all. This system utilizes polling for all data transfers and results. Since the interrupt pins are reflected in bits in the Status register, the software can read the Status register periodically looking for one of these to be set. If it finds an INT bit set, the appropriate Result Available bit is examined to determine if the "interrupt" is a data transfer or completion result. If a data transfer is called for, the DACK line is used to enter or read the data from the 8273. If the interrupt is a completion result, the appropriate result register is read to determine the good/bad completion of the operation.

The actual selection of either DMA or non-DMA modes is controlled by a command issued during initialization. This command is covered in detail during the software discussion.

# APPLICATIONS

The final block of the CPU module interface is the Data Bus Buffer. This block supplies the tri-state, bidirectional data bus interface to allow communication to and from the 8273.

## Modem Interface

As the name implies, the modem interface is the modem side of the 8273. It consists of two major blocks: the modem control block and the serial data timing block.

The modem control block provides both dedicated and user-defined modem control functions. All signals supported by this interface are active low so that EIA inverting drivers (MC1488) and inverting receivers (MC1489) may be used to interface to standard modems.

Port A is a modem control input port. Its representation on the data bus is shown in Figure 13. Bits  $D_0$  and  $D_1$  have dedicated functions.  $D_0$  reflects the logical state of the CTS (Clear-to-Send) pin. [If CTS is active (low),  $D_0$  is a 1.] This signal is used to condition the start of a transmission. The 8273 waits until CTS is active before it starts transmitting a frame. While transmitting, if CTS goes inactive, the frame is aborted and the CPU is interrupted. When the CPU reads the interrupt result, a CTS failure is indicated.

$D_1$  reflects the logical state of the CD (Carrier Detect) pin. CD is used to condition the start of a frame reception. CD must be active in time for a frame's address field. If CD is lost (goes inactive) while receiving a frame, an interrupt is generated with a CD failure result. CD may go inactive between frames.

Bits  $D_2$  thru  $D_4$  reflect the logical state of the  $\overline{PA}_2$  thru  $\overline{PA}_4$  pins respectively. These inputs are user defined. The 8273 does not interrogate or manipulate these bits. Bits  $D_5$ ,  $D_6$ , and  $D_7$  are not used and each is read as a 1 for a Read Port A command.

Port B is a modem control output port. Its data bus representation is shown in Figure 14. As in Port A, the bit values represent the logical condition of the pins.  $D_0$  and  $D_5$  are dedicated function outputs.  $D_0$  represents the RTS (Request-to-Send) pin. RTS is normally used to notify the modem that the 8273 wishes to transmit. This function is handled automatically by the 8273. If RTS is inactive (pin is high) when the 8273 is commanded to transmit, the 8273 makes it active and then waits for CTS before transmitting the frame. One byte time after the end of the frame, the 8273 returns RTS to its inactive state. However, if RTS was active when a transmit command is issued, the 8273 leaves it active when the frame is complete.

Bit  $D_5$  reflects the state of the Flag Detect pin. This pin is activated whenever an active receiver sees a flag character. This function is useful to activate a timer for line activity timeout purposes.

Bits  $D_1$  thru  $D_4$  provide four user-defined outputs. Pins  $\overline{PB}_1$  thru  $\overline{PB}_4$  reflect the logical state of these bits. The 8273 does not interrogate or manipulate these bits.  $D_6$  and  $D_7$  are not used. In addition to being able to output to Port B, Port B may be read using a Read Port B command. All Modem control output pins are forced high on

reset. (All commands mentioned in this section are covered in detail later.)

The final block to be covered is the serial data timing block. This block contains two sections: the serial data logic and the digital phase locked loop (DPLL).

Elements of the serial data logic section are the data pins, Tx $\overline{D}$  (transmit data output) and Rx $\overline{D}$  (receive data input), and the respective data clocks, Tx $\overline{C}$  and Rx $\overline{C}$ . The transmit and receive data is synchronized by the Tx $\overline{C}$  and Rx $\overline{C}$  clocks. Figure 15 shows the timing for these signals. The leading edge (negative transition) of Tx $\overline{C}$  generates new transmit data and the trailing edge (positive transition) of Rx $\overline{C}$  is used to capture the receive data.

It is possible to reconfigure this section under program control to perform diagnostic functions; both data and clock loopback are available. In data loopback mode, the Tx $\overline{D}$  pin is internally routed to the Rx $\overline{D}$  pin. This allows simple board checkout since the CPU can send an SDLC message to itself. (Note that transmitted data will still appear on the Tx $\overline{D}$  pin.)

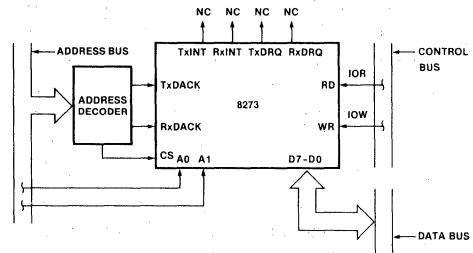


Figure 12. Polled System

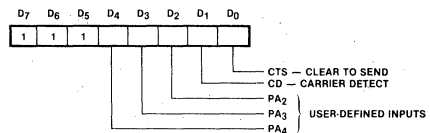


Figure 13. Port A (Input) Bit Definition

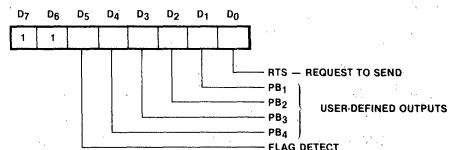


Figure 14. Port B (Output) Bit Definition

# APPLICATIONS

When data loopback is utilized, the receiver may be presented incorrect sample timing ( $\overline{\text{Rx}}\text{C}$ ) by the external circuitry. Clock loopback overcomes this problem by allowing the internal routing of  $\overline{\text{Tx}}\text{C}$  and  $\overline{\text{Rx}}\text{C}$ . Thus the same clock used to transmit the data is used to receive it. Examination of Figure 15 shows that this method ensures bit synchronism. The final element of the serial data logic is the Digital Phase Locked Loop.

The DPLL provides a means of clock recovery from the received data stream. This feature allows the 8273 to interface without external synchronizing logic to low cost asynchronous modems (modems which do not supply clocks). It also makes the problem of clock timing in loop configurations trivial.

To use the DPLL, a clock at 32 times the required baud rate must be supplied to the  $32 \times \text{CLK}$  pin. This clock provides the interval that the DPLL samples the received data. The DPLL uses the  $32 \times$  clock and the received data to generate a pulse at the DPLL output pin. This DPLL pulse is positioned at the nominal center of the received data bit cell. Thus the DPLL output may be wired to  $\overline{\text{Rx}}\text{C}$  and/or  $\overline{\text{Tx}}\text{C}$  to supply the data timing. The exact position of the pulse is varied depending on the line noise and bit distortion of the received data. The adjustment of the DPLL position is determined according to the rules outlined in Figure 16.

Adjustments to the sample phase of DPLL with respect to the received data is made in discrete increments. Referring to Figure 16, following the occurrence of DPLL pulse A, the DPLL counts  $32 \times \text{CLK}$  pulses and examines the received data for a data edge. Should no edge be detected in 32 pulses, the DPLL positions the next DPLL pulse (B) at 32 clock pulses from pulse A. Since no new phase information is contained in the data stream, the sample phase is assumed to be at nominal  $1 \times$  baud rate. Now assume a data edge occurs after

DPLL pulse B. The distance from B to the next pulse C is influenced according to which quadrant ( $A_1$ ,  $B_1$ ,  $B_2$ , or  $A_2$ ) the data edge falls in. (Each quadrant represents  $8 \times 32 \times \text{CLK}$  times.) For example, if the edge is detected in quadrant  $A_1$ , it is apparent that pulse B was too close to the data edge and the time to the next pulse must be shortened. The adjustment for quadrant  $A_1$  is specified as  $-2$ . Thus, the next DPLL pulse, pulse C, is positioned  $32 - 2$  or  $30 \times 32 \times \text{CLK}$  pulses following DPLL pulse B. This adjustment moves pulse C closer to the nominal bit center of the next received data cell. A data edge occurring in quadrant  $B_2$  would have caused the adjustment to be small, namely  $32 + 1$  or  $33 \times 32 \times \text{CLK}$  pulses. Using this technique, the DPLL pulse converges to the nominal bit center within 12 data transitions, worse case — 4-bit times adjusting through quadrant  $A_1$  or  $A_2$  and 8-bit times adjusting through  $B_1$  or  $B_2$ .

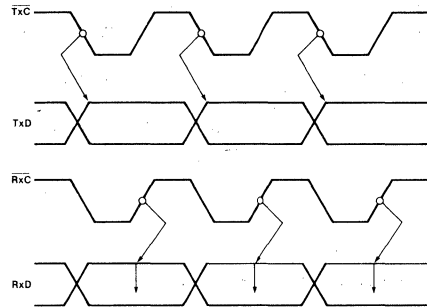


Figure 15. Transmit/Receive Timing

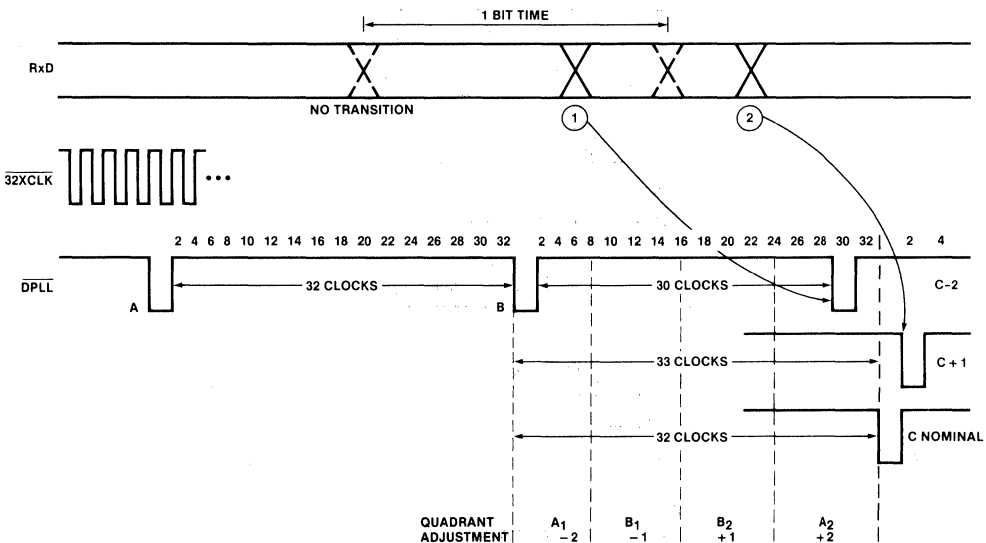


Figure 16. DPLL Phase Adjustments

# APPLICATIONS

When the receive data stream goes idle after 15 ones, DPLL pulses are generated at 32 pulse intervals of the 32x CLK. This feature allows the DPLL pulses to be used as both transmitter and receiver clocks.

In order to guarantee sufficient transitions of the received data to enable the DPLL to lock, NRZI encoding of the data is recommended. This ensures that, within a frame, data transitions occur at least every five bit times — the longest sequence of 1s which may be transmitted with zero bit insertion. It is also recommended that frames following a line idle be transmitted with pre-frame sync characters which provide a minimum of 12 transitions. This ensures that the DPLL is generating DPLL pulses at the nominal bit centers in time for the opening flag. (Two 00H characters meet this requirement by supplying 16 transitions with NRZI encoding. The 8273 contains a mode which supplies such a pre-frame sync.)

Figure 17 illustrates 8273 clock configurations using either synchronous or asynchronous modems. Notice how the DPLL output is used for both Tx̄C and Rx̄C in the asynchronous case. This feature eliminates the need for external clock generation logic where low cost asynchronous modems are used and also allows direct connection of 8273s for the ultimate in low cost data links. The configuration for loop applications is discussed in a following section.

This completes our discussion of the hardware aspects of the 8273. Its software aspects are now discussed.

## SOFTWARE ASPECTS OF THE 8273

The software aspects of the 8273 involve the communication of both commands from the CPU to the 8273 and the return of results of those commands from the 8273

to the CPU. Due to the internal processor architecture of the 8273, this CPU-8273 communication is basically a form of interprocessor communication. Such communication usually requires a form of protocol of its own. This protocol is implemented through use of handshaking supplied in the 8273 Status register. The bit definition of this register is shown in Figure 18.

**CBSY: Command Busy** — CBSY indicates when the 8273 is in the command phase. CBSY is set when the CPU writes a command into the Command register, starting the Command phase. It is reset when the last parameter is deposited in the Parameter register and accepted by the 8273, completing the Command phase.

**CBF: Command Buffer Full** — When set, this bit indicates that a byte is present in the Command register. This bit is normally not used.

**CPBF: Command Parameter Buffer Full** — This bit indicates that the Parameter register contains a parameter. It is set when the CPU deposits a parameter in the Parameter register. It is reset when the 8273 accepts the parameter.

**CRBF: Command Result Buffer Full** — This bit is set when the 8273 places a result from an immediate type command in the Result register. It is reset when the CPU reads the result from the Result register.

**RxINT: Receiver Interrupt** — The state of the RxINT pin is reflected by this bit. RxINT is set by the 8273 whenever the receiver needs servicing. RxINT is reset when the CPU reads the results or performs the data transfer.

**TxINT: Transmitter Interrupt** — This bit is identical to RxINT except action is initiated based on transmitter interrupt sources.

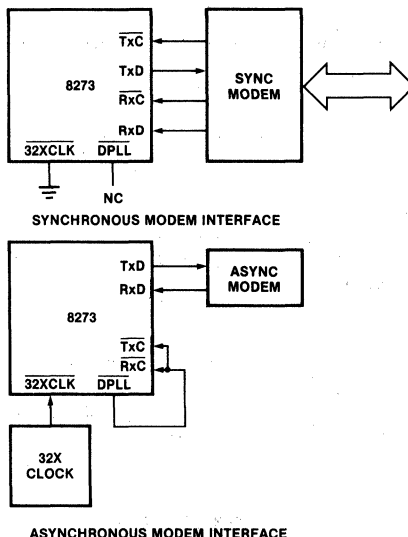


Figure 17. Serial Data Timing Configuration

# APPLICATIONS

**RxIRA: Receiver Interrupt Result Available** — RxIRA is set when the 8273 places an interrupt result byte into the RxI/R register. RxIRA is reset when the CPU reads the RxI/R register.

**TxIRA: Transmitter Interrupt Result Available** — TxIRA is the corresponding Result Available bit for the transmitter. It is set when the 8273 places an interrupt result byte in the TxI/R register and reset when the CPU reads the register.

The significance of each of these bits will be evident shortly. Since the software requirements of each 8273 phase are essentially independent, each phase is covered separately.

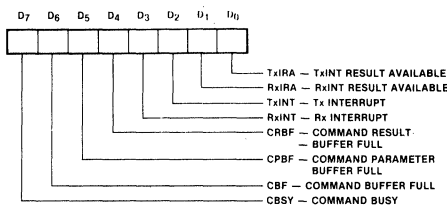


Figure 18. Status Register Format

## Command Phase Software

Recalling the Command phase description in an earlier section, the CPU starts the Command phase by writing a command byte into the 8273 Command register. If further information about the command is required by the 8273, the CPU writes this information into the Parameter register. Figure 19 is a flowchart of the Command phase. Notice that the CBSY and CPBF bits of the Status register are used to handshake the command and parameter bytes. Also note that the chart shows that a command may not be issued if the Status register indicates the 8273 is busy (CBSY = 1). If a command is issued while CBSY = 1, the original command is overwritten and lost. (Remember that CBSY signifies the command phase is in progress and not the actual execution of the command.) The flowchart also includes a Parameter buffer full check. The CPU must wait until CPBF = 0 before writing a parameter to the Parameter register. If a parameter is issued while CPBF = 1, the previous parameter is overwritten and lost. An example of command output assembly language software is provided in Figure 20a. This software assumes that a command buffer exists in memory. The buffer is pointed at by the HL register. Figure 20b shows the command buffer structure.

The 8273 is a full duplex device, i.e., both the transmitter and receiver may be executing commands or passing interrupt results at any given time. (Separate Rx and Tx interrupt pins and result registers are provided for this reason.) However, there is only one Command register. Thus, the Command register must be used for only one command sequence at a time and the transmitter and receiver may never be simultaneously in a command

phase. A detailed description of the commands and their parameters is presented in a following section.

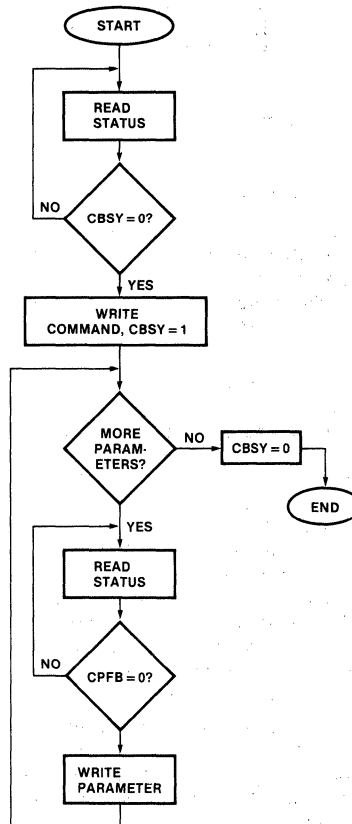
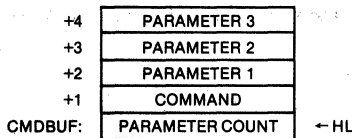


Figure 19. Command Phase Flowchart

```

;FUNCTION: COMMAND DISPATCHER
;INPUTS: HL - COMMAND BUFFER ADDRESS
;OUTPUTS: NONE
;CALLS: NONE
;DESTROYS: A,B,H,L,F/F'S
;DESCRIPTION: CMDOUT ISSUES THE COMMAND + PARAMETERS
;IN THE COMMAND BUFFER POINTED AT BY HL
;
CMDOUT: LXI    H,CMDBUF;POINT HL AT BUFFER
        MOV    B,M    ;1ST ENTRY IS PAR. COUNT
        INX   H      ;POINT AT COMMAND BYTE
CMD1:  IN    STAT73 ;READ 8273 STATUS
        RLC   ;ROTATE CBSY INTO CARRY
        JC    CMD1  ;WAIT UNTIL CBSY=0
        MOV   A,M   ;MOVE COMMAND BYTE TO A
        OUT  COMM73 ;PUT COMMAND IN COMMAND REG
CMD2:  MOV   A,B   ;GET PARAMETER COUNT
        ANA  A    ;TEST IF ZERO
        RZ    ;IF 0 THEN DONE
        INX  H    ;NOT DONE, SO POINT AT NEXT PAR
        DCR  B    ;DEC PARAMETER COUNT
CMD3:  IN    STAT73 ;READ 8273 STATUS
        ANI  CPBF ;TEST CPBF BIT
        JNZ  CMD3  ;WAIT UNTIL CPBF IS 0
        MOV  A,M   ;GET PARAMETER FROM BUFFER
        OUT  PARM73 ;OUTPUT PAR TO PARAMETER REG
        JMP  CMD2  ;CHECK IF MORE PARAMETERS
    
```

Figure 20A. Command Phase Software



**Figure 20B. Command Buffer Format**

### Execution Phase Software

During the Execution phase, the operation specified by the Command phase is performed. If the system utilizes DMA for data transfers, there is no CPU involvement during this phase, so no software is required. If non-DMA data transfers are used, either interrupts or polling is used to signal a data transfer request.

For interrupt-driven transfers the 8273 raises the appropriate INT pin. When responding to the interrupt, the CPU must determine whether it is a data transfer request or an interrupt signaling that an operation is complete and results are available. The CPU determines the cause by reading the Status register and interrogating the associated IRA (Interrupt Result Available) bit (TxIRA for TxINT and RxIRA for RxINT). If the IRA = 0, the interrupt is a data transfer request. If the IRA = 1, an operation is complete and the associated Interrupt Result register must be read to determine the completion status (good/bad/etc.). A software interrupt handler implementing the above sequence is presented as part of the Result phase software.

When polling is used to determine when data transfers are required, the polling routine reads the Status register looking for one of the INT bits to be set. When a set INT bit is found, the corresponding IRA bit is examined. Like in the interrupt-driven case, if the IRA = 0, a data transfer is required. If IRA = 1, an operation is complete and the Interrupt Result register needs to be read. Again, example polling software is presented in the next section.

### Result Phase Software

During the Result phase the 8273 notifies the CPU of the outcome of a command. The Result phase is initiated by either a successful completion of an operation or an error detected during execution. Some commands such as reading or writing the I/O ports provide immediate results, that is, there is essentially no delay from the issuing of the command and when the result is available. Other commands such as frame transmit, take time to complete so their result is not available immediately. Separate result registers are provided to distinguish these two types of commands and to avoid interrupt handling for simple results.

Immediate results are provided in the Result register. Validity of information in this register is indicated to the CPU by way of the CRBF bit in the Status register. When the CPU completes the Command phase of an immediate command, it polls the Status register waiting until CRBF = 1. When this occurs, the CPU may read the

Result register to obtain the immediate result. The Result register provides only the results from immediate commands.

Example software for handling immediate results is shown in Figure 21. The routine returns with the result in the accumulator. The CPU then uses the result as is appropriate.

All non-immediate commands deal with either the transmitter or receiver. Results from these commands are provided in the TxI/R (Transmit Interrupt Result) and RxI/R (Receive Interrupt Result) registers respectively. Results in these registers are conveyed to the CPU by the TxIRA and RxIRA bits of the Status register. Results of non-immediate commands consist of one byte result interrupt code indicating the condition for the interrupt and, if required, one or more bytes supplying additional information. The interrupt codes and the meaning of the additional results are covered following the detailed command description.

Non-immediate results are passed to the CPU in response to either interrupts or polling of the Status register. Figure 22 illustrates an interrupt-driven result handler. (Please note that all of the software presented in this application note is not optimized for either speed or code efficiency. They are provided as a guide and to illustrate concepts.) This handler provides for interrupt-driven data transfers as was promised in the last section. Users employing DMA-based transfers do not need the lines where the IRA bit is tested for zero. (These lines are denoted by an asterisk in the comments column.) Note that the INT bit is used to determine when all results have been read. All results must be read. Otherwise, the INT bit (and pin) will remain high and further interrupts may be missed. These routines place the results in a result buffer pointed at by RCRBUF and TxRBUF.

A typical result handler for systems utilizing polling is shown in Figure 23. Data transfers are also handled by this routine. This routine utilizes the routines of Figure 22 to handle the results.

At this point, the reader should have a good conceptual feel about how the 8273 operates. It is now time for the particulars of each command to be discussed.

```

;FUNCTION: IMDRLT
;INPUTS: NONE
;OUTPUTS: RESULT REGISTER IN A
;CALLS: NONE
;DESTROYS: A, F/F'S
;DESCRIPTION: IMDRLT IS CALLED AFTER A CMDOUT FOR AN
;IMMEDIATE COMMAND TO READ THE RESULT REGISTER
IMDRLT: IN     STAT73 ;READ 8273 STATUS
        ANI    CRBF  ;TEST IF RESULT REG READY
        JZ     IMDRLT ;WAIT IF CRBF=0
        IN     RESL73 ;READ RESULT REGISTER
        RET    ;RETURN
    
```

**Figure 21. Immediate Result Handler**

# APPLICATIONS

```

;FUNCTION: RXI - INTERRUPT DRIVEN RESULT/DATA HANDLER
;INPUTS: RCRBUF, RCVPT
;CALLS: NONE
;OUTPUTS: RCRBUF, RCVPT
;DESTROYS: NOTHING
;DESCRIPTION: RXI IS ENTERED AT A RECEIVER INTERRUPT.
;THE INTERRUPT IS TESTED FOR DATA TRANSFER (IRA=0)
;OR RESULT (IRA=1). FOR DATA TRANSFER, THE DATA IS
;PLACED IN A BUFFER AT RCVPT. RESULTS ARE PLACED IN
;A BUFFER AT RCRBUF.
;A FLAG(RXFLAG) IS SET IF THE INTERRUPT WAS A RESULT.
;(DATA TRANSFER INSTRUCTIONS ARE DENOTED BY (*) AND
;MAYBE ELIMINATED BY USERS USING DMA.
;
RXI:  PUSH  H           ;SAVE HL
      PUSH  PSW        ;SAVE PSW
      PUSH  B           ;SAVE B
      IN   STAT73      ;(*) READ 8273 STATUS
      ANI  RXIRA       ;(*) TEST IRA BIT
      JZ   RXI2        ;(*) IF 0, DATA TRANSFER NEEDED
RXI1:  LHLD  RCRBUF    ;GET RESULT BUFFER POINTER
      IN   STAT73      ;READ 8273 STATUS AGAIN
      ANI  RXINT       ;TEST INT BIT
      JZ   RXI4        ;IF 0, THEN DONE
      IN   STAT73      ;READ 8273 STATUS AGAIN
      ANI  RXIRA       ;TEST IRA AGAIN
      JZ   RXI1        ;LOOP UNTIL RESULT IS READY
      IN   RXIR73     ;READY, READ RXI/R
      MOV  M,A         ;STORE RESULT IN BUFFER
      INX  H           ;BUMP RESULT POINTER
      SHLD RCRBUF     ;RESTORE BUFFER POINTER
      JMP  RXI2        ;GO BACK TO SEE IF MORE
RXI2:  SHLD  RCVPT     ;(*) GET DATA BUFFER POINTER
      IN   RCVDAT     ;(*) READ DATA VIA RXDACK
      MOV  M,A         ;(*) STORE DATA IN BUFFER
      INX  H           ;(*) BUMP DATA POINTER
      JMP  RXI3        ;(*) DONE
RXI4:  MVI  A,01H     ;SET RX FLAG TO SHOW COMPLETION
      STA  RXFLAG     ;COMPLETION
RXI3:  POP  B           ;RESTORE BC
      POP  PSW        ;RESTORE PSW
      POP  H           ;RESTORE HL
      EI             ;ENABLE INTERRUPTS
      RET             ;DONE

```

```

;FUNCTION: TXI - INTERRUPT DRIVEN RESULT/DATA HANDLER
;INPUTS: TXRBUF, TXPNT, TXFLAG
;OUTPUTS: TXRBUF, TXPNT, TXFLAG
;CALLS: NONE
;DESTROYS: NOTHING
;DESCRIPTION: TXI IS ENTERED AT A TRANSMITTER INTERRUPT.
;THE INTERRUPT IS TESTED BY WAY OF THE IRA BIT TO SEE
;IF A DATA TRANSFER OR RESULT COMPLETION HAS OCCURED.
;FOR DATA TRANSFERS (IRA=0), THE DATA IS OBTAINED FROM
;A BUFFER LOCATION POINTED AT BY TXPNT. FOR COMPLETION,
;(IRA=1), THE RESULTS ARE READ AND PLACED AT A RESULT
;BUFFER POINTED AT BY TXRBUF, AND THE TXFLAG IS SET.
;TO INDICATE TO THE MAIN PROGRAM THAT A OPERATION IS
;COMPLETE. TX OPERATIONS HAVE ONLY ONE RESULT.
;DATA TRANSFER INSTRUCTIONS ARE DENOTED BY (*). THESE
;MAYBE REMOVED BY USERS USING DMA.
;
TXI:  PUSH  H           ;SAVE HL
      PUSH  PSW        ;SAVE PSW
      IN   STAT73      ;(*) READ 8273 STATUS
      ANI  TXIRA       ;(*) TEST TXIRA BIT
      JZ   TXI2        ;(*) IF 0, DATA TRANSFER
      IN   TXIR73     ;1, THEN READ TXIR
      LHLD TXRBUF     ;GET RESULT BUFFER POINTER
      MOV  M,A         ;STORE RESULT IN BUFFER
      INX  H           ;BUMP RESULT POINTER
      SHLD TXRBUF     ;RESTORE RESULT POINTER
      MVI  A,01H     ;SET TXFLAG TO SHOW COMPLETION
      STA  TXFLAG     ;SET FLAG
TXI1:  POP  PSW        ;RESTORE PSW
      POP  H           ;RESTORE HL
      EI             ;ENABLE INTERRUPTS
      RET             ;DONE
TXI2:  LHLD  TXPNT     ;(*) GET DATA POINTER
      MOV  A,M         ;(*) GET DATA FROM BUFFER
      OUT  TXDATA     ;(*) OUTPUT TO 8273 VIA TXDACK
      INX  H           ;(*) BUMP DATA POINTER
      SHLD TXPNT     ;(*) RESTORE POINTER
      JMP  TXI1       ;(*) RETURN AFTER RESTORE

```

**Figure 22. Interrupt-Driven Result Handlers  
with Non-DMA Data Transfers**

```

;FUNCTION: POLOP
;INPUTS: NONE
;OUTPUTS: C=0 (NO STATUS), =1 (RX COMPLETION),
;        =2 (TX COMPLETION), =3 (BOTH)
;CALLS: TXI, RXI
;DESTROYS: B,C
;DESCRIPTION: POLOP IS CALLED TO POLL THE 8273 FOR
;DATA TRANSFERS AND COMPLETION RESULTS. THE
;ROUTINES TXI AND RXI ARE USED FOR THE ACTUAL
;TRANSFERS AND BUFFER WORK. POLOP RETURNS
;THE STATUS OF THEIR ACTION.
;
POLOP:  PUSH  PSW      ;SAVE PSW
        MVI  C,00H    ;CLEAR C
POLOP1: IN   STAT73   ;READ 8273 STATUS
        ANI  INT      ;ARE TXINT OR RXINT SET?
        JZ   EXIT     ;NO, EXIT
        IN   STAT73   ;READ 8273 STATUS
        ANI  RXINT    ;TEST RX INT
        JNZ  RXIC     ;YES, GO SERVICE RX
        CALL TXI      ;MUST BE TX, GO SERVICE IT
        LDA  TXFLAG   ;GET TX FLAG
        CPI  01H     ;WAS IT A COMPLETION? (01)
        JNZ  PEXIT    ;NO, SO JUST EXIT
        INR  C        ;YES, UPDATE C
        INK  C
        JMP  POLOP1   ;TRY AGAIN
;
        ;
RXIC:   CALL  RXI     ;GO SERVICE RX
        LDA  RXFLAG   ;GET RX FLAG
        CPI  01H     ;WAS IT A COMPLETION? (01)
        JNZ  PEXIT    ;NO, SO JUST EXIT
        INR  C        ;YES, UPDATE C
        JMP  POLOP1   ;TRY AGAIN
;
PEXIT:  POP  PSW      ;RESTORE PSW
        RET           ;RETURN WITH COMP. STATUS IN C

```

**Figure 23. Polling Result Handler**

## 8273 COMMAND DESCRIPTION

In this section, each command is discussed in detail. In order to shorten the notation, please refer to the command key in Table 1. The 8273 utilizes five different command types: Initialization/Configuration, Receive, Transmit, Reset, and Modem Control.

### Initialization/Configuration Commands

The Initialization/Configuration commands manipulate registers internal to the 8273 that define the various operating modes. These commands either set or reset specified bits in the registers depending on the type of command. One parameter is required. Set commands perform a logical OR operation of the parameter (mask) and the internal register. This mask contains 1s where register bits are to be set. A 0 in the mask causes no change in the corresponding register bit. Reset commands perform a logical AND operation of the parameter (mask) and the internal register, i.e., the mask is 0 to reset a register bit and a 1 to cause no change. Before presenting the commands, the register bit definitions are discussed.

**TABLE 1. COMMAND SUMMARY KEY**

$B_0, B_1$  — LSB AND MSB OF RECEIVE BUFFER LENGTH  
 $R_0, R_1$  — LSB AND MSB OF RECEIVED FRAME LENGTH  
 $L_0, L_1$  — LSB AND MSB OF TRANSMIT FRAME LENGTH  
 $A_1, A_2$  — MATCH ADDRESSES FOR SELECTIVE RECEIVE  
 RIC — RECEIVER INTERRUPT RESULT CODE  
 TIC — TRANSMITTER INTERRUPT RESULT CODE  
 A — ADDRESS FIELD OF RECEIVED FRAME  
 C — CONTROL FIELD OF RECEIVED FRAME



## Operating Mode Register (Figure 24)

D<sub>7</sub>-D<sub>6</sub>: *Not Used* — These bits must not be manipulated by any command; i.e., D<sub>7</sub>-D<sub>6</sub> must be 0 for the Set command and 1 for the Reset command.

D<sub>5</sub>: *HDLC Abort* — When this bit is set, the 8273 will interrupt when 7 1s (HDLC Abort) are received by an active receiver. When reset, an SDLC Abort (8 1s) will cause an interrupt.

D<sub>4</sub>: *EOP Interrupt* — Reception of an EOP character (0 followed by 7 1s) will cause the 8273 to interrupt the CPU when this bit is set. Loop controller stations use this mode as a signal that a polling frame has completed the loop. No EOP interrupt is generated when this bit is reset.

D<sub>3</sub>: *Early Tx Interrupt* — This bit specifies when the transmitter should generate an end of frame interrupt. If this bit is set, an interrupt is generated when the last data character has been passed to the 8273. If the user software issues another transmit command within two byte times, the final flag interrupt does not occur and the new frame is transmitted with only one flag of separation. If this restriction is not met, more than one flag will separate the frames and a frame complete interrupt is generated after the closing flag. If the bit is reset, only the frame complete interrupt occurs. This bit, when set, allows a single flag to separate consecutive frames.

D<sub>2</sub>: *Buffered Address and Control* — When set, the address and control fields of received frames are buffered in the 8273 and passed to the CPU as results after a received frame interrupt (they are not transferred to memory with the information field). On transmit, the A and C fields are passed to the 8273 as parameters. This mode simplifies buffer management. When this bit is reset, the A and C fields are passed to and from memory as the first two data transfers.

D<sub>1</sub>: *Preframe Sync* — When set, the 8273 prefaces each transmitted frame with two characters before the opening flag. These two characters provide 16 transitions to allow synchronization of the opposing receiver. To guarantee 16 transitions, the two characters are 55H-55H for non-NRZI mode (see Serial I/O Register description) or 00H-00H for NRZI mode. When reset, no preframe characters are transmitted.

D<sub>0</sub>: *Flag Stream* — When set, the transmitter will start sending flag characters as soon as it is idle; i.e., immediately if idle when the command is issued or after a transmission if the transmitter is active when this bit is set. When reset, the transmitter starts sending idle characters on the next character boundary if idle already, or at the end of a transmission if active.

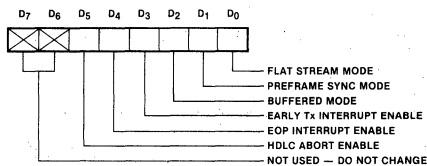


Figure 24. Operating Mode Register

## Serial I/O Mode Register (Figure 25)

D<sub>7</sub>-D<sub>3</sub>: *Not Used* — These bits must be 0 for the Set command and 1 for the Reset command.

D<sub>2</sub>: *Data Loopback* — When set, transmitted data (Tx<sub>D</sub>) is internally routed to the receive data circuitry. When reset, Tx<sub>D</sub> and Rx<sub>D</sub> are independent.

D<sub>1</sub>: *Clock Loopback* — When set, Tx<sub>C</sub> is internally routed to Rx<sub>C</sub>. When reset, the clocks are independent.

D<sub>0</sub>: *NRZI (Non-Return to Zero Inverted)* — When set, the 8273 assumes the received data is NRZI encoded, and NRZI encodes the transmitted data. When reset, the received and transmitted data are treated as a normal positive logic bit stream.

## Data Transfer Mode Register (Figure 26)

D<sub>7</sub>-D<sub>1</sub>: *Not Used* — These bits must be 0 for the Set command and 1 for the Reset command.

D<sub>0</sub>: *Interrupt Data Transfer* — When set, the 8273 will interrupt the CPU when data transfers are required (the corresponding IRA Status register bit will be 0 to signify a data transfer interrupt rather than a Result phase interrupt). When reset, 8273 data transfers are performed through DMA requests on the DRQ pins without interrupting the CPU.

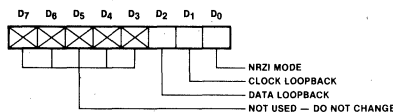


Figure 25. Serial I/O Mode Register

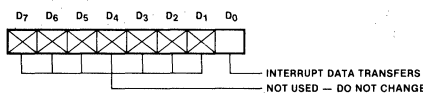


Figure 26. Data Transfer Mode Register

# APPLICATIONS

## One Bit Delay Register (Figure 27)

D<sub>7</sub>: **One Bit Delay** — When set, the 8273 retransmits the received data stream one bit delayed. This mode is entered and exited at a received character boundary. When reset, the transmitted and received data are independent. This mode is utilized for loop operation and is discussed in a later section.

D<sub>6</sub>-D<sub>0</sub>: **Not Used** — These bits must be 0 for the Set command and 1 for the Reset command.

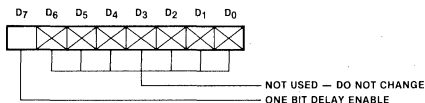


Figure 27. One Bit Delay Mode Register

Figure 28 shows the Set and Reset commands associated with the above registers. The mask which sets or resets the desired bits is treated as a single parameter. These commands do not interrupt nor provide results during the Result phase. After reset, the 8273 defaults to all of these bits reset.

REGISTER	COMMAND	HEX CODE	PARAMETER
ONE BIT DELAY MODE	SET	A4	SET MASK
	RESET	64	RESET MASK
DATA TRANSFER MODE	SET	97	SET MASK
	RESET	57	RESET MASK
OPERATING MODE	SET	91	SET MASK
	RESET	51	RESET MASK
SERIAL I/O MODE	SET	A0	SET MASK
	RESET	60	RESET MASK

Figure 28. Initialization/Configuration Command Summary

## Receive Commands

The 8273 supports three receive commands plus a receiver disable function.

### General Receive

When commanded to General Receive, the 8273 passes all frames either to memory (DMA mode) or to the CPU (non-DMA mode) regardless of the contents of the frame's address field. This command is used for primary and loop controller stations. Two parameters are required: B<sub>0</sub> and B<sub>1</sub>. These parameters are the LSB and MSB of the receiver buffer size. Giving the 8273 this extra information alleviates the CPU of the burden of checking for buffer overflow. The 8273 will interrupt the CPU if the received frame attempts to overflow the allotted buffer space.

### Selective Receive

In Selective Receive, two additional parameters besides B<sub>0</sub> and B<sub>1</sub> are required: A<sub>1</sub> and A<sub>2</sub>. These parameters are two address match bytes. When commanded to Selective Receive, the 8273 passes to memory or the CPU only those frames having an address field matching either A<sub>1</sub> or A<sub>2</sub>. This command is usually used for secondary stations with A<sub>1</sub> being the secondary address and A<sub>2</sub> is the "All Parties" address. If only one match byte is needed, A<sub>1</sub> and A<sub>2</sub> should be equal. As in General Receive, the 8273 counts the incoming data bytes and interrupts the CPU if B<sub>0</sub>, B<sub>1</sub> is exceeded.

### Selective Loop Receive

This command is very similar in operation to Selective Receive except that One Bit Delay mode must be set and that the loop is captured by placing transmitter in Flag Stream mode automatically after an EOP character is detected following a selectively received frame. The details of using the 8273 in loop configurations is discussed in a later section so please hold questions until then.

The handling of interrupt results is common among the three commands. When a frame is received without error, i.e., the FCS is correct and  $\overline{CD}$  (Carrier Detect) was active throughout the frame or no attempt was made to overflow the buffer; the 8273 interrupts the CPU following the closing flag to pass the completion results. These results, in order, are the receiver interrupt result code (RIC), and the byte length of the information field of the received frame (R<sub>0</sub>, R<sub>1</sub>). If Buffered mode is selected, the address and control fields are passed as two additional results. If Buffered mode is not selected, the address and control fields are passed as the first two data transfers and R<sub>0</sub>, R<sub>1</sub> reflect the information field length plus two.

### Receive Disable

The receiver may also be disabled using the Receive Disable command. This command terminates any receive operation immediately. No parameters are required and no results are returned.

The details for the Receive command are shown in Figure 29. The interrupt result code key is shown in Figure 30. Some explanation of these result codes is appropriate.

The interrupt result code is the first byte passed to the CPU in the R<sub>x</sub>I/R register during the Result phase. Bits D<sub>4</sub>-D<sub>0</sub> define the cause of the receiver interrupt. Since each result code has specific implications, they are discussed separately below.

COMMAND	HEX CODE	PARAMETERS	RESULTS* R <sub>x</sub> I/R
GENERAL RECEIVE	C0	B <sub>0</sub> , B <sub>1</sub>	RIC, R <sub>0</sub> , R <sub>1</sub> , A, C
SELECTIVE RECEIVE	C1	B <sub>0</sub> , B <sub>1</sub> , A <sub>1</sub> , A <sub>2</sub>	RIC, R <sub>0</sub> , R <sub>1</sub> , A, C
SELECTIVE LOOP RECEIVE	C2	B <sub>0</sub> , B <sub>1</sub> , A <sub>1</sub> , A <sub>2</sub>	RIC, R <sub>0</sub> , R <sub>1</sub> , A, C
DISABLE RECEIVER	C5	NONE	NONE

\*A AND C ARE PASSED AS RESULTS ONLY IN BUFFERED MODE.

Figure 29. Receiver Command Summary

# APPLICATIONS

RIC D7-D0	RECEIVER INTERRUPT RESULT CODE	Rx STATUS AFTER INT
* 00000	A <sub>1</sub> MATCH OR GENERAL RECEIVE	ACTIVE
* 00001	A <sub>2</sub> MATCH	ACTIVE
000 00011	CRC ERROR	ACTIVE
000 00100	ABORT DETECTED	ACTIVE
000 00101	IDLE DETECTED	DISABLED
000 00110	EOP DETECTED	DISABLED
000 00111	FRAME < 32 BITS	ACTIVE
000 01000	DMA OVERRUN	DISABLED
000 01001	MEMORY BUFFER OVERFLOW	DISABLED
000 01010	CARRIER DETECT FAILURE	DISABLED
000 01011	RECEIVER INTERRUPT OVERRUN	DISABLED

*D7-D5	PARTIAL BYTE RECEIVED
111	ALL 8 BITS OF LAST BYTE
000	D <sub>0</sub>
100	D <sub>1</sub> -D <sub>0</sub>
010	D <sub>2</sub> -D <sub>0</sub>
110	D <sub>3</sub> -D <sub>0</sub>
001	D <sub>4</sub> -D <sub>0</sub>
101	D <sub>5</sub> -D <sub>0</sub>
011	D <sub>6</sub> -D <sub>0</sub>

Figure 30. Receiver Interrupt Result Codes (RIC)

The first two result codes result from the error-free reception of a frame. If the frame is received correctly after a General Receive command, the first result is returned. If either Selective Receive command was used (normal or loop), a match with A<sub>1</sub> generates the first result code and a match with A<sub>2</sub> generates the second. In either case, the receiver remains active after the interrupt; however, the internal buffer size counters are not reset. That is, if the receive command indicated 100 bytes were allocated to the receive buffer (B<sub>0</sub>, B<sub>1</sub>) and an 80-byte frame was received correctly, the maximum next frame size that could be received without recommanding the receiver (resetting B<sub>0</sub> and B<sub>1</sub>) is 20 bytes. Thus, it is common practice to recommand the receiver after each frame reception. DMA and/or memory pointers are usually updated at this time. (Note that users who do not wish to take advantage of the 8273's buffer management features may simply use B<sub>0</sub>, B<sub>1</sub> = 0FFH for each receive command. Then frames of 65K bytes may be received without buffer overflow errors.)

The third result code is a CRC error. This indicates that a frame was received in the correct format (flags, etc.); however, the received FCS did not check with the internally generated FCS. The frame should be discarded. The receiver remains active. (Do not forget that even though an error condition has been detected, all frame information up until that error has either been transferred to memory or passed to the CPU. This information should be invalidated. This applies to all receiver error conditions.) Note that the FCS, either transmitted or received, is never available to the CPU.

The Abort Detect result occurs whenever the receiver sees either an SDLC (8 1s) or an HDLC (7 1s), depending on the Operating Mode register. However, the intervening Abort character between a closing flag and an Idle does not generate an interrupt. If an Abort character (seen by an active receiver within a frame) is not preceded by a flag and is followed by an Idle, an interrupt will be generated for the Abort, followed by an Idle inter-

rupt one character time later. The Idle Detect result occurs whenever 15 consecutive 1s are received. After the Abort Detect interrupt, the receiver remains active. After the Idle Detect interrupt, the receiver is disabled and must be recommanded before further frames may be received.

If the EOP Interrupt bit is set in the Operating Mode register, the EOP Detect result is returned whenever an EOP character is received. The receiver is disabled, so the Idle following the EOP does not generate an Idle Detect interrupt.

The minimum number of bits in a valid frame between the flags is 32. Fewer than 32 bits indicates an error. If Buffered mode is selected, such frames are ignored, i.e., no data transfers or interrupts are generated. In non-Buffered mode, a < 32-bit frame generates an interrupt with the < 32-bit Frame result since data transfers may already have disturbed the 8257 or interrupt handler. The receiver remains active.

The DMA Overrun result results from the DMA controller being too slow in extracting data from the 8273, i.e., the RxDACK signal is not returned before the next received byte is ready for transfer. The receiver is disabled if this error condition occurs.

The Memory Buffer Overflow result occurs when the number of received bytes exceeds the receiver buffer length supplied by the B<sub>0</sub> and B<sub>1</sub> parameters in the receive command. The receiver is disabled.

The Carrier Detect Failure result occurs when the CD pin goes high (inactive) during reception of a frame. The CD pin is used to qualify reception and must be active by the time the address field starts to be received. If CD is lost during the frame, a CD Failure interrupt is generated and the receiver is disabled. No interrupt is generated if CD goes inactive between frames.

If a condition occurs requiring an interrupt to be generated before the CPU has finished reading the previous interrupt results, the second interrupt is generated after the current Result phase is complete (the RxINT pin and status bit go low then high). However, the interrupt result for this second interrupt will be a Receive Interrupt Overrun. The actual cause of the second interrupt is lost. One case where this may occur is at the end of a received frame where the line goes idle. The 8273 generates a received frame interrupt after the closing flag and then 15-bit times later, generates an Idle Detect interrupt. If the interrupt service routine is slow in reading the first interrupt's results, the internal RxI/R register still contains result information when the Idle Detect interrupt occurs. Rather than wiping out the previous results, the 8273 adds a Receive Interrupt Overrun result as an extra result. If the system's interrupt structure is such that the second interrupt is not acknowledged (interrupts are still disabled from the first interrupt), the Receive Interrupt Overrun result is read as an extra result, after those from the first interrupt. If the second interrupt is serviced, the Receive Interrupt Overrun is returned as a single result. (Note that the INT pins supply the necessary transitions to support a Program-

# APPLICATIONS

mable Interrupt Controller such as the Intel 8259. Each interrupt generates a positive-going edge on the appropriate INT pin and the high level is held until the interrupt is completely serviced.) In general, it is possible to have interrupts occurring at one character time intervals. Thus the interrupt handling software must have at least that much response and service time.

The occurrence of Receive Interrupt Overruns is an indication of marginal software design; the system's interrupt response and servicing time is not sufficient for the data rates being attempted. It is advisable to configure the interrupt handling software to simply read the interrupt results, place them into a buffer, and clear the interrupt as quickly as possible. The software can then examine the buffer for new results at its leisure, and take appropriate action. This can easily be accomplished by using a result buffer flag that indicates when new results are available. The interrupt handler sets the flag and the main program resets it once the results are retrieved.

Both SDLC and HDLC allow frames which are of arbitrary length (>32 bits). The 8273 handles this N-bit reception through the high order bits ( $D_7$ - $D_5$ ) of the result code. These bits code the number of valid received bits in the last received information field byte. This coding is shown in Figure 30. The high order bits of the received partial byte are indeterminate. [The address, control, and information fields are transmitted least significant bit ( $A_0$ ) first. The FCS is complemented and transmitted most significant bit first.]

## Transmit Commands

The 8273 transmitter is supported by three Transmit commands and three corresponding Abort commands.

### Transmit Frame

The Transmit Frame command simply transmits a frame. Four parameters are required when Buffered mode is selected and two when it is not. In either case, the first two parameters are the least and the most significant bytes of the desired frame length ( $L_0$ ,  $L_1$ ). In Buffered mode,  $L_0$  and  $L_1$  equal the length in bytes of the desired information field, while in the non-Buffered mode,  $L_0$  and  $L_1$  must be specified as the information field length plus two. ( $L_0$  and  $L_1$  specify the number of data transfers to be performed.) In Buffered mode, the address and control fields are presented to the transmitter as the third and fourth parameters respectively. In non-Buffered mode, the A and C fields must be passed as the first two data transfers.

When the Transmit Frame command is issued, the 8273 makes  $\overline{RTS}$  (Request-to-Send) active (pin low) if it was not already. It then waits until  $\overline{CTS}$  (Clear-to-Send) goes active (pin low) before starting the frame. If the Preframe Sync bit in the Operating Mode register is set, the transmitter prefaces two characters (16 transitions) before the opening flag. If the Flag Stream bit is set in the Operating Mode register, the frame (including Preframe Sync if selected) is started on a flag boundary. Otherwise the frame starts on a character boundary.

At the end of the frame, the transmitter interrupts the CPU (the interrupt results are discussed shortly) and returns to either Idle or Flag Stream, depending on the Flag Stream bit of the Operating Mode register. If  $\overline{RTS}$  was active before the transmit command, the 8273 does not change it. If it was inactive, the 8273 will deactivate it within one character time.

### Loop Transmit

Loop Transmit is similar to Frame Transmit (the parameter definition is the same). But since it deals with loop configurations, One Bit Delay mode must be selected.

If the transmitter is not in Flag Stream mode when this command is issued, the transmitter waits until after a received EOP character has been converted to a flag (this is done automatically) before transmitting. (The one bit delay is, of course, suspended during transmit.) If the transmitter is already in Flag Stream mode as a result of a selectively received frame during a Selective Loop Receive command, transmission will begin at the next flag boundary for Buffered mode or at the third flag boundary for non-Buffered mode. This discrepancy is to allow time for enough data transfers to occur to fill up the internal transmit buffer. At the end of a Loop Transmit, the One Bit Delay mode is re-entered and the flag stream mode is reset. More detailed loop operation is covered later.

### Transmit Transparent

The Transmit Transparent command enables the 8273 to transmit a block of raw data. This data is without SDLC protocol, i.e., no zero bit insertion, flags, or FCS. Thus it is possible to construct and transmit a Bi-Sync message for front-end processor switching or to construct and transmit an SDLC message with incorrect FCS for diagnostic purposes. Only the  $L_0$  and  $L_1$  parameters are used since there are not fields in this mode. (the 8273 does not support a Receive Transparent command.)

### Abort Commands

Each of the above transmit commands has an associated Abort command. The Abort Frame Transmit command causes the transmitter to send eight contiguous ones (no zero bit insertion) immediately and then revert to either idle or flag streaming based on the Flag Stream bit. (The 8 1s as an Abort character is compatible with both SDLC and HDLC.)

For Loop Transmit, the Abort Loop Transmit command causes the transmitter to send one flag and then revert to one bit delay. Loop protocol depends upon FCS errors to detect aborted frames.

The Abort Transmit Transparent simply causes the transmitter to revert to either idles or flags as a function of the Flag Stream mode specified.

The Abort commands require no parameters, however, they do generate an interrupt and return a result when complete.

A summary of the Transmit commands is shown in Figure 31. Figure 32 shows the various transmit interrupt result codes. As in the receiver operation, the transmitter generates interrupts based on either good

# APPLICATIONS

completion of an operation or an error condition to start the Result phase.

The Early Transmit Interrupt result occurs after the last data transfer to the 8273 if the Early Transmit Interrupt bit is set in the Operating Mode register. If the 8273 is commanded to transmit again within two character times, a single flag will separate the frames. (Buffered mode must be used for a single flag to separate the frames. If non-Buffered mode is selected, three flags will separate the frames.) If this time constraint is not met, another interrupt is generated and multiple flags or idles will separate the frames. The second interrupt is the normal Frame Transmit Complete interrupt. The Frame Transmit Complete result occurs at the closing flag to signify a good completion.

The DMA Underrun result is analogous to the DMA Overrun result in the receiver. Since SDLC does not support intraframe time fill, if the DMA controller or CPU does not supply the data in time, the frame must be aborted. The action taken by the transmitter on this error is automatic. It aborts the frame just as if an Abort command had been issued.

Clear-to-Send Error result is generated if  $\overline{\text{CTS}}$  goes inactive during a frame transmission. The frame is aborted as above.

The Abort Complete result is self-explanatory. Please note however that no Abort Complete interrupt is generated when an automatic abort occurs. The next command type consists of only one command.

COMMAND	HEX CODE	PARAMETERS*	RESULTS Tx/I/R
TRANSMIT FRAME ABORT	C8 CC	L <sub>0</sub> , L <sub>1</sub> , A, C NONE	TIC TIC
LOOP TRANSMIT ABORT	CA CE	L <sub>0</sub> , L <sub>1</sub> , A, C NONE	TIC TIC
TRANSMIT TRANSPARENT ABORT	C0 CD	L <sub>0</sub> , L <sub>1</sub> NONE	TIC TIC

\*A AND C ARE PASSED AS PARAMETERS IN BUFFERED MODE ONLY.

Figure 31. Transmitter Command Summary

TIC D <sub>7</sub> -D <sub>0</sub>	TRANSMITTER INTERRUPT RESULT CODE	Tx STATUS AFTER INT
000 01100	EARLY Tx INTERRUPT	ACTIVE
000 01101	FRAME Tx COMPLETE	IDLE OR FLAGS
000 01110	DMA UNDERRUN	ABORT
000 01111	CLEAR TO SEND ERROR	ABORT
000 10000	ABORT COMPLETE	IDLE OR FLAGS

Figure 32. Transmitter Interrupt Result Codes

## Reset Command

The Reset command provides a software reset function for the 8273. It is a special case and does not utilize the normal command interface. The reset facility is provided in the Test Mode register. The 8273 is reset by simply outputting a 01H followed by a 00H to the Test Mode register. Writing the 01 followed by the 00 mimicks the action required by the hardware reset. Since the 8273 requires time to process the reset internally, at least 10 cycles of the  $\phi$ CLK clock must occur between the

writing of the 01 and the 00. The action taken is the same as if a hardware reset is performed, namely:

1. The modem control outputs are forced high inactive).
2. The 8273 Status register is cleared.
3. Any commands in progress cease.
4. The 8273 enters an idle state until the next command is issued.

## Modem Control Commands

The modem control ports were discussed earlier in the Hardware section. The commands used to manipulate these ports are shown in Figure 33. The Read Port A and Read Port B commands are immediate. The bit definition for the returned byte is shown in Figures 13 and 14. Do not forget that the returned value represents the logical condition of the pin, i.e., pin active (low) = bit set.

PORT	COMMAND	HEX CODE	PARAMETER	REG RESULT
A INPUT	READ	22	NONE	PORT VALUE
	READ	23	NONE	PORT VALUE
B OUTPUT	SET	A3	SET MASK	NONE
	RESET	63	RESET MASK	NONE

Figure 33. Modem Control Command Summary

The Set and Reset Port B commands are similar to the Initialization commands in that they use a mask parameter which defines the bits to be changed. Set Port B utilizes a logical OR mask and Reset Port B uses a logical AND mask. Setting a bit makes the pin active (low). Resetting the bit deactivates the pin (high).

To help clarify the numerous timing relationships that occur and their consequences, Figures 34 and 35 are provided as an illustration of several typical sequences. It is suggested that the reader go over these diagrams and re-read the appropriate part of the previous sections if necessary.

## HDLC CONSIDERATIONS

The 8273 supports HDLC as well as SDLC. Let's discuss how the 8273 handles the three basic HDLC/SDLC differences: extended addressing, extended control, and the 7 is Abort character.

Recalling Figure 4A, HDLC supports an address field of indefinite length. The actual amount of extension used is determined by the least significant bit of the characters immediately following the opening flag. If the LSB is 0, more address field bytes follow. If the LSB is 1, this byte is the final address field byte. Software must be used to determine this extension.

If non-Buffered mode is used, the A, C, and I fields are in memory. The software must examine the initial characters to find the extent of the address field. If Buffered mode is used, the characters corresponding to the SDLC A and C fields are transferred to the CPU as interrupt results. Buffered mode assumes the two characters following the opening flag are to be transferred as interrupt results regardless of content or meaning. (The 8273

# APPLICATIONS

does not know whether it is being used in an SDLC or an HDLC environment.) In SDLC, these characters are necessarily the A and C field bytes, however in HDLC, their meaning may change depending on the amount of extension used. The software must recognize this and examine the transferred results as possible address field extensions.

Frames may still be selectively received as is needed for secondary stations. The Selective Receive command is still used. This command qualifies a frame reception on the first byte following the opening flag matching either of the  $A_1$  or  $A_2$  match byte parameters. While this does not allow qualification over the complete range of HDLC addresses, it does perform a qualification on the first address byte. The remaining address field bytes, if any, are then examined via software to completely qualify the frame.

Once the extent of the address field is found, the following bytes form the control field. The same LSB test used for the address field is applied to these bytes to determine the control field extension, up to two bytes maximum. The remaining frame bytes in memory represent the information field.

The Abort character difference is handled in the Operating Mode register. If the HDLC Abort Enable bit is set, the reception of seven contiguous ones by an active receiver will generate an Abort Detect interrupt rather than eight ones. (Note that both the HDLC Abort Enable bit and the EOP Interrupt bit must not be set simultaneously.)

Now let's move on to the SDLC loop configuration discussion.

## LOOP CONFIGURATION

Aside from use in the normal data link applications, the 8273 is extremely attractive in loop configuration due to the special frame-level loop commands and the Digital Phase Locked Loop. Toward this end, this section details the hardware and software considerations when using the 8273 in a loop application.

The loop configuration offers a simple, low-cost solution for systems with multiple stations within a small physical location, i.e., retail stores and banks. There are two primary reasons to consider a loop configuration. The interconnect cost is lower for a loop over a multi-point configuration since only one twisted pair or fiber optic cable is used. (The loop configuration does not support the passing of distinct clock signals from station to station.) In addition, loop stations do not need the intelligence of a multi-point station since the loop protocol is simpler. The most difficult aspects of loop station design are clock recovery and implementation of one bit delay (both are handled neatly by the 8273).

Figure 36 illustrates a typical loop configuration with one controller and two down-loop secondaries. Each station must derive its own data timing from the received data stream. Recalling our earlier discussion of the DPLL, notice that  $TxC$  and  $RxC$  clocks are provided by the DPLL output. The only clock required in the secondaries is a simple, non-synchronized clock at 32 times the desired baud rate. The controller requires both  $32\times$  and  $1\times$  clocks. (The  $1\times$  is usually implemented by dividing the  $32\times$  clock with a 5-bit divider. However, there is no synchronism requirement between these clocks so any convenient implementation may be used.)

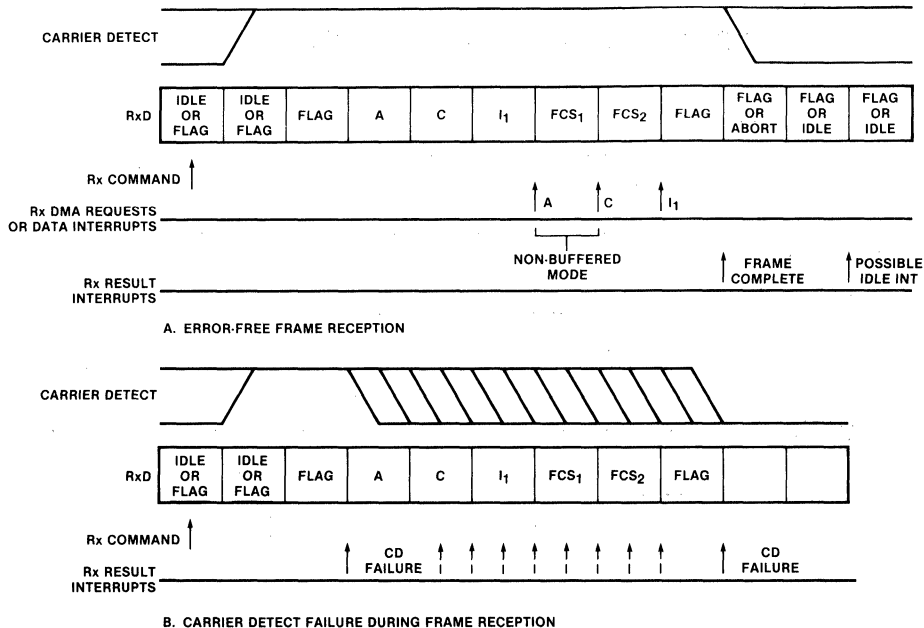


Figure 34. Sample Receiver Timing Diagrams

# APPLICATIONS

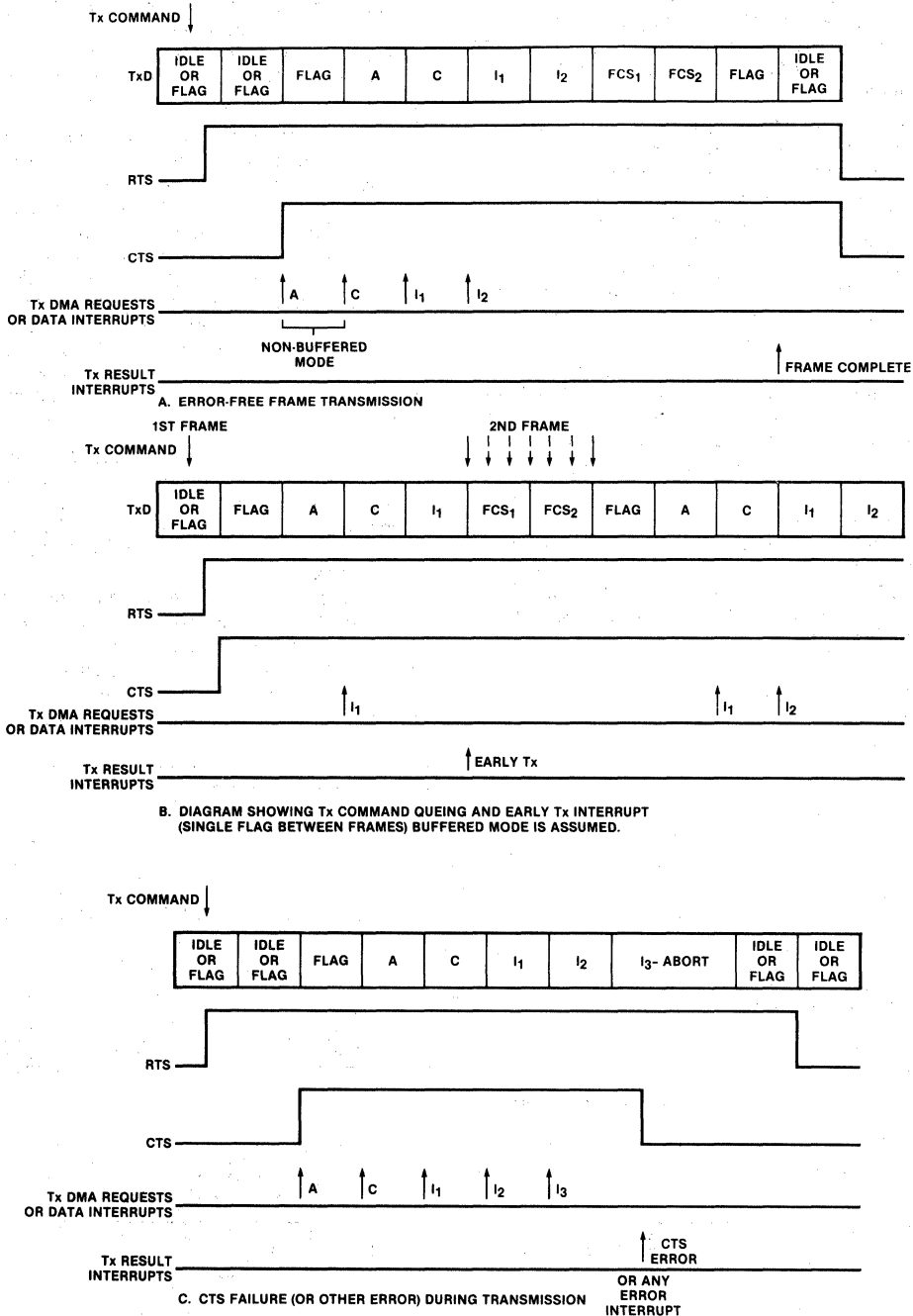


Figure 35. Sample Transmitter Timing Diagrams

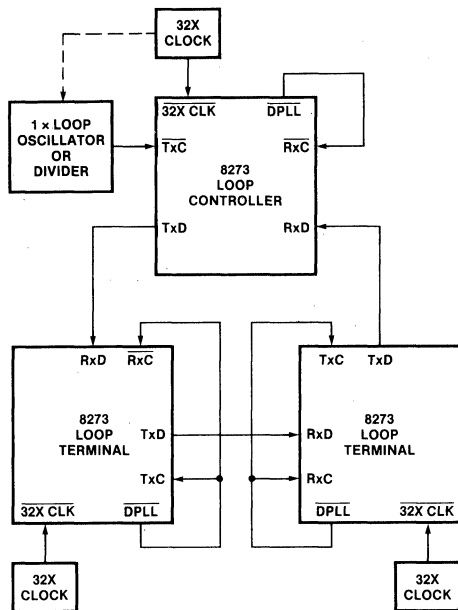


Figure 36. SDLC Loop Application

A quick review of loop protocol is appropriate. All communication on the loop is controlled by the loop controller. When the controller wishes to allow the secondaries to transmit, it sends a polling frame (the control field contains a poll code) followed by an EOP (End-of-Poll) character. The secondaries use the EOP character to capture the loop and insert a response frame as will be discussed shortly.

The secondaries normally operate in the repeater mode, retransmitting received data with one bit time of delay. All received frames are repeated. The secondary uses the one bit time of delay to capture the loop.

When the loop is idle (no frames), the controller transmits continuous flag characters. This keeps transitions on the loop for the sake of down-loop phase locked loops. When the controller has a non-polling frame to transmit, it simply transmits the frame and continues to send flags. The non-polling frame is then repeated around the loop and the controller receives it to signify a complete traversal of the loop. At the particular secondary addressed by the frame, the data is transferred to memory while being repeated. Other secondaries simply repeat it.

If the controller wants to poll the secondaries, it transmits a polling frame followed by all 1s (no zero bit insertion). The final zero of the closing frame plus the first seven 1s form an EOP. While repeating, the secondaries monitor their incoming line for an EOP. When an EOP is received, the secondary checks if it has any response for the controller. If not, it simply continues repeating. If the secondary has a response, it changes the seventh EOP one into a zero (the one bit time of delay allows time for this) and repeats it, forming a flag for the down-loop stations. After this flag is transmitted,

the secondary terminates its repeater function and inserts its response frame (with multiple preceding flags if necessary). After the closing flag of the response, the secondary re-enters its repeater function, repeating the up-loop controller 1s. Notice that the final zero of the response's closing flag plus the repeated 1s from the controller form a new EOP for the next down-loop secondary. This new EOP allows the next secondary to insert a response if it desires. This gives each secondary a chance to respond.

Back at the controller, after the polling frame has been transmitted and the continuous 1s started, the controller waits until it receives an EOP. Receiving an EOP signifies to the controller that the original frame has propagated around the loop followed by any responses inserted by the secondaries. At this point, the controller may either send flags to idle the loop or transmit the next frame. Let's assume that the loop is implemented completely with the 8273s and describe the command flows for a typical controller and secondary.

The loop controller is initialized with commands which specify that the NRZI, Preframe Sync, Flag Stream, and EOP Interrupt modes are set. Thus, the controller encodes and decodes all data using NRZI format. Preframe Sync mode specifies that all transmitted frames be prefaced with 16 line transitions. This ensures that the minimum of 12 transitions needed by the DPLLs to lock after an all 1s line have occurred by the time the secondary sees a frame's opening flag. Setting the Flag Stream mode starts the transmitter sending flags which idles the loop. And the EOP Interrupt mode specifies that the controller processor will be interrupted whenever the active receiver sees an EOP, indicating the completion of a poll cycle.

When the controller wishes to transmit a non-polling frame, it simply executes a Frame Transmit command. Since the Flag Stream mode is set, no EOP is formed after the closing flag. When a polling frame is to be transmitted, a General Receive command is executed first. This enables the receiver and allows reception of all incoming frames; namely, the original polling frame plus any response frames inserted by the secondaries. After the General Receive command, the frame is transmitted with a Frame Transmit command. When the frame is complete, a transmitter interrupt is generated. The loop controller processor uses this interrupt to reset Flag Stream mode. This causes the transmitter to start sending all 1s. An EOP is formed by the last flag and the first 7 1s. This completes the loop controller transmit sequence.

At any time following the start of the polling frame transmission the loop controller receiver will start receiving frames. (The exact time difference depends, of course, on the number of down-loop secondaries due to each inserting one bit time of delay.) The first received frame is simply the original polling frame. However, any additional frames are those inserted by the secondaries. The loop controller processor knows all frames have been received when it sees an EOP Interrupt. This interrupt is generated by the 8273 since the EOP Interrupt mode was set during initialization. At this point, the transmitter may be commanded either to enter Flag



# APPLICATIONS

Stream mode, idling the loop, or to transmit the next frame. A flowchart of the above sequence is shown in Figure 37.

The secondaries are initialized with the NRZI and One Bit Delay modes set. This puts the 8273 into the repeater mode with the transmitter repeating the received data with one bit time of delay. Since a loop station cannot transmit until it sees and EOP character, any transmit command is queued until an EOP is received. Thus whenever the secondary wishes to transmit a response, a Loop Transmit command is issued. The 8273 then waits until it receives an EOP. At this point, the receiver changes the EOP into a flag, repeats it, resets One Bit Delay mode stopping the repeater function, and sets the transmitter into Flag Stream mode. This captures the loop. The transmitter now inserts its message. At the closing flag, Flag Stream mode is reset, and One Bit Delay mode is set, returning the 8273 to repeater function and forming an EOP for the next down-loop station. These actions happen automatically after a Loop Transmit command is issued.

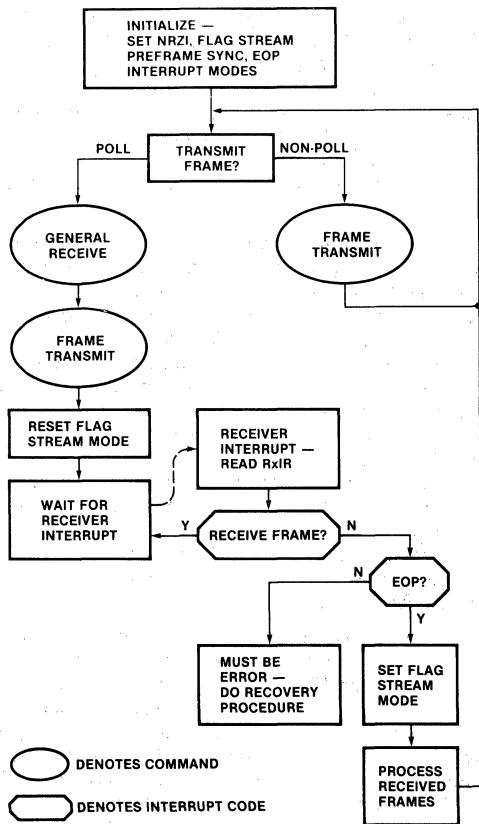


Figure 37. Loop Controller Flowchart

When the secondary wants its receiver enabled, a Selective Loop Receive command is issued. The receiver then looks for a frame having a match in the Address field. Once such a frame is received, repeated, and transferred to memory, the secondary's processor is interrupted with the appropriate Match interrupt result and the 8273 continues with the repeater function until an EOP is received, at which point the loop is captured as above. The processor should use the interrupt to determine if it has a message for the controller. If it does, it simply issues a Loop Transmit command and things progress as above. If the processor has no message, the software must reset the Flag Stream mode bit in the Operating Mode register. This will inhibit the 8273 from capturing the loop at the EOP. (The match frame and the EOP may be separated in time by several frames depending on how many up-loop stations inserted messages of their own.) If the timing is such that the receiver has already captured the loop when the Flag Stream mode bit is reset, the mode is exited on a flag boundary and the frame just appears to have extra closing flags before the EOP. Notice that the 8273 handles the queuing of the transmit commands and the setting and resetting of the mode bits automatically. Figure 38 illustrates the major points of the secondary command sequence.

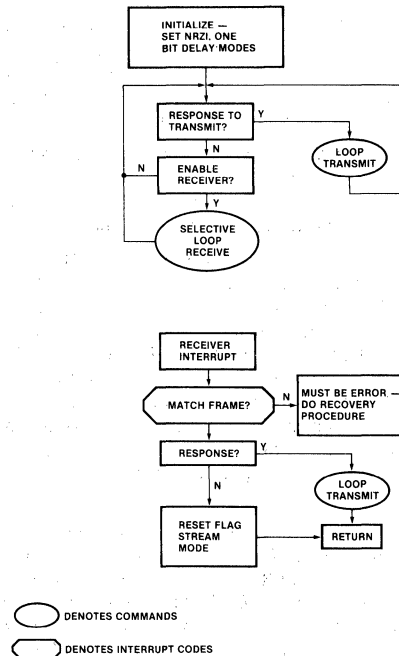


Figure 38. Loop Secondary Flowchart

# APPLICATIONS

When an off-line secondary wishes to come on-line, it must do so in a manner which does not disturb data on the loop. Figure 39 shows a typical hardware interface. The line labeled Port could be one of the 8273 Port B outputs and is assumed to be high (1) initially. Thus up-loop data is simply passed down-loop with no delay; however, the receiver may still monitor data on the loop. To come on-line, the secondary is initialized with only the EOP Interrupt mode set. The up-loop data is then monitored until an EOP occurs. At this point, the secondary's CPU is interrupted with an EOP interrupt. This signals the CPU to set One Bit Delay mode in the 8273 and then to set Port low (active). These actions switch the secondary's one bit delay into the loop. Since after the EOP only 1s are traversing the loop, no loop disturbance occurs. The secondary now waits for the next EOP, captures the loop, and inserts a "new on-line" message. This signals the controller that a new secondary exists and must be acknowledged. After the secondary receives its acknowledgement, the normal command flow is used.

It is hopefully evident from the above discussion that the 8273 offers a very simple and easy to implement solution for designing loop stations whether they are controllers or down-loop secondaries.

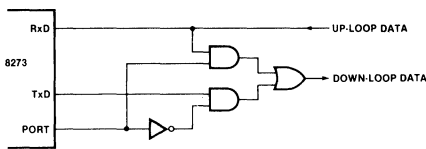


Figure 39. Loop Interface

## APPLICATION EXAMPLE

This section describes the hardware and software of the 8273/8085 system used to verify the 8273 implementation of SDLC on an actual IBM SDLC Link. This IBM link was gratefully volunteered by Raytheon Data Systems in Norwood, Mass. and I wish to thank them for their generous cooperation. The IBM system consisted of a 370 Mainframe, a 3705 Communications Processor, and a 3271 Terminal Controller. A Comlink II Modem supplied the modem interface and all communications took place at 4800 baud. In addition to observing correct responses, a Spectron D601B Datascope was used to verify the data exchanges. A block diagram of the system is shown in Figure 40. The actual verification was accomplished by the 8273 system receiving and responding to polls from the 3705. This method was used on both point-to-point and multi-point configurations. No attempt was made to implement any higher protocol software over that of the poll and poll responses since such software would not affect the verification of the 8273 implementation. As testimony to the ease of use of the 8273, the system worked on the first try.

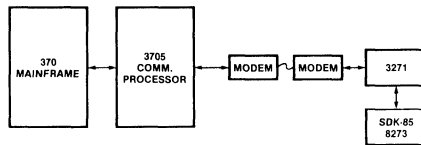


Figure 40. Raytheon Block Diagram

An SDK-85 (System Design Kit) was used as the core 8085 system. This system provides up to 4K bytes of ROM/EPROM, 512 bytes of RAM, 76 I/O pins, plus two timers as provided in two 8755 Combination EPROM/I/O devices and two 8155 Combination RAM/I/O/Timer devices. In addition, 5 interrupt inputs are supplied on the 8085. The address, data, and control buses are buffered by the 8212 and 8216 latches and bidirectional bus drivers. Although it was not used in this application, an 8279 Display Driver/Keyboard Encoder is included to interface the on-board display and keyboard. A block diagram of the SDK-85 is shown in Figure 41. The 8273 and associated circuitry was constructed on the ample wire-wrap area provided for the user.

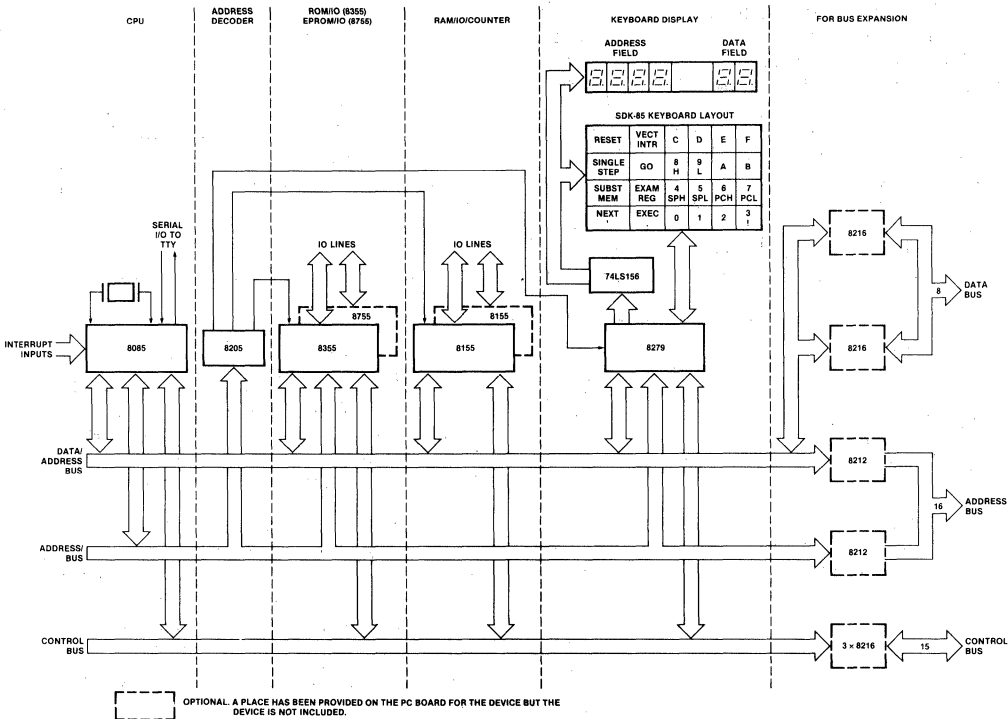
The example 8273/8085 system is interrupt-driven and uses DMA for all data transfers supervised by an 8257 DMA Controller. A 2400 baud asynchronous line, implemented with an 8251A USART, provides communication between the software and the user. 8253 Programmable Interval Timer is used to supply the baud rate clocks for the 8251A and 8273. (The 8273 baud rate clocks were used only during initial system debug. In actual operation, the modem supplied these clocks via the RS-232 interface.) Two 2142 1K x 4 RAMs provided 512 bytes of transmitter and 512 bytes of receiver buffer memory. (Command and result buffers, plus miscellaneous variables are stored in the 8155s.) The RS-232 interface utilized MC1488 and MC1489 RS-232 drivers and receivers. The schematic of the system is shown in Figure 42.

One detail to note is the DMA and interrupt structure of the transmit and receive channels. In both cases, the receiver is always given the higher priority (8257 DMA channel 0 has priority over the remaining channels and the 8085 RST 7.5 interrupt input has priority over the RST 6.5 input.) Although the choice is arbitrary, this technique minimizes the chance that received data could be lost due to other processor or DMA commitments.

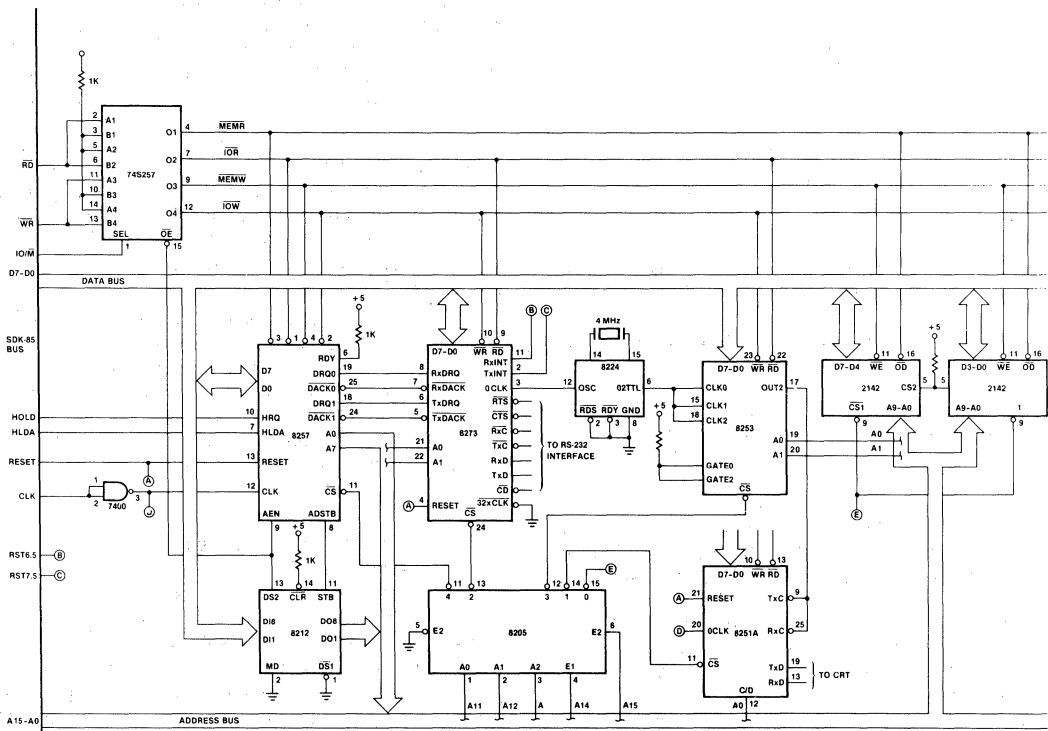
Also note that only one 8205 Decoder is used for both the peripherals' and the memorys' Chip Selects. This was done to eliminate separate memory and I/O decoders since it was known beforehand that neither address space would be completely filled.

The 4 MHz crystal and 8224 Clock Generator were used only to verify that the 8273 operates correctly at that maximum spec speed. In a normal system, the 3.072 MHz clock from the 8085 would be sufficient. (This fact was verified during initial checkout.)

# APPLICATIONS



**Figure 41. SDK-85 Functional Block Diagram**



**Figure 42. 8273/SDK-85 System**

# APPLICATIONS

The software consists of the normal monitor program supplied with the SDK-85 and a program to input commands to the 8273 and to display results. The SDK-85 monitor allows the user to read and write on-board RAM, start execution at any memory location, to single-step through a program, and to examine any of the 8085's internal registers. The monitor drives either the on-board keyboard/LED display or a serial TTY interface. This monitor was modified slightly in order to use the 8251A with a 2400 baud CRT as opposed to the 110 baud normally used. The 8273 program implements monitor-like user interface. 8273 commands are entered by a two-character code followed by any parameters required by that command. When 8273 interrupts occur, the source of the interrupt is displayed along with any results associated with it. To gain a flavor of how the user/program interface operates, a sample output is shown in Figure 43. The 8273 program prompt character is a "-" and user inputs are underlined.

The "SO 05" implements the Set Operating Mode command with a parameter of 05H. This sets the Buffer and Flag Stream modes. "SS 01" sets the 8273 in NRZI mode using the Set Serial I/O Mode command. The next command specifies General Receiver with a receiver buffer size of 0100H bytes (B<sub>0</sub>=00, B<sub>1</sub>=01). The "TF" command causes the 8273 to transmit a frame containing an address field of C2H and control field of 11H. The information field is 001122. The "TF" command has a special format. The L<sub>0</sub> and L<sub>1</sub> parameters are computed from the number of information field bytes entered.

After the TF command is entered, the 8273 transmits the frame (assuming that the modem protocol is observed). After the closing flag, the 8273 interrupts the 8085. The 8085 reads the interrupt results and places them in a buffer. The software examines this buffer for new results and if new results exist, the source of the interrupt is displayed along with the results.

In this example, the 0DH result indicates a Frame Complete interrupt. There is only one result for a transmitter interrupt, the interrupt's trailing zero results were included to simplify programming.

The next event is a frame reception. The interrupt results are displayed in the order read from the 8273. The EOH indicates a General Receive interrupt with the last byte of the information field received on an 8-bit boundary. The 03 00 (R<sub>0</sub>, R<sub>1</sub>) results show that there are 3H bytes of information field received. The remaining two results indicate that the received frame had a C2H address field and a 34H control field. The 3 bytes of information field are displayed on the next line.

## 8273 MONITOR V1.2

```

- SO 05
- SS 01
- GR 00 01
- TF C2 11 00 11 22
-
TxINT  - 0D 00 00 00 00
RxINT  - E0 03 00 C2 34
FF EE DD
-
    
```

Figure 43. Sample 8273 Monitor I/O

Figures 44 through 51 show the flowcharts used for the 8273 program development. The actual program listing is included as Appendix A. Figure 44 is the main status poll loop. After all devices are initialized and a prompt character displayed, a loop is entered at LOOPIT. This loop checks for a change of status in the result buffer or if a keyboard character has been received by the 8251 or if a poll frame has been received. If any of these conditions are met, the program branches to the appropriate routine. Otherwise, the loop is traversed again.

The result buffer is implemented as a 255-byte circular buffer with two pointers: CNADR and LDADR. CNADR is the console pointer. It points to the next result to be displayed LDADR is the load pointer. It points to the next empty position in the buffer into which the interrupt handler places the next result. The same buffer is used for both transmitter and receiver results. LOOPIT examines these pointers to detect when CNADR is not equal to LDADR indicating that the buffer contains results which have not been displayed. When this occurs, the program branches to the DISPLY routine.

DISPLY determines the source of the undisplayed results by testing the first result. This first result is necessarily the interrupt result code. If this result is 0CH or greater, the result is from a transmitter interrupt. Otherwise it is from a receiver source. The source of the result code is then displayed on the console along with the next four results from the buffer. If the source was a transmitter interrupt, the routine merely repoints the pointer CNADR and returns to LOOPIT. For a receiver source, the receiver data buffer is displayed in addition to the receiver interrupt results before returning to LOOPIT.

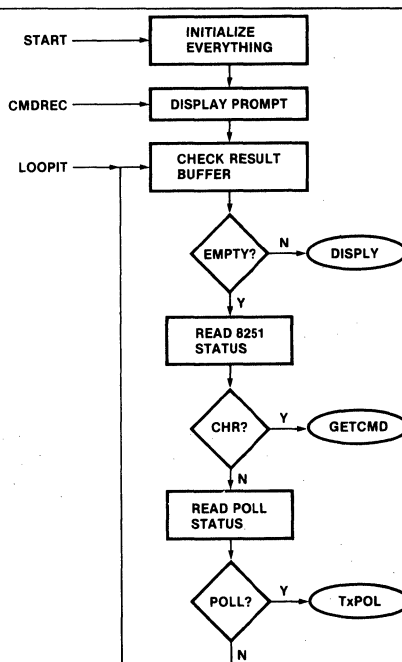


Figure 44. Main Status Poll Loop

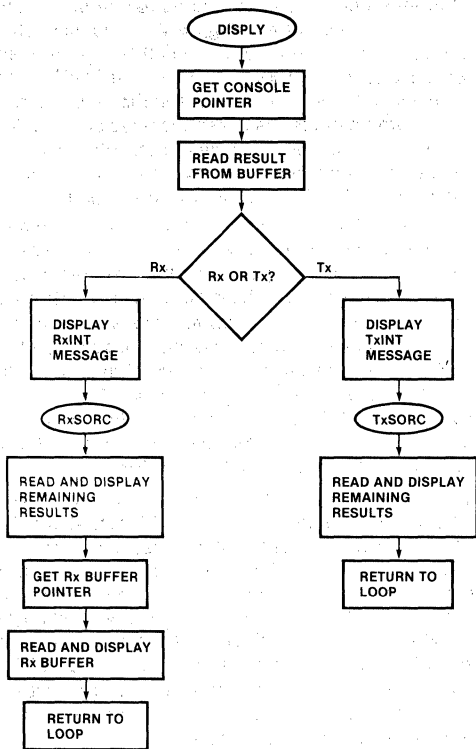


Figure 45. DISPLY Subroutine

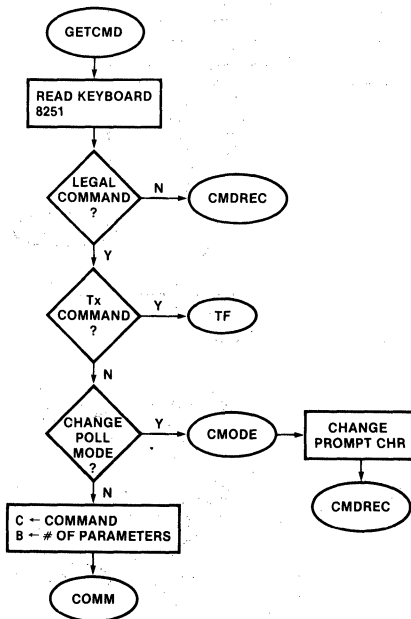


Figure 46. GETCMD Subroutine

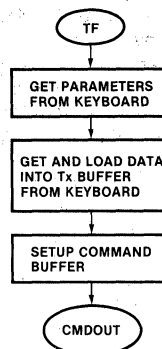


Figure 47. TF Subroutine

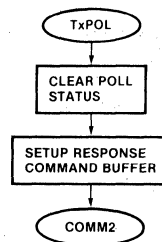


Figure 48. TxPOL Subroutine

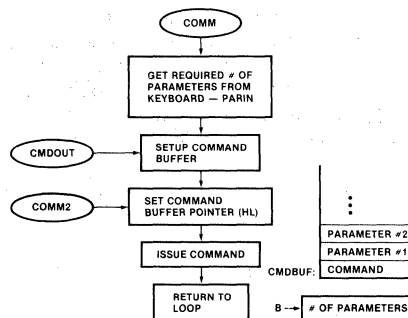


Figure 49. COMM Subroutine with Command Buffer Format

# APPLICATIONS

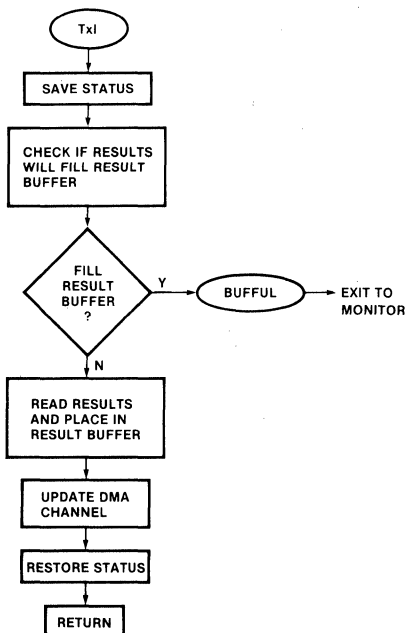


Figure 50. TxI (Transmitter Interrupt) Routine

If the result buffer pointers indicate an empty buffer, the 8251A is polled for a keyboard character. If the 8251 has a character, GETCMD is called. There the character is read and checked if legal. Illegal characters simply cause a reprompt. Legal characters indicate the start of a command input. Most commands are organized as two characters signifying the command action; i.e., GR — General Receive. The software recognizes the two character command code and takes the appropriate action. For non-Transmit type commands, the hex equivalent of the command is placed in the C register and the number of parameters associated with that command is placed in the B register. The program then branches to the COMM routine.

The COMM routine builds the command buffer by reading the required number of parameters from the keyboard and placing them at the buffer pointed at by CMDBUF. The routine at COMM2 then issues this command buffer to the 8273.

If a Transmit type command is specified, the command buffer is set up similarly to the COMM routine; however, since the information field data is entered from the keyboard, an intermediate routine, TF, is called. TF loads the transmit data buffer pointed at by TxBUF. It counts the number of data bytes entered and loads this number into the command buffer as L<sub>0</sub>, L<sub>1</sub>. The command is then issued to the 8273 by jumping to CMDOUT.

One command does not directly result in a command being issued to the 8273. This command, Z, operates a software flip-flop which selects whether the software will respond automatically to received polling frames. If

the Poll-Response mode is selected, the prompt character is changed to a '+'. If a frame is received which contains a prearranged poll control field, the memory location POLIN is made nonzero by the receiver interrupt handler. LOOPIT examines this location and if it is nonzero, causes a branch to the TxPOL routine. The TxPOL routine clears POLIN, sets a pointer to a special command buffer at CMDBUF1, and issues the command by way of the COMM2 entry in the COMM routine. The special command buffer contains the appropriate response frame for the poll frame received. These actions only occur when the Z command has changed the prompt to a '+'. If the prompt is normal '-', polling frames are displayed as normal frames and no response is transmitted. The Poll-Response mode was used during the IBM tests.

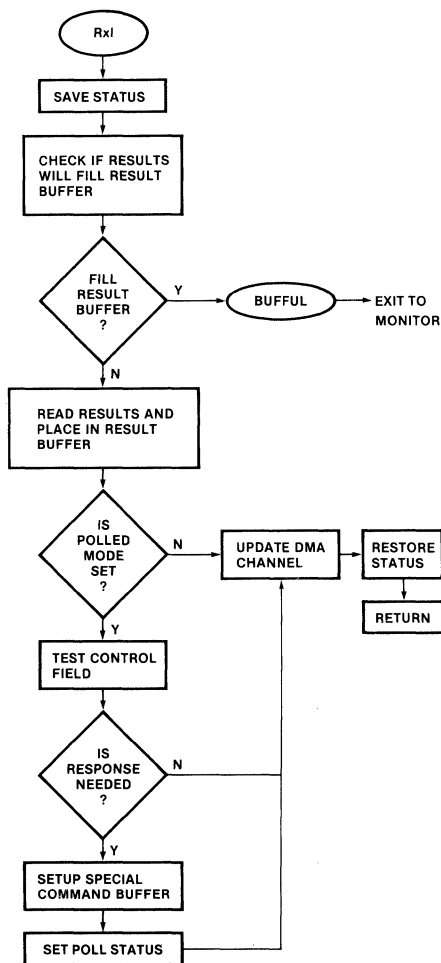


Figure 51. RxI (Receiver Interrupt) Routine

The final two software routines are the transmitter and receiver interrupt handlers. The transmit interrupt handler, TxI, simply saves the registers on the stack and checks if loading the result buffer will fill it. If the result buffer will overflow, the program is exited and control is passed to the SDK-85 monitor. If not, the results are read from the TxI/R register and placed in the result buffer at LDADR. The DMA pointers are then reset, the registers restored, and interrupts enabled. Execution then returns to the pre-interrupt location.

The receiver interrupt handler, RxI, is only slightly more complex. As in TxI, the registers are saved and the possibility of overflowing the result buffer is examined. If the result buffer is not full, the results are read from RxI/R and placed in the buffer. At this point the prompt character is examined to see if the Poll-Response mode is selected. If so, the control field is compared with two possible polling control fields. If there is a match, the

special command buffer is loaded and the poll indicator, POLIN, is made nonzero. If no match occurred, no action is taken. Finally, the receiver DMA buffer pointers are reset, the processor status restored, and interrupts are enabled. The RET instruction returns execution to the pre-interrupt location.

This completes the discussion of the 8273/8085 system design.

### CONCLUSION

This application note has covered the 8273 in some detail. The simple and low cost loop configuration was explored. And an 8273/8085 system was presented as a sample design illustrating the DMA/interrupt-driven interface. It is hoped that the major features of the 8273, namely the frame-level command structure and the Digital Phase Locked Loop, have been shown to be a valuable asset in an SDLC system design.

**APPENDIX**



# APPLICATIONS

## APPENDIX A

ASM80 :F1:RAYT73.SRC

IS15-II 8080/8085 MACRO ASSEMBLER, X108      MODULE    PAGE    1

LOC	OBJ	SEQ	SOURCE STATEMENT
		1	#NOPAGING MOD85 NOCOND
0000		2	TRUE EQU 00H ;00 FOR RAYTHEON
		3	; FF FOR SELF-TEST
0000		4	TRUE1 EQU 00H ;00 FOR NORMAL RESPONSE
		5	; FF FOR LOOP RESPONSE
0000		6	DEM EQU 00H ;00 FOR NO DEMO
		7	; FF FOR DEMO
		8	;
		9	;
		10	;GENERAL 8273 MONITOR WITH RAYTHEON POLL MODE ADDED
		11	;
		17	;
		18	;
		19	;COMMAND SUPPORTED ARE: RS - RESET SERIAL I/O MODE
		20	SS - SET SERIAL I/O MODE
		21	RO - RESET OPERATING MODE
		22	SO - SET OPERATING MODE
		23	RD - RECEIVER DISABLE
		24	GR - GENERAL RECEIVE
		25	SR - SELECTIVE RECEIVE
		26	TF - TRANSMIT FRAME
		27	AF - ABORT FRAME
		28	SP - SET PORT B
		29	RP - RESET PORT B
		30	RB - RESET ONE BIT DELAY (PAR = 7F)
		31	SB - SET ONE BIT DELAY (PAR = 80)
		32	SL - SELECTIVE LOOP RECEIVE
		33	TL - TRANSMIT LOOP
		34	Z - CHANGE MODES FLIP/FLOP
		38	;
		39	*****
		40	;
		41	;NOTE: 'SET' COMMANDS IMPLEMENT LOGICAL 'OR' FUNCTIONS
		42	'RESET' COMMANDS IMPLEMENT LOGICAL 'AND' FUNCTIONS
		43	;
		44	*****
		45	;
		46	;BUFFERED MODE MUST BE SELECTED WHEN SELECTIVE RECEIVE IS USED.
		47	;
		48	;COMMAND FORMAT IS: 'COMMAND (2 LTRS)' 'PAR.#1' 'PAR.#2' ETC.
		49	;
		50	;THE TRANSMIT FRAME COMMAND FORMAT IS: 'TF' 'A' 'C' 'BUFFER CONTENTS'.
		51	NO LENGTH COUNT IS NEEDED. BUFFER CONTENTS IS ENDED WITH A CR.
		52	;
		53	*****
		54	;
		55	;POLLED MODE: WHEN POLLED MODE IS SELECTED (DENOTED BY A '+' PROMPT), IF

# APPLICATIONS

```

56 ;          A SNRM-P OR RR(0)-P IS RECEIVED, A RESPONSE FRAME OF NSA-F
57 ;          OR RR(0)-F IS TRANSMITTED.  OTHER COMMANDS OPERATE NORMALLY.
62 ;
63 ;*****
64 ;
65 ;8273 EQUATES
66 ;
0090 67 STAT73 EQU 90H          ;STATUS REGISTER
0090 68 COMM73 EQU 90H        ;COMMAND REGISTER
0091 69 PARM73 EQU 91H        ;PARAMETER REGISTER
0091 70 RESL73 EQU 91H        ;RESULT REGISTER
0092 71 TXIR73 EQU 92H        ;TX INTERRUPT RESULT REGISTER
0093 72 RXIR73 EQU 93H        ;RX INTERRUPT RESULT REGISTER
0092 73 TEST73 EQU 92H        ;TEST MODE REGISTER
0020 74 CPBF EQU 20H          ;PARAMETER BUFFER FULL BIT
0004 75 TXINT EQU 04H         ;TX INTERRUPT BIT IN STATUS REGISTER
0008 76 RXINT EQU 08H         ;RX INTERRUPT BIT IN STATUS REGISTER
0001 77 TXIRA EQU 01H         ;TX INT RESULT AVAILABLE BIT
0002 78 RXIRA EQU 02H         ;RX INT RESULT AVAILABLE BIT
79 ;
80 ;8253 EQUATES
81 ;
009B 82 MODE53 EQU 9BH        ;8253 MODE WORD REGISTER
009C 83 CNT053 EQU 9CH        ;COUNTER 0 REGISTER
009D 84 CNT153 EQU 9DH        ;COUNTER 1 REGISTER
009E 85 CNT253 EQU 9EH        ;COUNTER 2 REGISTER
000C 86 COBR EQU 000CH        ;CONSOLE BAUD RATE (2400)
0036 87 MDCNT0 EQU 36H        ;MODE FOR COUNTER 0
00B6 88 MDCNT2 EQU 0B6H       ;MODE FOR COUNTER 2
2017 89 LKBR1 EQU 2017H       ;8273 BAUD RATE LSB ADR
2018 90 LKBR2 EQU 2018H       ;8273 BAUD RATE MSB ADR
91 ;
92 ;BAUD RATE TABLE:      BAUD RATE      LKBR1  LKBR2
93 ;          *****          *****  *****
94 ;          9600             2E        00
95 ;          4800             5C        00
96 ;          2400             89        00
97 ;          1200             72        01
98 ;          600              E5        02
99 ;          300              C9        05
100 ;
101 ;
102 ;8257 EQUATES
103 ;
00A8 104 MODE57 EQU 0A8H      ;8257 MODE PORT
00A0 105 CH0ADR EQU 0A0H      ;CH0 (RX) ADR REGISTER
00A1 106 CH0TC EQU 0A1H      ;CH0 TERMINAL COUNT REGISTER
00A2 107 CH1ADR EQU 0A2H      ;CH1 (TX) ADR REGISTER
00A3 108 CH1TC EQU 0A3H      ;CH1 TERMINAL COUNT REGISTER
00A8 109 STAT57 EQU 0A8H      ;STATUS REGISTER
8200 110 RXBUF EQU 8200H      ;RX BUFFER START ADDRESS
8000 111 TXBUF EQU 8000H      ;TX BUFFER START ADDRESS
0062 112 DROMA EQU 62H        ;DISABLE RX DMA CHANNEL, TX STILL ON
41FF 113 RXTC EQU 41FFH      ;TERMINAL COUNT AND MODE FOR RX CHANNEL
0063 114 ENDMA EQU 63H        ;ENABLE BOTH TX AND RX CHANNELS-EXT. WR. TX STOP
0061 115 DTDMA EQU 61H        ;DISABLE TX DMA CHANNEL, RX STILL ON
81FF 116 TXXC EQU 81FFH      ;TERMINAL COUNT AND MODE FOR TX CHANNEL
117 ;

```

# APPLICATIONS

```

118 ; 8251A EQUATES
119 ;
0089      120 CNTL51 EQU   89H           ; CONTROL WORD REGISTER
0089      121 STAT51 EQU   89H           ; STATUS REGISTER
0088      122 TXD51  EQU   88H           ; TX DATA REGISTER
0088      123 RXD51  EQU   88H           ; RX DATA REGISTER
00CE      124 MDE51  EQU   0CEH          ; MODE 16X 2 STOP, NO PARITY
0027      125 CMD51  EQU   27H           ; COMMAND, ENABLE TX&RX
0002      126 RDY   EQU   02H           ; RXRDY BIT
127 ;
128 ; MONITOR SUBROUTINE EQUATES
129 ;
061F      130 GETCH  EQU   061FH          ; GET CHR FROM KEYBOARD, ASCII IN CH
05F8      131 ECHO   EQU   05F8H          ; ECHO CHR TO DISPLAY
075E      132 VALDG  EQU   075EH          ; CHECK IF VALID DIGIT, CARRY SET IF VALID
058B      133 CNVBN  EQU   058BH          ; CONVERTS ASCII TO HEX
05EB      134 CRLF   EQU   05EBH          ; DISPLAY CR, HENCE LF TOO
06C7      135 NMOU   EQU   06C7H          ; CONVERT BYTE TO 2 ASCII CHR AND DISPLAY
136 ;
137 ; MISC EQUATES
138 ;
20C0      139 STKSRT EQU   20C0H          ; STACK START
0003      140 CNTLC  EQU   03H           ; CNTL-C EQUIVALENT
0008      141 MONTR  EQU   0008H          ; MONITOR
2000      142 CMDBUF EQU   2000H          ; START OF COMMAND BUFFER
2020      143 CMDBF1 EQU   2020H          ; POLL MODE SPECIAL TX COMMAND BUFFER
0000      144 CR     EQU   00H           ; ASCII CR
000A      145 LF     EQU   0AH           ; ASCII LF
2004      146 RST75  EQU   2004H          ; RST7.5 JUMP ADDRESS
20CE      147 RST65  EQU   20CEH          ; RST6.5 JUMP ADDRESS
2010      148 LOADR  EQU   2010H          ; RESULT BUFFER LOAD POINTER STORAGE
2013      149 CNADR  EQU   2013H          ; RESULT BUFFER CONSOLE POINTER STORAGE
2800      150 RESBUF  EQU   2800H          ; RESULT BUFFER START - 255 BYTES
0093      151 SNRMP  EQU   93H           ; SNRM-P CONTROL CODE
0011      152 RR0P  EQU   11H           ; RR(0)-P CONTROL CODE
0073      153 NSAF  EQU   73H           ; NSR-F CONTROL CODE
0011      154 RR0F  EQU   11H           ; RR(0)-F CONTROL CODE
2015      155 PRMPT  EQU   2015H          ; PRMPT STORAGE
2016      156 POLIN  EQU   2016H          ; POLL MODE SELECTION INDICATOR
2027      157 DEMODE  EQU   2027H          ; DEMO MODE INDICATOR
161 ;
162 ; *****
163 ;
164 ; RAM STORAGE DEFINITIONS:
165 ;      LOC      DEF
166 ;      ---      ---
167 ;      2000-200F  COMMAND BUFFER
168 ;      2010-2011  RESULT BUFFER LOAD POINTER
169 ;      2013-2014  RESULT BUFFER CONSOLE POINTER
170 ;      2015      PROMPT CHARACTER STORAGE
171 ;      2016      POLL MODE INDICATOR
172 ;      2017      BAUD RATE LSB FOR SELF-TEST
173 ;      2018      BAUD RATE MSB FOR SELF-TEST
177 ;      2019      SPARE
179 ;      2020-2026  RESPONSE COMMAND BUFFER FOR POLL MODE
180 ;      2800-28FF  RESULT BUFFER
181 ;
182 ; *****

```

# APPLICATIONS

```

183 ;
184 ;PROGRAM START
185 ;
186 ;INITIALIZE 8253, 8257, 8251A, AND RESET 8273.
187 ;ALSO SET NORMAL MODE, AND PRINT SIGNON MESSAGE
188 ;
0800 189      ORG      900H
190
0800 310020 191 START: LXI    SP, STKSRT      ;INITIALIZE SP
0803 3E36   192      MVI    A, MDCNT0    ;8253 MODE SET
0805 D39B   193      OUT    MODE53        ;8253 MODE PORT
0807 3A1720 194      LDA    LKBR1          ;GET 8273 BAUD RATE LSB
080A D39C   195      OUT    CNT053        ;USING COUNTER 0 AS BAUD RATE GEN
080C 3A1820 196      LDA    LKBR2          ;GET 8273 BAUD RATE MSB
080F D39C   197      OUT    CNT053        ;COUNTER 0
0811 CD1A0B 198      CALL   RXDMA           ;INITIALIZE 8257 RX DMA CHANNEL
0814 CD350B 199      CALL   TXDMA           ;INITIALIZE 8257 TX DMA CHANNEL
0817 3E01   200      MVI    A, 01H        ;OUTPUT 1 FOLLOWED BY A 0
0819 D392   201      OUT    TEST73       ;TO TEST MODE REGISTER
081B 3E00   202      MVI    A, 00H        ;TO RESET THE 8273
081D D392   203      OUT    TEST73       ;
081F 3E2D   204      MVI    A, ' '        ;NORMAL MODE PROMPT CHR
0821 321520 205      STA    PRMPT           ;PUT IN STORAGE
0824 3E00   206      MVI    A, 00H        ;TX POLL RESPONSE INDICATOR
0826 321620 207      STA    POLIN          ;0 MEANS NO SPECIAL TX
0829 322720 208      STA    DEMODE         ;CLEAR DEMO MODE
082C 21A30C 212      LXI    H, SIGNON      ;SIGNON MESSAGE ADR
082F CD920C 213      CALL   TYMSG          ;DISPLAY SIGNON
214 ;
215 ;MONITOR USES JUMPS IN RAM TO DIRECT INTERRUPTS
216 ;
0832 21D420 217      LXI    H, RST75        ;RST7.5 JUMP LOCATION USED BY MONITOR
0835 01000C 218      LXI    B, RXI          ;ADDRESS OF RX INT ROUTINE
0838 36C3   219      MVI    M, 0C3H       ;LOAD 'JMP' OPCODE
083A 23     220      INX    H          ;INC POINTER
083B 71     221      MOV    M, C        ;LOAD RXI LSB
083C 23     222      INX    H          ;INC POINTER
083D 70     223      MOV    M, B        ;LOAD RXI MSB
083E 21CE20 224      LXI    H, RST65        ;RST6.5 JUMP LOCATION USED BY MONITOR
0841 01CE0C 225      LXI    B, TXI          ;ADDRESS OF TX INT ROUTINE
0844 36C3   226      MVI    M, 0C3H       ;LOAD 'JMP' OPCODE
0846 23     227      INX    H          ;INC POINTER
0847 71     228      MOV    M, C        ;LOAD TXI LSB
0848 23     229      INX    H          ;INC POINTER
0849 70     230      MOV    M, B        ;LOAD TXI MSB
084A 3E18   231      MVI    A, 18H        ;GET SET TO RESET INTERRUPTS
084C 30     232      SIM             ;RESET INTERRUPTS
084D FB     233      EI              ;ENABLE INTERRUPTS
234 ;
235 ;INITIALIZE BUFFER POINTER
236 ;
237 ;
084E 210028 238      LXI    H, RESBUF        ;SET RESULT BUFFER POINTERS
0851 221320 239      SHLD  CNADR          ;RESULT CONSOLE POINTER
0854 221020 240      SHLD  LDADR          ;RESULT LOAD POINTER
241 ;
242 ;MAIN PROGRAM LOOP - CHECKS FOR CHANGE IN RESULT POINTERS, USART STATUS,
243 ;OR POLL STATUS

```

# APPLICATIONS

	244 ;			
0857 CDEB05	245 CMDREC:	CALL	CRLF	; DISPLAY CR
085A 3A1520	246	LDA	PRMPT	; GET CURRENT PROMPT CHR
085D 4F	247	MOV	C, A	; MOVE TO C
085E CDF805	248	CALL	ECHO	; DISPLAY IT
0861 2A1320	249 LOOPIT:	LHLD	CNADR	; GET CONSOLE POINTER
0864 7D	250	MOV	A, L	; SAVE POINTER LSB
0865 2A1020	251	LHLD	LDADR	; GET LOAD POINTER
0868 BD	252	CMP	L	; SAME LSB?
0869 C2390A	253	JNZ	DISPY	; NO, RESULTS NEED DISPLAYING
086C DB09	259	IN	STAT51	; YES, CHECK KEYBOARD
086E E602	260	ANI	RDY	; CHR RECEIVED?
0870 C27D08	261	JNZ	GETCMD	; MUST BE CHR SO GO GET IT
0873 3A1620	262	LDA	POLIN	; GET POLL MODE STATUS
0876 A7	263	ANA	A	; IS IT 0?
0877 C24C09	264	JNZ	TXPOL	; NO, THEN POLL OCCURRED
087A C36108	265	JMP	LOOPIT	; YES, TRY AGAIN
	266 ;			
	267 ;			
	268 ;	COMMAND RECOGNIZER ROUTINE		
	269 ;			
	270 ;			
087D CD1F06	271 GETCMD:	CALL	GETCH	; GET CHR
0880 CDF805	272	CALL	ECHO	; ECHO IT
0883 79	273	MOV	A, C	; SETUP FOR COMPARE
0884 FE52	274	CPI	'R'	; R?
0886 CAFF08	275	JZ	RDWN	; GET MORE
0889 FE53	276	CPI	'S'	; S?
088B CAD708	277	JZ	SDWN	; GET MORE
088E FE47	278	CPI	'G'	; G?
0890 CAFF08	279	JZ	GDWN	; GET MORE
0893 FE54	280	CPI	'T'	; T?
0895 CA0E09	281	JZ	TOWN	; GET MORE
0898 FE41	282	CPI	'A'	; A?
089A CA2209	283	JZ	ADWN	; GET MORE
089D FE5A	284	CPI	'Z'	; Z?
089F CA3109	285	JZ	CNODE	; YES, GO CHANGE MODE
08A2 FE03	290	CPI	CNTLC	; CNTL-C?
08A4 CA0808	291	JZ	MONTOR	; EXIT TO MONITOR
08A7 0E3F	292 ILLEG:	MVI	C, '?'	; PRINT ?
08A9 CDF805	293	CALL	ECHO	; DISPLAY IT
08AC C35708	294	JMP	CMDREC	; LOOP FOR COMMAND
	295			
08AF CD1F06	296 RDWN:	CALL	GETCH	; GET NEXT CHR
08B2 CDF805	297	CALL	ECHO	; ECHO IT
08B5 79	298	MOV	A, C	; SETUP FOR COMPARE
08B6 FE4F	299	CPI	'O'	; O?
08B8 CAD009	300	JZ	ROCMD	; RO COMMAND
08BB FE53	301	CPI	'S'	; S?
08BD CA6709	302	JZ	RSCMD	; RS COMMAND
08C0 FE44	303	CPI	'D'	; D?
08C2 CA7109	304	JZ	RDCMD	; RD COMMAND
08C5 FE50	305	CPI	'P'	; P?
08C7 CAD009	306	JZ	RPCMD	; RP COMMAND
08CA FE52	307	CPI	'R'	; R?
08CC CA0808	308	JZ	START	; START OVER
08CF FE42	309	CPI	'B'	; B?
08D1 CA7B09	310	JZ	RBCMD	; RB COMMAND

# APPLICATIONS

```

08D4 C3A708      311      JMP      ILLEG      ; ILLEGAL, TRY AGAIN
                  312
08D7 CD1F06     313 SDWN:  CALL     GETCH      ; GET NEXT CHR
08DA CDF005     314      CALL     ECHO       ; ECHO IT
08DD 78         315      MOV      A, B       ; SETUP FOR COMPARE
08DE FE4F       316      CPI      '0'       ; 0?
08E0 CAA009     317      JZ       SOCMD      ; SO COMMAND
08E3 FE53       318      CPI      'S'       ; S?
08E5 CAB009     319      JZ       SSCMD      ; SS COMMAND
08E8 FE52       320      CPI      'R'       ; R?
08EA CABA09     321      JZ       SRCMD      ; SR COMMAND
08ED FE50       322      CPI      'P'       ; P?
08EF CAE209     323      JZ       SPCMD      ; SP COMMAND
08F2 FE42       324      CPI      'B'       ; B?
08F4 CA8509     325      JZ       SBCMD      ; SB COMMAND
08F7 FE4C       326      CPI      'L'       ; L?
08F9 CA8F09     327      JZ       SLCMD      ; SL COMMAND
08FC C3A708     328      JMP      ILLEG      ; ILLEGAL, TRY AGAIN
                  329
08FF CD1F06     330 GDWN:  CALL     GETCH      ; GET NEXT CHR
0902 CDF005     331      CALL     ECHO       ; ECHO IT
0905 78         332      MOV      A, B       ; SETUP FOR COMPARE
0906 FE52       333      CPI      'R'       ; R?
0908 CAC409     334      JZ       GRCMD      ; GR COMMAND
090B C3A708     335      JMP      ILLEG      ; ILLEGAL, TRY AGAIN
                  336
090E CD1F06     337 TDWN:  CALL     GETCH      ; GET NEXT CHR
0911 CDF005     338      CALL     ECHO       ; ECHO IT
0914 78         339      MOV      A, B       ; SETUP FOR COMPARE
0915 FE46       340      CPI      'F'       ; F?
0917 CAE009     341      JZ       TFCMD      ; TF COMMAND
091A FE4C       342      CPI      'L'       ; L?
091C CA9909     343      JZ       TLCMD      ; TL COMMAND
091F C3A708     344      JMP      ILLEG      ; ILLEGAL, TRY AGAIN
                  345
0922 CD1F06     346 ADWN:  CALL     GETCH      ; GET NEXT CHR
0925 CDF005     347      CALL     ECHO       ; ECHO IT
0928 78         348      MOV      A, B       ; SETUP FOR COMPARE
0929 FE46       349      CPI      'F'       ; F?
092B CACE09     350      JZ       AFCMD      ; AF COMMAND
092E C3A708     351      JMP      ILLEG      ; ILLEGAL, TRY AGAIN
                  352 ;
                  353 ; RESET POLL MODE RESPONSE - CHANGE PROMPT CHR AS INDICATOR
                  354 ;
0931 F3         355 CMODE:  DI          ; DISABLE INTERRUPTS
0932 3A1520     356      LDA      PRMPT      ; GET CURRENT PROMPT
0935 FE2D       357      CPI      '-'       ; NORMAL MODE?
0937 C24309     358      JNZ     SW          ; NO, CHANGE IT
093A 3E2B       359      MVI     A, '+'     ; NEW PROMPT
093C 321520     360      STA      PRMPT      ; STORE NEW PROMPT
093F FB         365      EI          ; ENABLE INTERRUPTS
0940 C35708     366      JMP      CHDREC     ; RETURN TO LOOP
0943 3E2D       367 SW:   MVI     A, '-'     ; NEW PROMPT CHR
0945 321520     368      STA      PRMPT      ; STORE IT
0948 FB         369      EI          ; ENABLE INTERRUPTS
0949 C35708     370      JMP      CHDREC     ; RETURN TO LOOP
                  371 ;
                  372 ;

```

# APPLICATIONS

```

373 ; TRANSMIT ANSWER TO POLL SETUP
374 ;
094C 3E00 382 TXPOL: MVI A, 00H ; CLEAR POLL INDICATOR
094E 321620 384 STA POLIN ; INDICATOR ADR
0951 216100 385 LXI H, LOOPIT ; SETUP STACK FOR COMMAND OUTPUT
0954 E5 386 PUSH H ; PUT RETURN TO CMDREC ON STACK
0955 0604 387 MVI B, 04H ; GET # OF PARAMETERS READY
0957 212020 388 LXI H, CMDBF1 ; POINT TO SPECIAL BUFFER
095A C3FF00 389 JMP COMM2 ; JUMP TO COMMAND OUTPUTER
390 ;
391 ;
392 ;
393 ; COMMAND IMPLEMENTING ROUTINES
394 ;
395 ;
396 ; RO - RESET OPERATING MODE
397 ;
095D 0601 398 ROCMD: MVI B, 01H ; # OF PARAMETERS
095F 0E51 399 MVI C, 51H ; COMMAND
0961 CDE500 400 CALL COMM ; GET PARAMETERS AND ISSUE COMMAND
0964 C35700 401 JMP CMDREC ; GET NEXT COMMAND
402 ;
403 ; RS - RESET SERIAL I/O MODE COMMAND
404 ;
0967 0601 405 RSCMD: MVI B, 01H ; # OF PARAMETERS
0969 0E60 406 MVI C, 60H ; COMMAND
096B CDE500 407 CALL COMM ; GET PARAMETERS AND ISSUE COMMAND
096E C35700 408 JMP CMDREC ; GET NEXT COMMAND
409 ;
410 ; RD - RECEIVER DISABLE COMMAND
411 ;
0971 0600 412 RDCMD: MVI B, 00H ; # OF PARAMETERS
0973 0EC5 413 MVI C, 0C5H ; COMMAND
0975 CDE500 414 CALL COMM ; ISSUE COMMAND
0978 C35700 415 JMP CMDREC ; GET NEXT COMMAND
416 ;
417 ; RB - RESET ONE BIT DELAY COMMAND
418 ;
097B 0601 419 RBCMD: MVI B, 01H ; # OF PARAMETERS
097D 0E64 420 MVI C, 64H ; COMMAND
097F CDE500 421 CALL COMM ; GET PARAMETER AND ISSUE COMMAND
0982 C35700 422 JMP CMDREC ; GET NEXT COMMAND
423 ;
424 ; SB - SET ONE BIT DELAY COMMAND
425 ;
0985 0601 426 SBCMD: MVI B, 01H ; # OF PARAMETERS
0987 0EA4 427 MVI C, 0A4H ; COMMAND
0989 CDE500 428 CALL COMM ; GET PARAMETER AND ISSUE COMMAND
098C C35700 429 JMP CMDREC ; GET NEXT COMMAND
430 ;
431 ; SL - SELECTIVE LOOP RECEIVE COMMAND
432 ;
098F 0604 433 SLCMD: MVI B, 04H ; # OF PARAMETERS
0991 0EC2 434 MVI C, 0C2H ; COMMAND
0993 CDE500 435 CALL COMM ; GET PARAMETERS AND ISSUE COMMAND
0996 C35700 436 JMP CMDREC ; GET NEXT COMMAND
437 ;
438 ; TL - TRANSMIT LOOP COMMAND

```

# APPLICATIONS

```

439 ;
0999 210020 440 TLCMD: LXI H,CMDBUF ;SET COMMAND BUFFER POINTER
099C 0602 441 MVI B,02H ;LOAD PARAMETER COUNTER
099E 36CA 442 MVI M,0CAH ;LOAD COMMAND INTO BUFFER
09A0 210220 443 LXI H,CMDBUF+2 ;POINT AT ADR AND CNTL POSITIONS
09A3 C3F609 444 JMP TFCMD1 ;FINISH OFF COMMAND IN TF ROUTINE
445 ;
446 ;SO - SET OPERATING MODE COMMAND
447 ;
09A6 0601 448 SOCMD: MVI B,01H ;# OF PARAMETERS
09A8 0E91 449 MVI C,91H ;COMMAND
09AA CDE50A 450 CALL COMM ;GET PARAMETER AND ISSUE COMMAND
09AD C35708 451 JMP CMDREC ;GET NEXT COMMAND
452 ;
453 ;SS - SET SERIAL I/O COMMAND
454 ;
09B0 0601 455 SSCMD: MVI B,01H ;# OF PARAMETERS
09B2 0EA0 456 MVI C,0A0H ;COMMAND
09B4 CDE50A 457 CALL COMM ;GET PARAMETER AND ISSUE COMMAND
09B7 C35708 458 JMP CMDREC ;GET NEXT COMMAND
459 ;
460 ;SR - SELECTIVE RECEIVE COMMAND
461 ;
09BA 0604 462 SRCMD: MVI B,04H ;# OF PARAMETERS
09BC 0EC1 463 MVI C,0C1H ;COMMAND
09BE CDE50A 464 CALL COMM ;GET PARAMETERS AND ISSUE COMMAND
09C1 C35708 465 JMP CMDREC ;GET NEXT COMMAND
466 ;
467 ;GR - GENERAL RECEIVE COMMAND
468 ;
09C4 0602 469 GRCMD: MVI B,02H ;NO PARAMETERS
09C6 0EC0 470 MVI C,0C0H ;COMMAND
09C8 CDE50A 471 CALL COMM ;ISSUE COMMAND
09CB C35708 472 JMP CMDREC ;GET NEXT COMMAND
473 ;
474 ;AF - ABORT FRAME COMMAND
475 ;
09CE 0600 476 AFCMD: MVI B,00H ;NO PARAMETERS
09D0 0ECC 477 MVI C,0CCH ;COMMAND
09D2 CDE50A 478 CALL COMM ;ISSUE COMMAND
09D5 C35708 479 JMP CMDREC ;GET NEXT COMMAND
480 ;
481 ;RP - RESET PORT COMMAND
482 ;
09D8 0601 483 RPCMD: MVI B,01H ;# OF PARAMETERS
09DA 0E63 484 MVI C,63H ;COMMAND
09DC CDE50A 485 CALL COMM ;GET PARAMETER AND ISSUE COMMAND
09DF C35708 486 JMP CMDREC ;GET NEXT COMMAND
487 ;
488 ;SP - SET PORT COMMAND
489 ;
09E2 0601 490 SPCMD: MVI B,01H ;# OF PARAMETERS
09E4 0EA3 491 MVI C,0A3H ;COMMAND
09E6 CDE50A 492 CALL COMM ;GET PARAMETER AND ISSUE COMMAND
09E9 C35708 493 JMP CMDREC ;GET NEX COMMAND
494 ;
495 ;TF - TRANSMIT FRAME COMMAND
496 ;

```



# APPLICATIONS

```

09EC 210020    497 TFCMD: LXI    H, CMBUF    ; SET COMMAND BUFFER POINTER
09EF 0602     498     MVI    B, 02H        ; LOAD PARAMETER COUNTER
09F1 36C8     499     MVI    M, 0C8H        ; LOAD COMMAND INTO BUFFER
09F3 210220    500     LXI    H, CMBUF+2    ; POINT AT ADR AND CNTL POSITIONS
09F6 78       501 TFCMD1: MOV    A, B        ; TEST PARAMETER COUNT
09F7 A7       502     ANA    A            ; IS IT 0?
09F8 CA070A    503     JZ     TBUFL        ; YES, LOAD TX DATA BUFFER
09FB CDAD0A    504     CALL   PARIN        ; GET PARAMETER
09FE DAA708    505     JC     ILLEG        ; ILLEGAL CHR RETURNED
0A01 23       506     INX   H            ; INC COMMAND BUFFER POINTER
0A02 05       507     DCR   B            ; DEC PARAMETER COUNTER
0A03 77       508     MOV    M, A        ; LOAD PARAMETER INTO COMMAND BUFFER
0A04 C3F609    509     JMP    TFCMD1        ; GET NEXT PARAMETER
                    510
0A07 210080    511 TBUFL: LXI    H, TXBUF    ; LOAD TX DATA BUFFER POINTER
0A0A 010000    512     LXI    B, 0000H    ; CLEAR BC - BYTE COUNTER
0A0D C5       513 TBUFL1: PUSH   B        ; SAVE BYTE COUNTER
0A0E CDAD0A    514     CALL   PARIN        ; GET DATA, ALIAS PARAMETER
0A11 DAB00A    515     JC     ENDCHK        ; MAYBE END IF ILLEGAL
0A14 77       516     MOV    M, A        ; LOAD DATA BYTE INTO BUFFER
0A15 23       517     INX   H            ; INC BUFFER POINTER
0A16 C1       518     POP   B            ; RESTORE BYTE COUNTER
0A17 03       519     INX   B            ; INC BYTE COUNTER
0A18 C30D0A    520     JMP    TBUFL1        ; GET NEXT DATA
0A1B FE0D     521 ENDCHK: CPI    CR        ; RETURNED ILLEGAL CHR CR?
0A1D CA240A    522     JZ     TBUFL        ; YES, THEN TX BUFFER FULL
0A20 C1       523     POP   B            ; RESTORE B TO SAVE STACK
0A21 C3A708    524     JMP    ILLEG        ; ILLEGAL CHR
0A24 C1       525 TBUFL: POP   B            ; RESTORE BYTE COUNTER
0A25 210120    526     LXI    H, CMBUF+1    ; POINT INTO COMMAND BUFFER
0A28 71       527     MOV    M, C        ; STORE BYTE COUNT LSB
0A29 23       528     INX   H            ; INC POINTER
0A2A 70       529     MOV    M, B        ; STORE BYTE COUNT MSB
0A2B 0604     530     MVI    B, 04H        ; LOAD PARAMETER COUNT INTO B
0A2D 21300A    531     LXI    H, TFRET        ; GET RETURN ADR FOR THIS ROUTINE
0A30 C5       532     PUSH  B            ; PUSH ONCE
0A31 E3       533     XTHL        ; PUT RETURN ON STACK
0A32 C5       534     PUSH  B            ; PUSH IT SO CMDOUT CAN USE IT
0A33 C3FB0A    535     JMP    CMDOUT        ; ISSUE COMMAND
0A36 C35708    536 TFRET: JMP    CMDREC        ; GET NEXT COMMAND
                    537 ;
                    538 ;
539 ; ROUTINE TO DISPLAY RESULT IN RESULT BUFFER WHEN LOAD AND CONSOLE
540 ; POINTERS ARE DIFFERENT.
541 ;
542 ;
0A39 1605     543 DISPY: MVI    D, 05H        ; D IS RESULT COUNTER
0A3B 2A1320    544     LHLD  CNADR        ; GET CONSOLE POINTER
0A3E E5       545     PUSH  H            ; SAVE IT
0A3F 7E       546     MOV    A, M        ; GET RESULT IC
0A40 E61F     547     ANI    1FH          ; LIMIT TO RESULT CODE
0A42 FE0C     548     CPI    0CH          ; TEST IF RX OR TX SOURCE
0A44 DA620A    549     JC     RXSORC        ; CARRY, THEN RX SOURCE
0A47 21C30C    550 TXSORC: LXI    H, TXMSG        ; TX INT MESSAGE
0A4A CD920C    551     CALL   TVMSG        ; DISPLAY IT
0A4D E1       552 DISPY2: POP   H            ; RESTORE CONSOLE POINTER
0A4E 7E       553 DISPY1: MOV    A, M        ; GET RESULT
0A4F CDC706    554     CALL   NMOUT        ; CONVERT AND DISPLAY

```

# APPLICATIONS

```

0A52 0E20      555      MVI      C, '/'      ; SP CHR
0A54 CDF805    556      CALL     ECHO         ; DISPLAY IT
0A57 2C        557      INR      L           ; INC BUFFER POINTER
0A58 15        558      DCR      D           ; DEC RESULT COUNTER
0A59 C24E0A    559      JNZ      DISPY1      ; NOT DONE
0A5C 221320    560      SHLD    CNADR        ; UPDATE CONSOLE POINTER
0A5F C35708    561      JMP      CMDREC       ; RETURN TO LOOP
562 ;
563 ;
564 ; RECEIVER SOURCE - DISPLAY RESULTS AND RECEIE BUFFER CONTENTS
565 ;
566 ;
0A62 21B80C    567      RXSORC: LXI     H, RXIMSG      ; RX INT MESSAGE ADR
0A65 C0920C    568      CALL    TYMSG       ; DISPLAY MESSAGE
0A68 E1        569      POP     H           ; RESTORE CONSOLE POINTER
0A69 7E        570      RXS1:  MOV     A, M        ; RETRIEVE RESULT FROM BUFFER
0A6A CDC706    571      CALL    NMOUT       ; CONVERT AND DISPLAY IT
0A6D 0E20      572      MVI     C, '/'      ; ASCII SP
0A6F CDF805    573      CALL    ECHO         ; DISPLAY IT
0A72 2C        574      INR     L           ; INC CONSOLE POINTER
0A73 15        575      DCR     D           ; DEC RESULT COUNTER
0A74 7A        576      MOV     A, D        ; GET SET TO TEST COUNTER
0A75 FE04      577      CPI     04H        ; IS THE RESULT R0?
0A77 CA820A    578      JZ      R0PT       ; YES, GO SAVE IT
0A7A FE03      579      CPI     03H        ; IS THE RESULT R1?
0A7C CA870A    580      JZ      R1PT       ; YES, GO SAVE IT
0A7F A7        581      RXS2:  ANA     A           ; TEST RESULT COUNTER
0A80 C2690A    582      JNZ     RXS1        ; NOT DONE YET, GET NEXT RESULT
0A83 221320    583      SHLD   CNADR        ; DONE, SO UPDATE CONSOLE POINTER
0A86 CDEB05    584      CALL    CRLF        ; DISPLAY CR
0A89 210082    585      LXI     H, RXBUF     ; POINT AT RX BUFFER
0A8C C1        586      POP     B           ; RETRIEVE RECEIVED COUNT
0A8D 78        587      RXS3:  MOV     A, B        ; IS COUNT 0?
0A8E B1        588      ORA     C           ;
0A8F CA5708    589      JZ      CMDREC       ; YES, GO BACK TO LOOP
0A92 7E        590      MOV     A, M        ; NO, GET CHR
0A93 C5        591      PUSH    B           ; SAVE BC
0A94 CDC706    592      CALL    NMOUT       ; CONVERT AND DISPLAY CHR
0A97 0E20      593      MVI     C, '/'      ; ASCII SP
0A99 CDF805    594      CALL    ECHO         ; DISPLAY IT TO SEPARATE DATA
0A9C C1        595      POP     B           ; RESTORE BC
0A9D 0B        596      DCX     B           ; DEC COUNT
0A9E 23        597      INX     H           ; INC POINTER
0A9F C38D0A    598      JMP     RXS3        ; GET NEXT CHR
599 ;
0AA2 4E        600      R0PT:  MOV     C, M        ; GET R0 FOR RESULT BUFFER
0AA3 C5        601      PUSH    B           ; SAVE IT
0AA4 C37F0A    602      JMP     RXS2        ; RETURN
603 ;
0AA7 C1        604      R1PT:  POP     B           ; GET R0
0AA8 46        605      MOV     B, M        ; GET R1 FOR RESULT BUFFER
0AA9 C5        606      PUSH    B           ; SAVE IT
0AAA C37F0A    607      JMP     RXS2        ;
608 ;
609 ;
610 ;
611 ; PARAMETER INPUT - PARAMETER RETURNED IN E REGISTER
612 ;

```

# APPLICATIONS

```

613 ;
0A0D C5      614 PARIN:  PUSH  B           ;SAVE BC
0A0E 1601    615      MVI  D,01H        ;SET CHR COUNTER
0A00 CD1F06  616      CALL  GETCH          ;GET CHR
0A03 CDF805  617      CALL  ECHO          ;ECHO IT
0A06 79      618      MOV   A,C           ;PUT CHR IN A
0A07 FE20    619      CPI   ' '          ;SP?
0A09 C2E00A  620      JNZ  PARIN1        ;NO, ILLEGAL, TRY AGAIN
0A0C CD1F06  621 PARIN3:  CALL  GETCH          ;GET CHR OF PARAMETER
0A0F CDF805  622      CALL  ECHO          ;ECHO IT
0A02 CD5E07  623      CALL  VALDG         ;IS IT A VALID CHR?
0A05 D2E00A  624      JNC  PARIN1        ;NO, TRY AGAIN
0A08 C0BB05  625      CALL  CNVBN         ;CONVERT IT TO HEX
0A0B 4F      626      MOV   C,A           ;SAVE IT IN C
0A0C 7A      627      MOV   A,D           ;GET CHR COUNTER
0A0D A7      628      ANA   A           ;IS IT 0?
0A0E CADC0A  629      JZ   PARIN2        ;YES, DONE WITH THIS PARAMETER
0A01 15      630      DCR  D           ;DEC CHR COUNTER
0A02 AF      631      XRA  A           ;CLEAR CARRY
0A03 79      632      MOV   A,C           ;RECOVER 1ST CHR
0A04 17      633      RAL          ;ROTATE LEFT 4 PLACES
0A05 17      634      RAL
0A06 17      635      RAL
0A07 17      636      RAL
0A08 5F      637      MOV   E,A           ;SAVE IT IN E
0A09 C3BC0A  638      JMP  PARIN3        ;GET NEXT CHR
0A0C 79      639 PARIN2:  MOV   A,C           ;2ND CHR IN A
0A0D B3      640      ORA  E           ;COMBINE BOTH CHRS
0A0E C1      641      POP  B           ;RESTORE BC
0A0F C9      642      RET          ;RETURN TO CALLING PROGRAM
0A00 79      643 PARIN1:  MOV   A,C           ;PUT ILLEGAL CHR IN A
0A01 37      644      STC          ;SET CARRY AS ILLEGAL STATUS
0A02 C1      645      POP  B           ;RESTORE BC
0A03 C9      646      RET          ;RETURN TO CALLING PROGRAM
647 ;
648 ;
649 ; JUMP HERE IF BUFFER FULL
650 ;
0A04 CF      651 BUFFUL: DB   0CFH        ;EXIT TO MONITOR
652 ;
653 ;
654 ; COMMAND DISPATCHER
655 ;
656 ;
0A05 210020  657 COMM:  LXI  H,CMDBUF      ;SET POINTER
0A08 C5      658      PUSH B           ;SAVE BC
0A09 71      659      MOV  M,C           ;LOAD COMMAND INTO BUFFER
0A0A 78      660 COMM1:  MOV  A,B           ;CHECK PARAMETER COUNTER
0A0B A7      661      ANA  A           ;IS IT 0?
0A0C CAFB0A  662      JZ   CMDOUT        ;YES, GO ISSUE COMMAND
0A0F CDAD0A  663      CALL PARIN          ;GET PARAMETER
0A02 DAA708  664      JC   ILLEG         ;ILLEGAL CHR RETURNED
0A05 23      665      INX  H           ;INC BUFFER POINTER
0A06 05      666      DCR  B           ;DEC PARAMETER COUNTER
0A07 77      667      MOV  M,A           ;PARAMETER TO BUFFER
0A08 C3EA0A  668      JMP  COMM1         ;GET NEXT PARAMETER
0A0B 210020  669 CMDOUT: LXI  H,CMDBUF      ;REPOINT POINTER
0A0E C1      670      POP  B           ;RESTORE PARAMETER COUNT

```

# APPLICATIONS

```

0AFF DB90      671 COMM2: IN      STAT73      ; READ 8273 STATUS
0B01 07        672      RLC                          ; ROTATE CARRY INTO CARRY
0B02 DAFF0A    673      JC      COMM2      ; WAIT FOR OK
0B05 7E        674      MOV      A, M      ; OK, MOVE COMMAND INTO A
0B06 D390      675      OUT     COMM73     ; OUTPUT COMMAND
0B08 78        676 PAR1: MOV      A, B      ; GET PARAMETER COUNT
0B09 A7        677      ANA      A        ; IS IT 0?
0B0A C8        678      RZ                          ; YES, DONE, RETURN
0B0B 23        679      INX      H        ; INC COMMAND BUFFER POINTER
0B0C 05        680      DCR      B        ; DEC PARAMETER COUNT
0B0D DB90      681 PAR2: IN      STAT73      ; READ STATUS
0B0F E620      682      ANI      CPBF      ; IS CPBF BIT SET?
0B11 C20D0B    683      JNZ     PAR2      ; WAIT TIL ITS 0
0B14 7E        684      MOV      A, M      ; OK, GET PARAMETER FROM BUFFER
0B15 D391      685      OUT     PARM73     ; OUTPUT PARAMETER
0B17 C3080B    686      JMP      PAR1      ; GET NEXT PARAMETER
687 ;
688 ;
689 ; INITIALIZE AND ENABLE RX DMA CHANNEL
690 ;
691 ;
0B1A 3E62      692 RXDMA: MVI      A, DRDMA      ; DISABLE RX DMA CHANNEL
0B1C D3A8      693      OUT     MODE57     ; 8257 MODE PORT
0B1E 010002    694      LXI      B, RXBUF      ; RX BUFFER START ADDRESS
0B21 79        695      MOV      A, C        ; RX BUFFER LSB
0B22 D3A0      696      OUT     CH0ADR      ; CH0 ADR PORT
0B24 78        697      MOV      A, B        ; RX BUFFER MSB
0B25 D3A0      698      OUT     CH0ADR      ; CH0 ADR PORT
0B27 01FF41    699      LXI      B, RXTC      ; RX CH TERMINAL COUNT
0B2A 79        700      MOV      A, C        ; RX TERMINAL COUNT LSB
0B2B D3A1      701      OUT     CH0TC      ; CH0 TC PORT
0B2D 78        702      MOV      A, B        ; RX TERMINAL COUNT MSB
0B2E D3A1      703      OUT     CH0TC      ; CH0 TC PORT
0B30 3E63      704      MVI      A, ENDMA      ; ENABLE DMA WORD
0B32 D3A8      705      OUT     MODE57     ; 8257 MODE PORT
0B34 C9        706      RET                          ; RETURN
707 ;
708 ;
709 ; INITIALIZE AND ENABLE TX DMA CHANNEL
710 ;
711 ;
0B35 3E61      712 TXDMA: MVI      A, DTDMA      ; DISABLE TX DMA CHANNEL
0B37 D3A8      713      OUT     MODE57     ; 8257 MODE PORT
0B39 010008    714      LXI      B, TXBUF      ; TX BUFFER START ADDRESS
0B3C 79        715      MOV      A, C        ; TX BUFFER LSB
0B3D D3A2      716      OUT     CH1ADR      ; CH1 ADR PORT
0B3F 78        717      MOV      A, B        ; TX BUFFER MSB
0B40 D3A2      718      OUT     CH1ADR      ; CH1 ADR PORT
0B42 01FF81    719 TXDMA1: LXI      B, TXTC      ; TX CH TERMINAL COUNT
0B45 79        720      MOV      A, C        ; TX TERMINAL COUNT LSB
0B46 D3A3      721      OUT     CH1TC      ; CH1 TC PORT
0B48 78        722      MOV      A, B        ; TX TERMINAL COUNT MSB
0B49 D3A3      723      OUT     CH1TC      ; CH1 TC PORT
0B4B 3E63      724      MVI      A, ENDMA      ; ENABLE DMA WORD
0B4D D3A8      725      OUT     MODE57     ; 8257 MODE PORT
0B4F C9        726      RET                          ; RETURN
727 ;
728 ;

```

# APPLICATIONS

```

729 ; INERRUPT PROCESSING SECTION
730 ;
0C00 731      ORG      0C00H
732 ;
733 ;
734 ; RECEIVER INTERRUPT - RST 7.5 (LOC 3CH)
735 ;
0C00 E5      736  RXI:  PUSH  H           ; SAVE HL
0C01 F5      737      PUSH  PSH        ; SAVE PSH
0C02 C5      738      PUSH  B           ; SAVE BC
0C03 D5      739      PUSH  D           ; SAVE DE
0C04 3E62    740      MVI   A, DRDMA     ; DISABLE RX DMA
0C06 D3A8    741      OUT   MODE57     ; 8257 MODE PORT
0C08 3E18    742      MVI   A, 18H     ; RESET RST7.5 F/F
0C0A 30      743      SIM
0C0B 1604    744      MVI   D, 04H     ; D IS RESULT COUNTER
0C0D 2A1320  745      LHLD  LDADR     ; GET LOAD POINTER
0C10 E5      746      PUSH  H           ; SAVE IT
0C11 E5      747      PUSH  H           ; SAVE IT AGAIN
0C12 45      748      MOV   B, L       ; SAVE LSB
0C13 2A1320  749      LHLD  CNADR     ; GET CONSOLE POINTER
0C16 04      750  RXI1: INR   B           ; BUMP LOAD POINTER LSB
0C17 78      751      MOV   A, B       ; GET SET TO TEST
0C18 8D      752      CMP   L           ; LOAD=CONSOLE?
0C19 CAE40A  753      JZ    BUFFUL     ; YES, BUFFER FULL
0C1C 15      754      DCR   D           ; DEC COUNTER
0C1D C2160C  755      JNZ   RXI1     ; NOT DONE, TRY AGAIN
0C20 1605    756      MVI   D, 05H     ; RESET COUNTER
0C22 E1      757      POP   H           ; RESTORE LOAD POINTER
0C23 DB90    758  RXI2: IN    STAT73     ; READ STATUS
0C25 E608    759      ANI   RXINT     ; TEST RX INT BIT
0C27 CA390C  760      JZ    RXI3     ; DONE, GO FINISH UP
0C2A DB90    761      IN    STAT73     ; READ STATUS AGAIN
0C2C E602    762      ANI   RXIRA     ; IS RESULT READY?
0C2E CA230C  763      JZ    RXI2     ; NO, TEST AGAIN
0C31 DB93    764      IN    RXIR73    ; YES, READ RESULT
0C33 77      765      MOV   M, A       ; STORE IN BUFFER
0C34 2C      766      INR   L           ; INC BUFFER POINTER
0C35 15      767      DCR   D           ; DEC COUNTER
0C36 C3230C  768      JMP   RXI2     ; GET MORE RESULTS
0C39 7A      769  RXI3: MOV   A, D       ; GET SET TO TEST
0C3A A7      770      ANA   A           ; ALL RESULTS?
0C3B CA450C  771      JZ    RXI4     ; YES, SO FINISH UP
0C3E 3600    772      MVI   M, 00H     ; NO, LOAD 0 TIL DONE
0C40 2C      773      INR   L           ; BUMP POINTER
0C41 15      774      DCR   D           ; DEC COUNTER
0C42 C3390C  775      JMP   RXI3     ; GO AGAIN
0C45 221020  776  RXI4: SHLD  LDADR     ; UPDATE LOAD POINTER
0C48 3A1520  777      LDA   PRMPT     ; GET MODE INDICATOR
0C4B FE2D    778      CPI   ' '         ; NORMAL MODE?
0C4D CA850C  779      JZ    RXI6     ; YES, CLEAN UP BEFORE RETURN
780 ;
781 ;
782 ; POLL MODE SO CHECK CONTROL BYTE
783 ; IF CONTROL IS A POLL, SET UP SPECIAL TX COMMAND BUFFER
784 ; AND RETURN WITH POLL INDICATOR NOT 0
785 ;
0C50 E1      785      POP   H           ; GET PREVIOUS LOAD ADR POINTER
0C51 7E      786      MOV   A, M       ; GET IC BYTE FROM BUFFER

```

# APPLICATIONS

```

0C52 E61E      787      ANI      1EH      ;LOOK AT GOOD FRAME BITS
0C54 C2890C   788      JNZ      RXI5     ;IF NOT 0, INTERRUPT WASN'T FROM A GOOD FRAME
0C57 2C       789      INR      L        ;BYPASS R0 AND R1 IN BUFFER
0C58 2C       790      INR      L
0C59 2C       791      INR      L
0C5A 56       792      MOV      D,M      ;GET ADR BYTE AND SAVE IT IN D
0C5B 2C       793      INR      L
0C5C 7E       794      MOV      A,M      ;GET CNTL BYTE FROM BUFFER
0C5D FE93     795      CPI      SNRMP    ;WAS IT SNRM-P?
0C5F C6C0C    796      JZ       T1       ;YES, GO SET RESPONSE
0C62 FE11     797      CPI      RR0P     ;WAS IT RR(0)-P?
0C64 C2890C   798      JNZ      RXI5     ;YES, GO SET RESPONSE, OTHERWISE RETURN
0C67 1E11     799      MVI     E,RR0F   ;RR(0)-P SO SET RESPONSE TO RR(0)-F
0C69 C36E0C   800      JMP      TXRET    ;GO FINISH LOADING SPECIAL BUFFER
0C6C 1E73     801 T1:    MVI     E,NSAF   ;SNRM-P SO SET RESPONSE TO NSA-F
0C6E 212020   802 TXRET: LXI     H,CMBF1 ;SPECIAL BUFFER ADR
0C71 36C8     806      MVI     M,0C8H   ;LOAD TX FRAME COMMAND
0C73 23       808      INX     H        ;INC POINTER
0C74 3600     809      MVI     M,00H   ;L0=0
0C76 23       810      INX     H        ;INC POINTER
0C77 3600     811      MVI     M,00H   ;L1=0
0C79 23       812      INX     H        ;INC POINTER
0C7A 72       813      MOV     M,D      ;LOAD RCDV ADR BYTE
0C7B 23       814      INX     H        ;INC POINTER
0C7C 73       815      MOV     M,E      ;LOAD RESPONSE CNTL BYTE
0C7D 3E01     816      MVI     A,01H   ;SET POLL INDICATOR NOT 0
0C7F 321620   817      STA     POLIN   ;LOAD POLL INDICATOR
0C82 C3890C   818      JMP      RXI5     ;RETURN
819
0C85 E1       820 RXI6:  POP     H        ;CLEAN UP STACK IF NORMAL MODE
0C86 C3890C   821      JMP      RXI5     ;RETURN
822
0C89 CD1A0B   823 RXI5:  CALL    RXDMA    ;RESET DMA CHANNEL
0C8C D1       824      POP     D        ;RESTORE REGISTERS
0C8D C1       825      POP     B
0C8E F1       826      POP     PSH
0C8F E1       827      POP     H
0C90 FB       828      EI        ;ENABLE INTERRUPTS
0C91 C9       829      RET       ;RETURN
830 ;
831 ;
832 ;MESSAGE TYPER - ASSUMES MESSAGE STARTS AT HL
833 ;
834 ;
0C92 C5       835 TYMSG:  PUSH   B        ;SAVE BC
0C93 7E       836 TYMSG2: MOV    A,M      ;GET ASCII CHR
0C94 23       837      INX     H        ;INC POINTER
0C95 FEFF     838      CPI      0FFH   ;STOP?
0C97 CAA10C   839      JZ       TYMSG1  ;YES, GET SET FOR EXIT
0C9A 4F       840      MOV     C,A      ;SET UP FOR DISPLAY
0C9B CDF805   841      CALL    ECHO    ;DISPLAY CHR
0C9E C3930C   842      JMP      TYMSG2  ;GET NEXT CHR
0CA1 C1       843 TYMSG1: POP     B        ;RESTORE BC
0CA2 C9       844      RET       ;RETURN
845 ;
846 ;
847 ;SIGNON MESSAGE
848 ;

```

# APPLICATIONS

```

0CA3 0D      849 SIGNON: DB      CR, '8273 MONITOR V1.1', CR, 0FFH
0CA4 38323733
0CAB 204D4F4E
0CAC 49544F52
0CB0 20205631
0CB4 2E31
0CB6 0D
0CB7 FF

850 ;
851 ;
852 ;
853 ; RECEIVER INTERRUPT MESSAGES
854 ;
855 ;

0CB8 0D      856 RXMSG: DB      CR, 'RX INT - ', 0FFH
0CB9 52582049
0CBD 4E54202D
0CC1 20
0CC2 FF

857 ;
858 ; TRANSMITTER INTERRUPT MESSAGES
859 ;

0CC3 0D      860 TXMSG: DB      CR, 'TX INT - ', 0FFH
0CC4 54582049
0CC8 4E54202D
0CCC 20
0CCD FF

861 ;
862 ;
863 ; TRANSMITTER INTERRUPT ROUTINE
864 ;

0CCE E5      865 TXI:  PUSH  H      ; SAVE HL
0CCF F5      866      PUSH  PSW     ; SAVE PSW
0CD0 C5      867      PUSH  B      ; SAVE BC
0CD1 D5      868      PUSH  D      ; SAVE DE
0CD2 3E61    869      MVI  A, DTDMA ; DISABLE TX DMA
0CD4 D3A8    870      OUT  MODE57 ; 8257 MODE PORT
0CD6 1604    871      MVI  D, 04H  ; SET COUNTER
0CD8 2A1020  872      LHLD LDADR   ; GET LOAD POINTER
0CDB E5      873      PUSH  H      ; SAVE IT
0CDC 45      874      MOV  B, L     ; SAVE LSB IN B
0CDD 2A1320  875      LHLD CNADR   ; GET CONSOLE POINTER
0CE0 04      876 TXI1: INR  B      ; INC POINTER
0CE1 78      877      MOV  A, B     ; GET SET TO TEST
0CE2 8D      878      CMP  L      ; LOAD=CONSOLE?
0CE3 CAE40A  879      JZ   BUFFUL  ; YES, BUFFER FULL
0CE6 15      880      DCR  D      ; NO, TEST NEXT LOCATION
0CE7 C2E00C  881      JNZ  TXI1   ; TRY AGAIN
0CEA E1      882      POP  H      ; RESTORE LOAD POINTER
0CEB DB92    883      IN  TXIR73  ; READ RESULT
0CED 77      884      MOV  M, A     ; STORE IN BUFFER
0CEE 2C      885      INR  L      ; INR POINTER
0CEF 3600    886      MVI  M, 00H  ; EXTRA RESULT SPOTS 0
0CF1 2C      887      INR  L
0CF2 3600    888      MVI  M, 00H
0CF4 2C      889      INR  L
0CF5 3600    890      MVI  M, 00H
0CF7 2C      891      INR  L

```

# APPLICATIONS

```

0CF8 3600      892      MVI      M, 00H
0CFA 2C        893      INR      L
0CFB 221020    894      SHLD    LDADR      ; UPDATE LOAD POINTER
0CFE CD350B    899      CALL    TXDMA     ; RESET DMA CHANNEL
0D01 D1        900      POP     D         ; RESTORE DE
0D02 C1        901      POP     B         ; RESTORE BC
0D03 F1        902      POP     PSW      ; RESTORE PSW
0D04 E1        903      POP     H         ; RESTORE HL
0D05 FB        904      EI          ; ENABLE INTERRUPTS
0D06 C9        905      RET         ; RETURN
          906 ;
          907 ;
          952 ;
          953 ;
          954      END

```

## PUBLIC SYMBOLS

## EXTERNAL SYMBOLS

## USER SYMBOLS

ADWV A 0922	AFCMD A 09CE	BUFFUL A 0AE4	CHADR A 00A0	CHBTC A 00A1	CHLADR A 00A2	CHLTC A 00A3
CMD51 A 0027	CMDBF1 A 2020	CMDBUF A 2000	CMDOUT A 0AFB	CMDREC A 0057	CMODE A 0931	CNADR A 2013
CNT053 A 009C	CNT153 A 009D	CNT253 A 009E	CNTL51 A 0089	CNTLC A 0003	CNVBN A 05B8	COBR A 000C
COMM A 0AE5	COMM1 A 0AEA	COMM2 A 0AFF	COMM73 A 0090	CPBF A 0020	CR A 000D	CRLF A 05EB
DEM A 0000	DEMODE A 2027	DISPY A 0A39	DISPY1 A 0A4E	DISPY2 A 0A4D	DRDMA A 0062	DTDMA A 0061
ECHO A 05F8	ENDCHK A 0A1B	ENDMA A 0063	GDWV A 00FF	GETCH A 061F	GETCMD A 007D	GRCMD A 09C4
ILLEG A 00A7	LDADR A 2010	LF A 000A	LKBR1 A 2017	LKBR2 A 2018	LOOPIT A 0061	MDCNT0 A 0036
MDCNT2 A 0006	MDE51 A 00CE	MODE53 A 0098	MODE57 A 00A8	MONTOR A 0008	NMOUT A 06C7	NSAF A 0073
PAR1 A 0008	PAR2 A 000D	PARIN A 00AD	PARIN1 A 0AE0	PARIN2 A 0ADC	PARIN3 A 0ABC	PARM73 A 0091
POLIN A 2016	PRMPT A 2015	ROPT A 00A2	R1PT A 00A7	RBCMD A 097B	RDCMD A 0971	RDWV A 00AF
RDV A 0002	RESBUF A 2000	RESL73 A 0091	ROCMD A 095D	RPCMD A 09D8	RR0F A 0011	RR0P A 0011
RSCMD A 0967	RST65 A 20CE	RST75 A 20D4	RXBUF A 0200	RXD51 A 0088	RXDMA A 001A	RXI A 0000
RX11 A 0C16	RX12 A 0C23	RX13 A 0C39	RX14 A 0C45	RX15 A 0C89	RX16 A 0C85	RXIMSG A 00B8
RXINT A 0008	RXIR73 A 0093	RXIRA A 0002	RX51 A 0A69	RX52 A 0A7F	RX53 A 0A8D	RXSORC A 0A62
RXTC A 41FF	SBCMD A 0985	SDWV A 00D7	SIGNON A 0CA3	SLCMD A 098F	SNRMP A 0093	SOCMD A 09A6
SPCMD A 09E2	SRCMD A 098A	SSCMD A 0980	START A 0000	STAT51 A 0089	STAT57 A 00A8	STAT73 A 0090
STKSRT A 20C0	SW A 0943	T1 A 0C6C	TBUFL A 0A24	TBUFL A 0A07	TBUFL1 A 0A0D	TDWV A 090E
TEST73 A 0092	TFCMD A 09EC	TFCMD1 A 09F6	TFRET A 0A36	TLCMD A 0999	TRUE A 0000	TRUE1 A 0000
TXBUF A 0000	TXD51 A 0088	TXDMA A 0035	TXDMA1 A 0B42	TXI A 00CE	TX11 A 0CE0	TXIMSG A 00C3
TXINT A 0004	TXIR73 A 0092	TXIRA A 0001	TXPOL A 094C	TXRET A 006E	TXSORC A 0A47	TXTC A 81FF
TYMSG A 0C92	TYMSG1 A 0CA1	TYMSG2 A 0C93	VALDG A 075E			

ASSEMBLY COMPLETE. NO ERRORS



Asynchronous Communication  
with the 8274 Multiple  
Protocol Serial Controller

Contents

INTRODUCTION	2-318
Communication Functions	
System Interface	
Scope	
SERIAL ASYNCHRONOUS DATA LINKS	2-319
Characters	
Framing	
Timing	
Parity	
Communication Modes	
Break Condition	
MPSC SYSTEM INTERFACE	2-321
Hardware Environment	
Operational Interface	
Reset	
External/Status Latches	
Error Reporting	
Transmitter/Receiver	
Initialization	
Polled Operation	
Interrupt Driven Operation	
Interrupt Configurations	
Interrupt Sources/Priorities	
Interrupt Initialization	
Interrupt Service Routines	
DATA LINK INTERFACE	2-328
Serial Data Interface	
Data Clocking	
Modem Control	
APPENDIX A	2-329
Command/Status Details	
Per Async	
Communication	
APPENDIX B	2-336
MPSC Polled Transmit/Receive	
Character Routines	
APPENDIX C	2-340
Interrupt Driven	
Transmit/Receive	
Software	
APPENDIX D	2-342
Application Example	
Using SDK-86	

# APPLICATIONS

## ASYNCHRONOUS COMMUNICATION

### WITH THE 8274

#### MULTI-PROTOCOL SERIAL CONTROLLER

##### 1. Introduction

The 8274 Multi-protocol serial controller (MPSC) is a sophisticated dual-channel communications controller that interfaces microprocessor systems to high-speed serial data links (at speeds to 880K bits per second) using synchronous or asynchronous protocols. The 8274 interfaces easily to most common microprocessors (e.g., 8048, 8051, 8085, 8086, and 8088), to DMA controllers such as the 8237 and 8257, and to the 8089 I/O processor. Both MPSC communication channels are completely independent and can operate in a full-duplex communication mode (simultaneous data transmission and reception).

##### Communication Functions

The 8274 performs many communications oriented functions, including:

Converting data bytes from a microprocessor system into a serial bit stream for transmission over the data link to a receiving system.

Receiving serial bit streams and reconverting the data into parallel data bytes that can easily be processed by the microprocessor system.

Performing error checking during data transfers. Error checking functions include computing/transmitting error codes (such as parity bits or CRC bytes) and using these codes to check the validity of received data.

Operating independently of the system processor in a manner designed to reduce the system overhead involved in data transfers.

##### System Interface

The MPSC system interface is extremely flexible, supporting the following data transfer modes:

1. **Polled Mode.** The system processor periodically reads (polls) an 8274 status register to determine when a

character has been received, when a character is needed for transmission, and when transmission errors are detected.

2. **Interrupt Mode.** The MPSC interrupts the system processor when a character has been received, when a character is needed for transmission, and when transmission errors are detected.
3. **DMA Mode.** The MPSC automatically requests data transfers from system memory for both transmit and receive functions by means of two DMA request signals per serial channel. These DMA request signals may be directly interfaced to an 8237 or 8257 DMA controller or to an 8089 I/O processor.
4. **WAIT Mode.** The MPSC ready signal is used to synchronize processor data transfers by forcing the processor to enter wait states until the 8274 is ready for another data byte. This feature enables the 8274 to interface directly to an 8086 or 8088 processor by means of string I/O instructions for very high speed data links.

##### Scope

This application note describes the use of the 8274 in asynchronous communication modes. Asynchronous communication is typically used to transfer data to/from video display terminals, modems, printers, and other low-to-medium speed peripheral devices. Use of the 8274 in both interrupt driven and polled system environments is described. Use of the DMA and WAIT modes are not described since these modes are employed mainly in synchronous communication systems where extremely high data rates are common. Programming examples are written in PL/M-86 (Appendix B and Appendix C). PL/M-86 is executed by the iAPX-86 and iAPX-88 processor families. In addition, PL/M-86 is very similar to PL/M-80 (executed by the MCS-80 and MCS-85 processor families). In addition, Appendix D describes a simple application example using an SDK-86 in an iAPX-86/88 environment.

## 2. Serial Asynchronous Data Links

A serial asynchronous interface is a method of data transmission in which the receiving and transmitting systems need not be synchronized. Instead of transmitting clocking information with the data, locally generated clocks (16, 32 or 64 times as fast as the data transmission rate) are used by the transmitting and receiving systems. When a character of information is sent by the transmitting system, the character data is framed (preceded and followed) by special START and STOP bits. This framing information permits the receiving system to temporarily synchronize with the data transmission. (Refer to Figure 1 during the following discussion of asynchronous data transmission.)

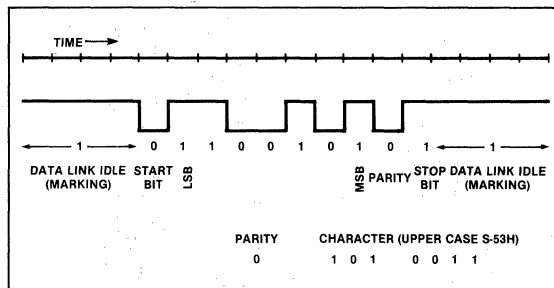


Figure 1. Transmission of a 7-bit ASCII Character with Even Parity

Normally the data link is in an idle or marking state, continuously transmitting a "mark" (binary 1). When a character is to be sent, the character data bits are immediately preceded by a "space" (binary 0 START bit). The mark-to-space transition informs the receiving system that a character of information will immediately follow the start bit. Figure 1 illustrates the transmission of a 7-bit ASCII character (upper case S) with even parity. Note that the character is transmitted immediately following the start bit. Data bits within the character are transmitted from least-significant to most-significant. The parity bit is transmitted immediately following the character data bits and the STOP framing bit (binary 1) signifies the end of the character.

Asynchronous interfaces are often used with human interface devices such as CRT/key-board units where the time between data transmissions is extremely variable.

## Characters

In asynchronous mode characters may vary in length from five to eight bits. The character length depends on the coding method used. For example, five-bit characters are used when transmitting Baudot Code, seven-bit characters are required for ASCII data, and eight-bit characters are needed for EBCDIC and binary data. To transmit messages composed of multiple characters, each character is framed and transmitted separately (Figure 2).

This framing method ensures that the receiving system can easily synchronize with the start and stop bits of each character, preventing receiver synchronization errors. In addition, this synchronization method makes both transmitting and receiving systems insensitive to possible time delays between character transmissions.

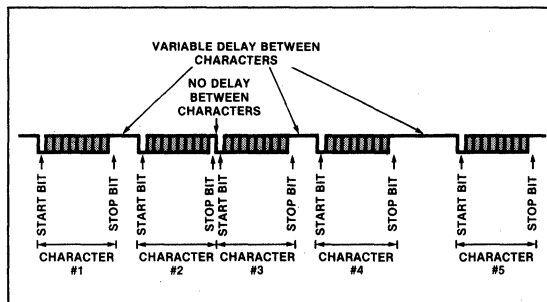


Figure 2. Multiple Character Transmission

## Framing

Character framing is accomplished by the START and STOP bits described previously. When the START bit transition (mark to space) is detected, the receiving system assumes that a character of data will follow. In order to test this assumption (and isolate noise pulses on the data link), the receiving system waits one-half bit time and samples the data link again. If the link has returned to the marking state, noise is assumed, and the receiver waits for another START bit transition.

When a valid START bit is detected, the receiver samples the data link for each bit of the following character. Character data bits and the parity bit (if required) are sampled at their nominal centers until all required characters are received. Immediately following the data bits, the receiver

# APPLICATIONS

samples the data link for the STOP bit, indicating the end of the character. Most systems permit specification of 1, 1 1/2, or 2 stop bits.

## Timing

The transmitter and receiver in an asynchronous data link arrangement are clocked independently. Normally, each clock is generated locally and the clocks are not synchronized. In fact, each clock may be a slightly different frequency. (In practice, the frequency difference should not exceed a few percent. If the transmitter and receiver clock rates vary substantially, errors will occur because data bits may be incorrectly identified as START or STOP framing bits.) These clocks are designed to operate at 16, 32, or 64 times the communications data rate. These clock speeds allow the receiving device to correctly sample the incoming bit stream.

Serial interface data rates are measured in bits/second. The term baud is used to specify the number of times per second that the transmitted signal level can change states. In general, the baud is not equal to the bit rate. Only when the transmitted signal has two states (electrical levels) is the baud rate equal to the bit rate. Most point-to-point serial data links use RS-232-C, RS-422, or RS-423 electrical interfaces. These specifications call for two electrical signal levels (the baud is equal to the bit rate). Modem interfaces, however, may often have differing bit and baud rates.

While there are generally no limitations on the data transmission rates used in an asynchronous data link, a limited set of rates has been standardized to promote equipment interconnection. These rates vary from 75 bits per second to 38,400 bits per second. Table 1 illustrates typical

asynchronous data rates and the associated clock frequencies required for the transmitter and receiver circuits.

## Parity

In order to detect transmission errors, a parity bit may be added to the character data as it is transferred over the data link. The parity bit is set or cleared to make the total number of "one" bits in the character even (even parity) or odd (odd parity). For example, the letter "A" is represented by the seven-bit ASCII code 1000001 (41H). The transmitted data code (with parity) for this character contains eight bits; 01000001 (41H) for even parity and 11000001 (0C1H) for odd parity. Note that a single bit error changes the parity of the received character and is therefore easily detected. The 8274 supports both odd and even parity checking as well as a parity disable mode to support binary data transfers.

## Communication Modes

Serial data transmission between two devices can occur in one of three modes. In the simplex transmission mode, a data link can transmit data in one direction only. In the half-duplex mode, the data link can transmit data in both directions, but not simultaneously. In the full-duplex mode (the most common), the data link can transmit data in both directions simultaneously. The 8274 directly supports the full-duplex mode and will interface to simplex and half-duplex communication data links with appropriate software controls.

Table 1. Communication Data Rates and Associated Transmitter/Receiver Clock Rates

Data Rate (bits/second)	Clock Rate (kHz)		
	X16	X32	X64
75	1.2	2.4	4.8
150	2.4	4.8	9.6
300	4.8	9.6	19.2
600	9.6	19.2	38.4
1200	19.2	38.4	76.8
2400	38.4	76.8	153.6
4800	76.8	153.6	307.2
9600	153.6	307.2	614.2
19200	307.2	614.4	—
38400	614.4	—	—

## Break Condition

Asynchronous data links often include a special sequence known as a break condition. A break condition is initiated when the transmitting device forces the data link to a spacing state (binary 0) for an extended length of time (typically 150 milliseconds). Many terminals contain keys to initiate a break sequence. Under software control, the 8274 can initiate a break sequence when transmitting data and detect a break sequence when receiving data.

# APPLICATIONS

## 3. MPSC System Interface

### Hardware Environment

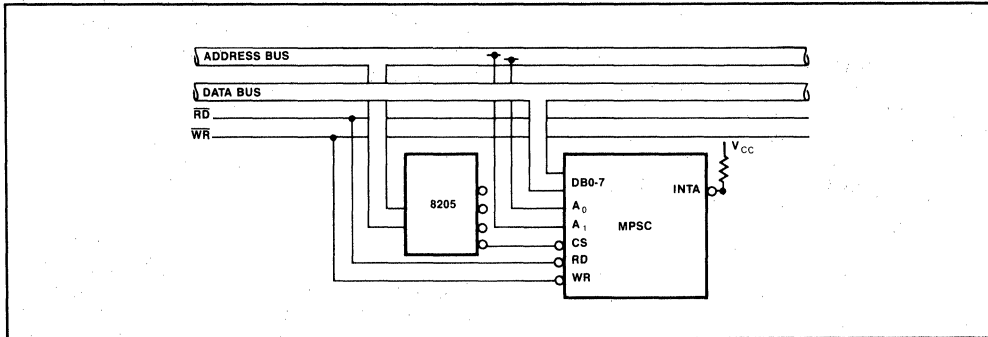
The 8274 MPSC interfaces to the system processor over an 8-bit data bus. Each serial I/O channel responds to two I/O or memory addresses as shown in Table 2. In addition, the MPSC supports vectored and daisy-chained interrupts.

Table 2: 8274 Addressing. The 8274 may be configured for memory mapped or I/O mapped operation.

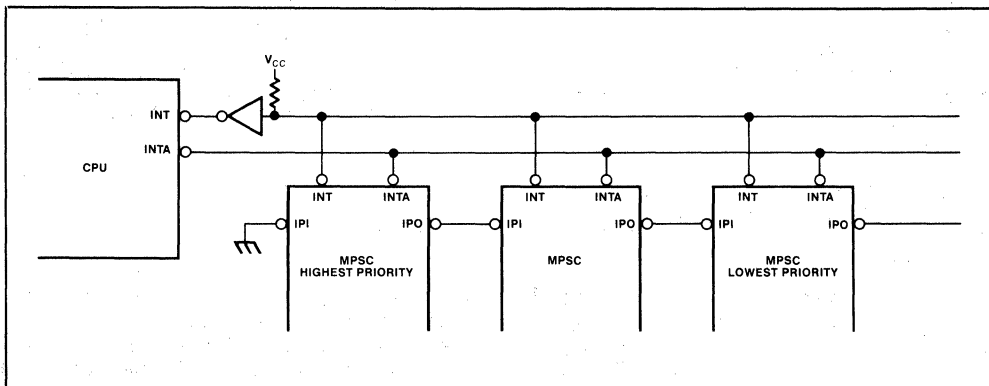
CS	A <sub>1</sub>	A <sub>0</sub>	Read Operation	Write Operation
0	0	0	Ch. A Data Read	Ch. A Data Write
0	1	0	Ch. A Status Read	Ch. A Command/Parameter
0	0	1	Ch. B Data Read	Ch. B Data Write
0	1	1	Ch. B Status Read	Ch. B Command/Parameter
1	X	X	High Impedance	High Impedance

The 8274/processor hardware interface can be configured in a flexible manner, depending on the operating mode selected -- polled, interrupt driven, DMA, or WAIT. Figure 3 illustrates typical MPSC configurations for use with an 8088 microprocessor in the polled and interrupt driven modes.

All serial-to-parallel conversion, parallel-to-serial conversion, and parity checking required during asynchronous serial I/O operation is automatically performed by the MPSC.



a) Polled Configuration



b) Daisy-chained Interrupt Configuration

Figure 3: 8274 Hardware Interface for Polled and Interrupt Driven Environments

# APPLICATIONS

## Operational Interface

Command, parameter, and status information is stored in 22 registers within the MPSC (8 writable registers and 3 readable registers for each channel). These registers are all accessed by means of the command/status ports for each channel. An internal pointer register selects which of

the command or status registers will be written or read during a command/status access of an MPSC channel. Figure 4 diagrams the command/status register architecture for each serial channel. In the following discussion, the writable registers will be referred to as WRO through WR7 and the readable registers will be referred to as RRO through RR2.

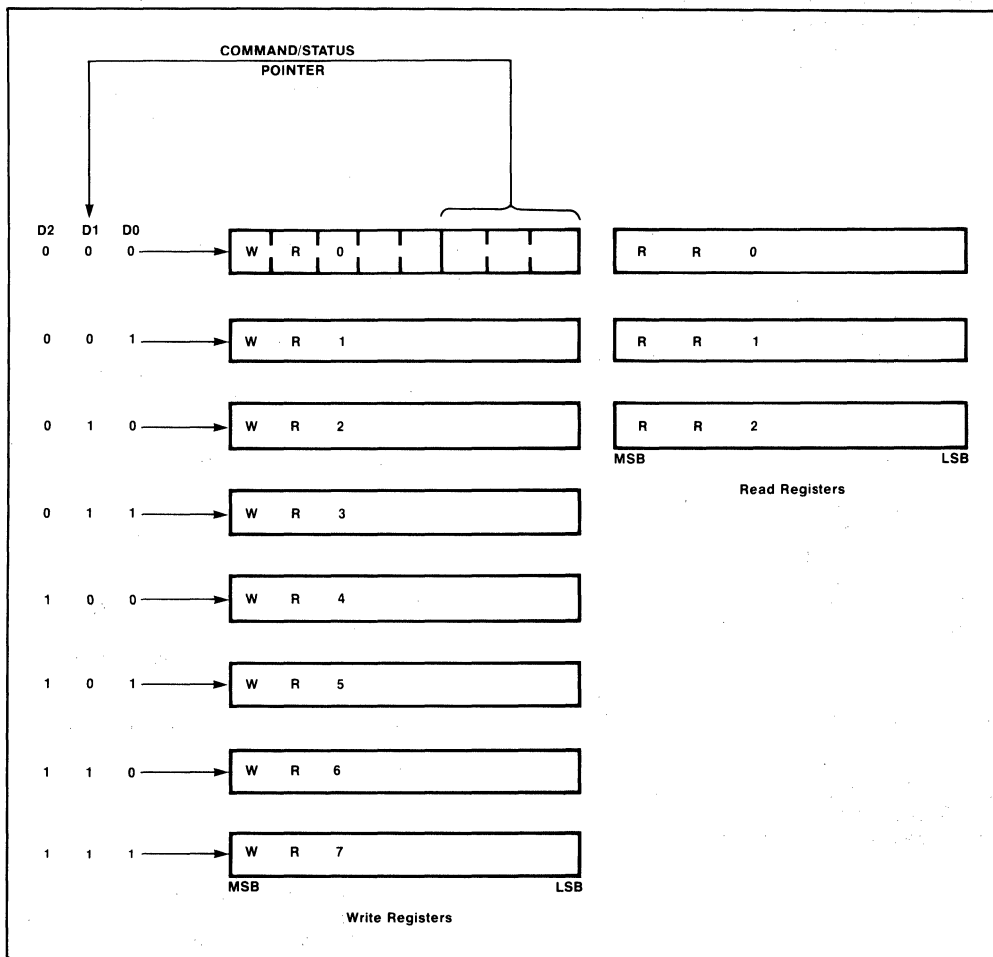


Figure 4: Command/Status Register Architecture (each serial channel)

The least significant three bits of WRO are automatically loaded into the pointer register every time WRO is written. After reset, WRO is set to zero so that the first write to a command register causes the data to be loaded into WRO (thereby setting the pointer register). After WRO is written, the following read or write accesses the

register selected by the pointer. The pointer is reset after the read or write operation is completed. In this manner, reading or writing an arbitrary MPSC channel register requires two I/O accesses. The first access is always a write command. This write command is used to set the pointer register. The second access is

# APPLICATIONS

either a read or a write command; the pointer register (previously set) will ensure that the correct internal register is read or written. After this second access, the pointer register is automatically reset. Note that writing WR0 and reading RRO does not require presetting of the pointer register.

During initialization and normal MPSC operation, various registers are read and/or written by the system processor. These actions are discussed in detail in the following paragraphs. Note that WR6 and WR7 are not used in the asynchronous communication modes.

## Reset

When the 8274 RESET line is activated, both MPSC channels enter the idle state. The serial output lines are forced to the marking state (high) and the modem interface signals (RTS, DTR) are forced high. In addition, the pointer register is set to zero.

## External/Status Latches

The MPSC continuously monitors the state of four external/status conditions:

1. CTS - clear to send input pin.
2. CD - carrier detect input pin.
3. SYNDET - sync detect input pin. This pin may be used as a general purpose input in the asynchronous communication mode.
4. BREAK - a break condition (series of space bits on the receiver input pin).

A change of state in any of these monitored conditions will cause the associated status bit in RRO (Appendix A) to be latched (and optionally cause an interrupt).

## Error Reporting

Three error conditions may be encountered during data reception in the asynchronous mode:

1. Parity. If parity bits are computed and transmitted with each character and the MPSC is set to check parity (bit 0 in WR4 is set), a parity error will occur whenever the number of "1" bits within the character (including the parity bit) does not match the odd/even setting of the parity check flag (bit 1 in WR4).

2. Framing. A framing error will occur if a stop bit is not detected immediately following the parity bit (if parity checking is enabled) or immediately following the most-significant data bit (if parity checking is not enabled).
3. Overrun. If an input character has been assembled but the receiver buffers are full (because the previously received characters have not been read by the system processor), an overrun error will occur. When an overrun error occurs, the input character that has just been received will overwrite the immediately preceding character.

## Transmitter/Receiver Initialization

In order to operate in the asynchronous mode, each MPSC channel must be initialized with the following information:

1. Clock Rate. This parameter is specified by bits 6 and 7 of WR4. The clock rate may be set to 16, 32, or 64 times the data link bit rate. (See Appendix A for WR4 details.)
2. Number of Stop Bits. This parameter is specified by bits 2 and 3 of WR4. The number of stop bits may be set to 1, 1 1/2, or 2. (See Appendix A for WR4 details.)
3. Parity Selection. Parity may be set for odd, even, or no parity by bits 0 and 1 of WR4. (See Appendix A for WR4 details.)
4. Receiver Character Length. This parameter sets the length of received characters to 5, 6, 7, or 8 bits. This parameter is specified by bits 6 and 7 of WR3. (See Appendix A for WR3 details.)
5. Receiver Enable. The serial channel receiver operation may be enabled or disabled by setting or clearing bit 0 of WR3. (See Appendix A for WR3 details.)
6. Transmitter Character Length. This parameter sets the length of transmitted characters to 5, 6, 7, or 8 bits. This parameter is specified by bits 5 and 6 of WR5. (See Appendix A for WR5 details.) Characters of less than 5 bits in length may be transmitted by setting the transmitted length to five bits (set bits 5 and 6 of WR5 to 1).

# APPLICATIONS

The MPSC then determines the actual number of bits to be transmitted from the character data byte. The bits to be transmitted must be right justified in the data byte, the next three bits must be set to 0 and all remaining bits must be set to 1. The following table illustrates the data formats for transmission of 1 to 5 bits of data:

D7	D6	D5	D4	D3	D2	D1	D0	Number of Bits Transmitted (Character Length)
1	1	1	1	0	0	0	c	1
1	1	1	0	0	0	c	c	2
1	1	0	0	0	c	c	c	3
1	0	0	0	c	c	c	c	4
0	0	0	c	c	c	c	c	5

7. **Transmitter Enable.** The serial channel transmitter operation may be enabled or disabled by setting or clearing bit 3 of WR5. (See Appendix A for WR5 details.)

For data transmission via a modem or RS-232-C interface, the following information must also be specified:

1. **Request to Send/Data Terminal Ready.** Must be set to indicate status of data terminal equipment. Request to send is controlled by bit 1 of WR5 and data terminal ready is controlled by bit 7. (See Appendix A for WR5 details.)
2. **Auto Enable.** May be set to allow the MPSC to automatically enable the channel transmitter when the clear to send signal is active and to automatically enable the receiver when the carrier detect signal is active. Auto Enable is controlled by bit 5 of WR3. (See Appendix A for WR3 details.)

During initialization, it is desirable to guarantee that the external/status latches reflect the latest interface information. Since up to two state changes are internally stored by the MPSC, at least two Reset External/Status Interrupt commands must be issued. This procedure is most easily accomplished by simply issuing this reset command whenever the pointer register is set during initialization.

An MPSC initialization procedure (MPSC\$RX\$INIT) for asynchronous communication is listed in Appendix B. Figure 5 illustrates typical MPSC initialization parameters for use with this procedure.

call MPSC\$RX\$INIT(41, 1,1,0,1, 3,1,1, 3,1,1,0,1);	
initializes the 8274 at address 41 as follows:	
X16 clock rate	Enable transmitter and receiver
1 stop bit	Auto enable set
Odd parity	DTR and RTS set
8-bit characters (Tx and Rx)	Break transmission disabled

Figure 5. Sample 8274 initialization procedure for polled operation.

## Polled Operation

In the polled mode, the processor must monitor the MPSC status by testing the appropriate bits in the read register. Data available, status, and error conditions are represented in RRO and RRI for channels A and B. An example of MPSC polled transmitter/receiver routines are given in Appendix B. The following routines are detailed:

1. **MPSC\$POLL\$RCV\$CHARACTER** - This procedure receives a character from the serial data link. The routine waits until the character available flag in RRO has been set. When this flag indicates that a character is available, RRI is checked for errors (overrun, parity, or framing). If an error is detected, the character in the MPSC receive buffer must be read and discarded and the error routine (RECEIVE\$ERROR) is called. If no receive errors have been detected, the character is input from the 8274 data port and returned to the calling program.

MPSC\$POLL\$RCV\$CHARACTER requires three parameters - the address of the 8274 channel data port (data\$port), the address of the 8274 channel command port (cmd\$port), and the address of a byte variable in which to store the received character (character\$ptr).



# APPLICATIONS

2. **MPSC\$POLL\$TRAN\$CHARACTER** - This procedure transmits a character to the serial data link. The routine waits until the transmitter buffer empty flag has been set in RRO before writing the character to the 8274.

**MPSC\$POLL\$TRAN\$CHARACTER** requires three parameters - the address of the 8274 channel data port (**data\$port**), the address of the 8274 channel command port (**cmd\$port**), and the character of data that is to be transmitted (**character**).

3. **RECEIVE\$ERROR** - This procedure processes receiver errors. First, an Error Reset command is written to the affected channel. All additional error processing is dependent on the specific application. For example, the receiving device may immediately request retransmission of the character or wait until a message has been completed. **RECEIVE\$ERROR** requires two parameters - the address of the affected 8274 command port (**cmd\$port**) and the error status (**status**) from 8274 register RR1.

## Interrupt Driven Operation

In an interrupt driven environment, all receiver operations are reported to the system processor by means of interrupts. Once a character has been received and assembled, the MPSC interrupts the system processor. The system processor must then read the character from the MPSC data buffer and clear the current interrupt. During transmission, the system processor starts serial I/O by writing the first character of a message to the MPSC. The MPSC interrupts the system processor whenever the next character is required (i.e., when the transmitter buffer is empty) and the processor responds by writing the next character of the message to the MPSC data port for the appropriate channel.

By using interrupt driven I/O, the MPSC proceeds independently of the system processor, signalling the processor only when characters are required for transmission, when characters are received from the data link, or when errors occur. In this manner, the system processor may continue execution of other tasks while serial I/O is performed concurrently.

## Interrupt Configurations

The 8274 is designed to interface to 8085- and 8086-type processors in much the same manner as the 8259A is designed. When

operating in the 8085 mode, the 8274 causes a "call" to a prespecified interrupt service routine location. In the 8086 mode, the 8274 presents the processor with a one-byte interrupt type number. This interrupt type number is used to "vector" through the 8086 interrupt service table. In either case, the interrupt service address or interrupt type number is specified during MPSC initialization.

To shorten interrupt latency, the 8274 can be programmed to modify the prespecified interrupt vector so that no software overhead is required to determine the cause of an interrupt. When this "status affects vector" mode is enabled, the following eight interrupts are differentiated automatically by the 8274 hardware:

1. Channel B Transmitter Buffer Empty
2. Channel B External/Status Transition
3. Channel B Character Available
4. Channel B Receive Error
5. Channel A Transmitter Buffer Empty
6. Channel A External/Status Transition
7. Channel A Character Available
8. Channel A Receive Error

## Interrupt Sources/Priorities

The 8274 has three interrupt sources for each channel:

1. Receiver (RxA, RxB). An interrupt is initiated when a character is available in the receiver buffer or when a receiver error (parity, framing, or overrun) is detected.
2. Transmitter (TxA, TxB). An interrupt is initiated when the transmitter buffer is empty and the 8274 is ready to accept another character for transmission.
3. External/Status (ExTA, ExTB). An interrupt is initiated when one of the external/status conditions (CD, CTS, SYNDT, BREAK) changes state.

The 8274 supports two interrupt priority orderings (selectable during MPSC initialization) as detailed in Appendix A, WR2, CH-A.

# APPLICATIONS

## Interrupt Initialization

In addition to the initialization parameters required for polled operation, the following parameters must be supplied to the 8274 to specify interrupt operation:

1. **Transmit Interrupt Enable.** Transmitter buffer empty interrupts are separately enabled by bit 1 of WR1. (See Appendix A for WR1 details.)
2. **Receive Interrupt Enable.** Receiver interrupts are separately enabled in one of three modes: a) interrupt on first received character only and on receive errors (used for message oriented transmission systems), b) interrupt on all received characters and on receive errors, but do not interrupt on parity errors, and c) interrupt on all received characters and on receive errors (including parity errors). The ability to separately disable parity interrupts can be extremely useful when transmitting messages. Since the parity error bit in RR1 is latched, it will not be reset until an error reset operation is performed. Therefore, the parity error bit will be set if any parity errors were detected in a multi-character message. If this mode is used, the serial I/O software must poll the parity error bit at the completion of a message and issue an error reset if appropriate. The receiver interrupt mode is controlled by bits 3 and 4 of WR1. (See Appendix A for WR1 details.)
3. **External/Status Interrupts.** External/Status interrupts can be separately enabled by bit 0 of WR1. (See Appendix A for WR1 details.)
4. **Interrupt Vector.** An eight-bit interrupt service routine location (8085) or interrupt type (8086) is specified through WR2 of channel B. (See Appendix A for WR2 details). Table 3 lists interrupt vector addresses generated by the 8274 in the "status affects vector" mode.
5. **Status Affects Vector Mode.** The 8274 will automatically modify the interrupt vector if bit 3 of WR1 is set. (See Appendix A for WR1 details.)
6. **System Configuration.** Specifies the 8274 data transfer mode. Three configuration modes are available: a) interrupt driven operation for both channels, b) DMA operation for both channels, and c) DMA operation for channel A, interrupt driven operation for channel B. The system configuration is specified by means of bits 0 and 1 of WR2 (channel A). (See Appendix A for WR2 details.)
7. **Interrupt Priorities.** The 8274 permits software specification of receive/transmit priorities by means of bit 2 of WR2 (channel A). (See Appendix A for WR2 details.)
8. **Interrupt Mode.** Specifies whether the MPSC is to operate in a non-vectorized mode (for use with an external interrupt controller), in an 8086 vectored mode, or in an 8085 vectored mode. This parameter is specified through bits 3 and 4 of WR2 (channel A). (See Appendix A for WR2 details.)

**Table 3. MPSC Generated Interrupt Vectors in "Status Affects Vector" Mode**

V7 V6 V5 V4 V3 V2 V1 V0	V7 V6 V5 V4 V3 V2 V1 V0	Original Vector (specified during initialization)
8086 Interrupt Type	8085 Interrupt Location	Interrupt Condition
V7 V6 V5 V4 V3 0 0 0	V7 V6 V5 0 0 0 V1 V0	Channel B Transmitter Buffer Empty
V7 V6 V5 V4 V3 0 0 1	V7 V6 V5 0 0 1 V1 V0	Channel B External/Status Change
V7 V6 V5 V4 V3 0 1 0	V7 V6 V5 0 1 0 V1 V0	Channel B Receiver Character Available
V7 V6 V5 V4 V3 0 1 1	V7 V6 V5 0 1 1 V1 V0	Channel B Receive Error
V7 V6 V5 V4 V3 1 0 0	V7 V6 V5 1 0 0 V1 V0	Channel A Transmitter Buffer Empty
V7 V6 V5 V4 V3 1 0 1	V7 V6 V5 1 0 1 V1 V0	Channel A External/Status Change
V7 V6 V5 V4 V3 1 1 0	V7 V6 V5 1 1 0 V1 V0	Channel A Receiver Character Available
V7 V6 V5 V4 V3 1 1 1	V7 V6 V5 1 1 1 V1 V0	Channel A Receive Error

An MPSC interrupt initialization procedure (MPSC\$INT\$INIT) is listed in Appendix C.

# APPLICATIONS

## Interrupt Service Routines

Appendix C lists four interrupt service procedures, a buffer transmission procedure, and a buffer reception procedure that illustrate the use of the 8274 in interrupt driven environments. Use of these procedures assumes that the 8086/8088 interrupt vector is set to 20H and that channel B is used with the "status affects vector" mode enabled.

1. **TRANSMIT\$BUFFER** - This procedure begins serial transmission of a data buffer. Two parameters are required - a pointer to the buffer (**buf\$ptr**) and the length of the buffer (**buf\$length**). The procedure first sets the global buffer pointer, buffer length, and initial index for the transmitter interrupt service routine and initiates transmission by writing the first character of the buffer to the 8274. The procedure then enters a wait loop until the I/O completion status is set by the transmit interrupt service routine (**MPSC\$TRANSMIT\$CHARACTER\$INT**).
2. **RECEIVE\$BUFFER** - This procedure inputs a line (terminated by a line feed) from a serial I/O port. Two parameters are required - a pointer to the input buffer (**buf\$ptr**) and a pointer to the buffer length variable (**buf\$length\$ptr**). The buffer length will be set by this procedure when the complete line has been input. The procedure first sets the global buffer pointer and initial index for the receiver interrupt service routine. **RECEIVE\$BUFFER** then enters a wait loop until the I/O completion status is set by the receive interrupt routine (**MPSC\$RECEIVE\$CHARACTER\$INT**).
3. **MPSC\$RECEIVE\$CHARACTER\$INT** - This procedure is executed when the MPSC Tx buffer empty interrupt is acknowledged. If the current transmit buffer index is less than the buffer length, the next character in the buffer is written to the MPSC data port and the buffer pointer is updated. Otherwise, the transmission complete status is posted.
4. **MPSC\$RECEIVE\$CHARACTER\$INT** - This procedure is executed when a character has been assembled by the MPSC and the MPSC has issued a character available interrupt. If no input buffer has been set up by **RECEIVE\$BUFFER**, the character is ignored. If a buffer has been set up, but it is full, a receive overrun error is posted. Otherwise, the received character is read from the MPSC data port and the buffer index is updated. Finally, if the received character is a line feed, the reception complete status is posted.
5. **RECEIVE\$ERROR\$INT** - This procedure is executed when a receive error is detected. First, the error conditions are read from **RR1** and the character currently in the MPSC receive buffer is read and discarded. Next, an Error Reset command is written to the affected channel. All additional error procession is application dependent.
6. **EXTERNAL\$STATUS\$CHANGE\$INT** - This procedure is executed when an external status condition change is detected. The status conditions are read from **RR0** and a Reset External/Status Interrupt command is issued. Further error processing is application dependent.

# APPLICATIONS

## 4. Data Link Interface

### Serial Data Interface

Each serial I/O channel within the 8274 MPSC interfaces to two data link lines -- one line for transmitting data and one for receiving data. During transmission, characters are converted from parallel data format (as supplied by the system processor or DMA device) into a serial bit stream (with START and STOP bits) and clocked out on the TxD pin. During reception, a serial bit stream is input on the RxD pin, framing bits are stripped out of the data stream, and the resulting character is converted to parallel data format and passed to the system processor or DMA device.

### Data Clocking

As discussed previously, the frequency of data transmission/reception on the data link is controlled by the MPSC clock in conjunction with the programmed clock divider (in register WR4). The 8274 is designed to permit all four serial interface lines (TxD and RxD for each channel) to operate at different data rates. Four clock input pins (TxC and RxC for each channel) are available for this function. Note that the clock rate divider specified in WR4 is used for both RxC and TxC on the appropriate channel; clock rate dividers for each channel are independent.

### Modem Control

The following four modem interface signals may be connected to the 8274:

1. Data Terminal Ready (DTR). This interface signal (output by the 8274) is software controlled through bit 7 of WR5. When active, DTR indicates that the data terminal/computer equipment is active and ready to interact with the data communications channel. In addition, this signal prepares the modem for connection to the communication

channel and maintains connections previously established (e.g., manual call origination).

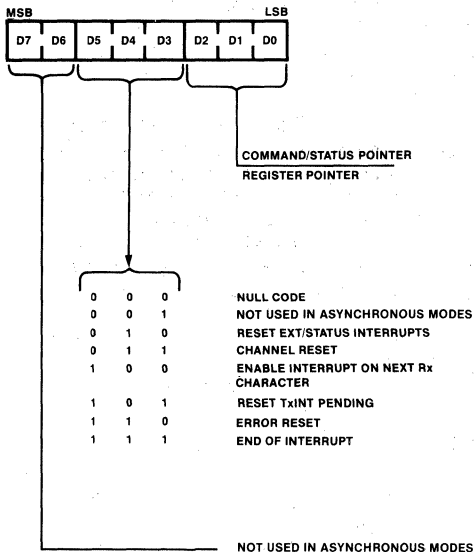
2. Request To Send (RTS). This interface signal (output by the 8274) is software controlled through bit 1 of WR5. When active, RTS indicates that the data terminal/computer equipment is ready to transmit data.
3. Clear To Send (CTS). This interface signal (input to the 8274) is supplied by the modem in response to an active RTS signal. CTS indicates that the data terminal/computer equipment is permitted to transmit data. The state of CTS is available to the programmer as bit 5 of RRO. In addition, if the auto enable control is set (bit 5 of WR3), the 8274 will not transmit data bytes until RTS has been activated. If CTS becomes inactive during transmission of a character, the current character transmission is completed before the transmitter is disabled.
4. Carrier Detect (CD). This interface signal (input to the 8274) is supplied by the modem to indicate that a data carrier signal has been detected and that a valid data signal is present on the RxD line. The state of CD is available to the programmer as bit 3 of RRO. In addition, if the auto enable control is set (bit 5 of WR3), the 8274 will not enable the serial receiver until CD has been activated. If the CD signal becomes inactive during reception of a character, the receiver is disabled, and the partially received character is lost.

In addition to the above modem interface signals, the 8274 SYNDET input pin for channel A may be used as a general purpose input in the asynchronous communication mode. The status of this signal is available to the programmer as bit 4 of status register RRO.

# APPLICATIONS

## Appendix A: Command/Status Details for Asynchronous Communication

### Write Register 0 (WRO):



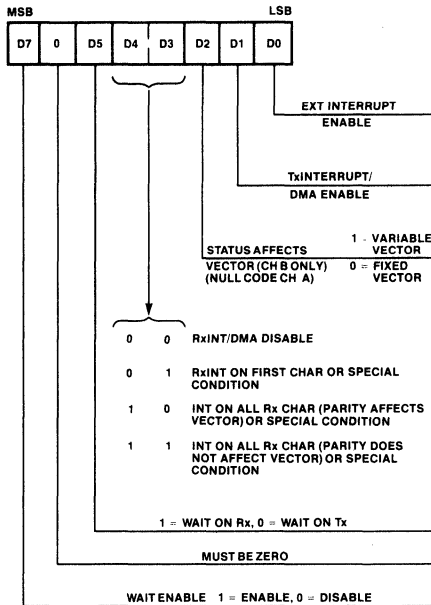
**D2, D1, D0** Command/Status Register Pointer bits determine which write-register the next byte is to be written into, or which read-register the next byte is to be read from. After reset, the first byte written into either channel goes into WRO. Following a read or write to any register (except WRO) the pointer will point to WRO.

**D5, D4, D3** Command bits determine which of the basic seven commands are to be performed.

- Command 0** Null--has no effect.
- Command 1** Not used in asynchronous modes.
- Command 2** Reset External/Status Interrupts--resets the latched status bits of RRO and re-enables them, allowing interrupts to occur again.
- Command 3** Channel Reset--resets the Latched Status bits of RRO, the interrupt prioritization logic and all control registers for the channel. Four extra system clock cycles should be allowed for MPSC reset time before any additional commands or controls are written into the channel.
- Command 4** Enable Interrupt on Next Receive Character--if the Interrupt on First Receive Character mode is selected, this command reactivates that mode after each complete message is received to prepare the MPSC for the next message.
- Command 5** Reset Transmitter Interrupt Pending--if The Transmit Interrupt mode is selected, the MPSC automatically interrupts data when the transmit buffer becomes empty. When there are no more characters to be sent, issuing this command prevents further transmitter interrupts until the next character has been completely sent.
- Command 6** Error Reset--error latches, Parity and Overrun errors in RRI are reset.
- Command 7** End of Interrupt--resets the interrupt-in-service latch of the highest-priority internal device under service.

# APPLICATIONS

## Write Register 1 (WR1):



	D4,D3	Receive Interrupt Mode
	0 0	Receive Interrupts/DMA Disabled
	0 1	Receive Interrupt on First Character Only or Special Condition
	1 0	Interrupt on All Receive Characters of Special Condition (Parity Error is a Special Receive Condition)
	1 1	Interrupt on All Receive Characters or Special Condition (Parity Error is not a Special Receive Condition).
	D5	Wait on Receive/Transmit--when the following conditions are met the RDY pin is activated, otherwise it is held in the High-Z state. (Conditions: Interrupt Enabled Mode, Wait Enabled, CS=0, A0=0/1, and A1=0). The RDY pin is pulled low when the transmitter buffer is full or the receiver buffer is empty and it is driven High when the transmitter buffer is empty or the receiver buffer is full. The RDY <sub>A</sub> and RDY <sub>B</sub> may be wired OR connected since only one signal is active at any one time while the other is in the High Z state.
	D6	Must be Zero.
	D7	Wait Enable--enables the wait function.

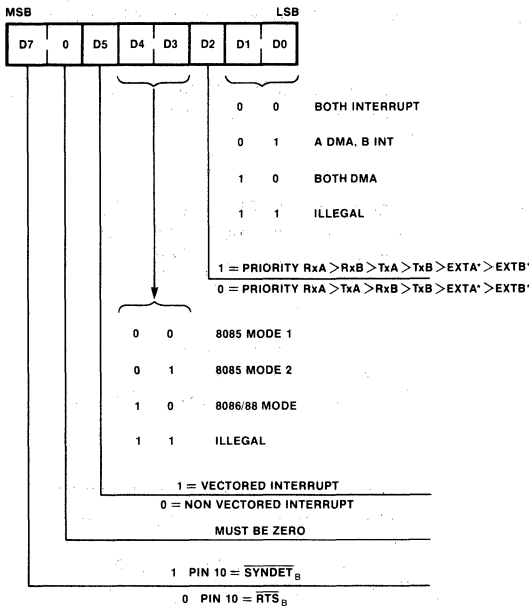
**D0** External/Status Interrupt Enable--allows interrupt to occur as the result of transitions on the CD, CTS or SYNDET inputs. Also allows interrupts as the result of a Break/Abort detection and termination, or at the beginning of CRC, or sync character transmission when the Transmit Underrun/EOM latch becomes set.

**D1** Transmitter Interrupt/DMA Enable--allows the MPSC to interrupt or request a DMA transfer when the transmitter buffer becomes empty.

**D2** Status Affects vector--(WR1,D2 active in channel B only.) If this bit is not set, then the fixed vector, programmed in WR2, is returned from an interrupt acknowledge sequence. If the bit is set then the vector returned from an interrupt acknowledge is variable as shown in the Interrupt Vector Table.

# APPLICATIONS

## Write Register 2 (WR2): Channel A

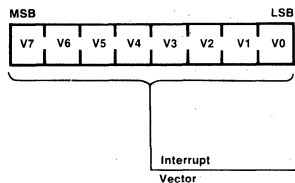


\*EXTERNAL STATUS INTERRUPT-- ONLY IF EXT INTERRUPT ENABLE (WR1: D0) IS SET

- D1, D0** System Configuration--These specify the data transfer from MPSC channels to the CPU, either interrupt or DMA based.
- 0 0 Channel A and Channel B both use interrupts
  - 0 1 Channel A uses DMA, Channel Buses interrupt.
  - 1 0 Channel A and Channel B both use DMA
  - 1 1 Illegal Code
- D2** Priority--this bit specifies the relative priorities of the internal MPSC interrupt/DMA sources.
- 0 (Highest) RxA, TxA, RxB, TxB, ExTA, ExTB (Lowest)
  - 1 (Highest) RxA, RxB, TxA, TxB, ExTA, ExTB (Lowest)

- D5, D4, D3** Interrupt Code--specifies the behavior of the MPSC when it receives an interrupt acknowledge sequence from the CPU. (See Interrupt Vector Mode Table).
- 0 X X Non-vectored interrupts--intended for use with an external interrupt controller such as the 8259A.
  - 1 0 0 8085 Vector Mode 1--intended for use as the primary MPSC in a daisy chained priority structure.
  - 1 0 1 8085 Vector Mode 2--intended for use as any secondary MPSC in a daisy chained priority structure.
  - 1 1 0 8086/88 Vector Mode--intended for use as either a primary or secondary in a daisy chained priority structure.
- D6** Must be zero.
- D7**
- 0 Pin 10 =  $\overline{RTS}_B$
  - 1 Pin 10 =  $\overline{SYNDET}_B$

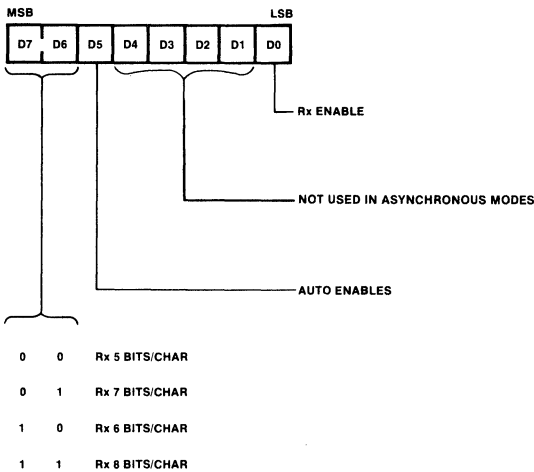
## Write Register 2 (WR2): Channel B



- D7-D0** Interrupt vector--This register contains the value of the interrupt vector placed on the data bus during acknowledge sequences.

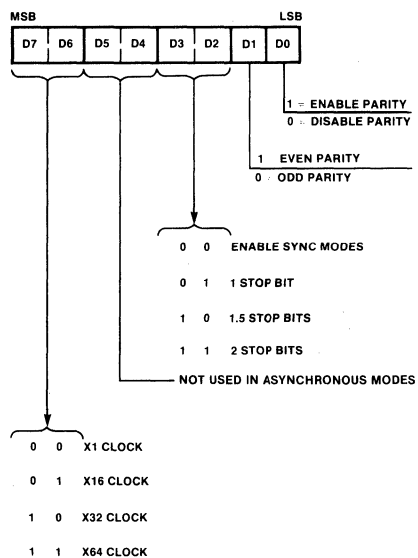
# APPLICATIONS

## Write Register 3 (WR3):



- D0 Receiver Enable--A one enables the receiver to begin. This bit should be set only after the receiver has been initialized.
  
- D5 Auto Enables--A one written to this bit caused  $\overline{CD}$  to be automatic enable signal for the receiver and CTC to be an automatic enable signal for the transmitter. A zero written to this bit limits the effect of  $\overline{CD}$  and CTS signals to setting/resetting their corresponding bits in the status register (RRO).
  
- D7,D6 Receive Character length
  - 0 0 Receive 5 Data bits/character
  - 0 1 Receive 7 Data bits/character
  - 1 0 Receive 6 Data bits/character
  - 1 1 Receive 8 Data bits/character

## Write Register 4 (WR4):



- D0 Parity--a one in this bit causes a parity bit to be added to the programmed number of data bits per character for both the transmitted and received character. If the MPSC is programmed to receive 8 bits per character, the parity bit is not transferred to the microprocessor. With other receiver character lengths, the parity bit is transferred to the microprocessor.
  
- D1 Even/Odd Parity--if parity is enabled, a one in this bit causes the MPSC to transmit and expect even parity, and zero causes it to send and expect odd parity.
  
- D3,D2 Stop Bits
  - 0 0 Selects synchronous modes.
  - 0 1 Async mode, 1 stop bit/character
  - 1 0 Async mode, 1-1/2 stop bits/character
  - 1 1 Async mode, 2 stop bits/character



# APPLICATIONS

D7,D6 Clock mode--selects the clock/data rate multiplier for both the receiver and the transmitter. If the 1x mode is selected, bit synchronization must be done externally.

0 0 Clock rate = Data rate x 1

0 1 Clock rate = Data rate x 16

1 0 Clock rate = Data rate x 32

1 1 Clock rate = Data rate x 64

D6,D5 Transmit Character length

0 0 Transmit 5 or less bits/character

0 1 Transmit 7 bits/character

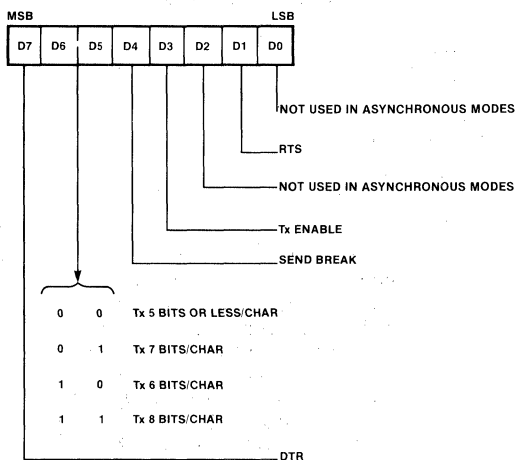
1 0 Transmit 6 bits/character

1 1 Transmit 8 bits/character

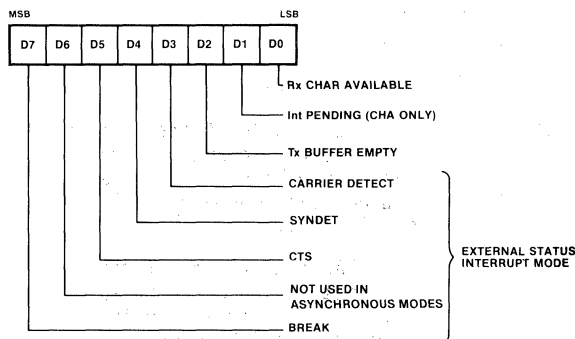
Bits to be sent must be right justified least significant bit first, eg:

D7 D6 D5 D4 D3 D2 D1 D0  
0 0 B5 B4 B3 B2 B1 B0

### Write Register 5 (WR5):



### Read Register 0 (RR0):



D0 Receive Character Available--this bit is set when the receive FIFO contains data and is reset when the FIFO is empty.

D1 Interrupt Pending--This Interrupt-Pending bit is reset when and E01 command is issued and there is no other interrupt request pending at that time. In vector mode this bit is set at the falling edge of the second INTA in an INTA cycle for an internal interrupt request. In non-vector mode, this bit is set at the falling edge of RD input after pointer 2 is specified. This bit is always zero in Channel B.

D2 Transmit Buffer Empty--This bit is set whenever the transmit buffer is empty except when CRC characters are being sent in a synchronous mode. This bit is reset when the transmit buffer is loaded. This bit is set after an MPSC reset.

D1 Request to Send--a one in this bit forces the RTS pin active (low) and zero in this bit forces the RTS pin inactive (high).

D3 Transmitter Enable--a zero in this bit forces a marking state on the transmitter output. If this bit is set to zero during data or sync character transmission, the marking state is entered after the character has been sent. If this bit is set to zero during transmission of a CRC character, sync or flag bits are substituted for the remainder of the CRC bits.

D4 Send Break--a one in this bit forces the transmit data low. A one in this bit allows normal transmitter operation.

# APPLICATIONS

**D3** Carrier Detect--This bit contains the state of the CD pin at the time of the last change of any of the External/Status bits (CD, CTS, Sync/Hunt, Break/Abort, or Tx Underrun/EOM). Any change of state of the CD pin causes the CD bit to be latched and causes an External/Status interrupt. This bit indicates current state of the CD pin immediately following a Reset External/Status Interrupt command.

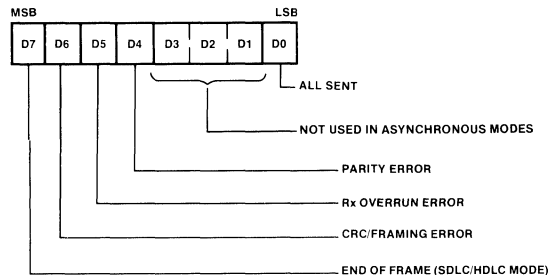
**D4** SYNDET--In asynchronous modes, the operation of this bit is similar to the CD status bit, except that it shows the state of the SYNDET input. Any High-to-Low transition on the SYNDET pin sets this bit, and causes an External/Status interrupt (if enabled). The Reset External/Status Interrupt command is issued to clear the interrupt. A Low-to-High transition clears this bit and sets the External/Status interrupt. When the External/Status interrupt is set by the change in state of any other input or condition, this bit shows the inverted state of the SYNDET pin at time of the change. This bit must be read immediately following a Reset External/Status Interrupt command to read the current state of the SYNDET input.

**D5** Clear to Send--this bit contains the inverted state of the CTS pin at the time of the last change of any of the External/Status bits (CD, CTS, Sync/Hunt, Break/Abort, or Tx Underrun/EOM). Any change of state of the CTS pin causes the CTS bit to be latched and causes an External/Status interrupt. This bit indicates the inverse of the current state of the CTS pin immediately following a Reset External/Status Interrupt command.

**D7** Break--in the Asynchronous Receive mode, this bit is set when a Break sequence (null character plus framing error) is detected in the data stream. The External/Status interrupt, if enabled, is set when break is detected. The interrupt service routine must issue the Reset External/Status Interrupt command (WRO, Command 2) to the break detection logic so the Break sequence termination can be recognized.

The Break bit is reset when the termination of the Break sequence is detected in the incoming data stream. The termination of the Break sequence also causes the External/Status interrupt to be set. The Reset External/Status Interrupt command must be issued to enable the break detection logic to look for the next Break sequence. A single extraneous null character is present in the receiver after the termination of a break; it should be read and discarded.

### Read Register 1 (RR1)



**D0** All sent--this bit is set when all characters have been sent, in asynchronous modes. It is reset when characters are in the transmitter, in asynchronous modes. In synchronous modes, this bit is always set.

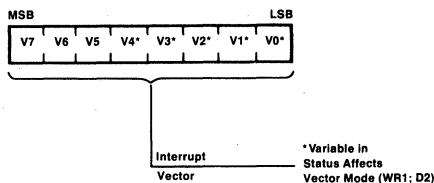
# APPLICATIONS

D4 Parity Error--If parity is enabled, this bit is set for received characters whose parity does not match the programmed sense (Even/Odd). This bit is latched. Once an error occurs, it remains set until the Error Reset command is written.

D5 Receive Overrun Error--this bit indicates that the receive FIFO has been overloaded by the receiver. The last character in the FIFO is overwritten and flagged with this error. Once the overwritten character is read, this error condition is latched until reset by the Error Reset command. If the MPSC is in the status affects vector mode, the overrun causes a special Receive Error Vector.

D6 Framing Error--In async modes, a one in this bit indicates a receive framing error. It can be reset by issuing an Error Reset command.

## Read Register 2 (RR2):



RR2  
D7-D0

Channel B  
Interrupt vector--contains the interrupt vector programmed into WR2. If the status affects vector mode is selected, it contains the modified vector. (See WR2) RR2 contains the modified vector for the highest priority interrupt pending. If no interrupts are pending, the variable bits in the vector are set to one.

# APPLICATIONS

## APPENDIX B: MPSC Polled Transmit/Receive Character Routines

```
MPSCSRX$INIT: procedure (cmd$port,  
                        clock$rate,stop$bits,parity$type,parity$enable,  
                        rx$char$length,rx$enable,auto$enable,  
                        tx$char$length,tx$enable,dtr,brk,rts);  
  
declare cmd$port      byte,  
        clock$rate    byte,  
        stop$bits     byte,  
        parity$type   byte,  
        parity$enable byte,  
        rx$char$length byte,  
        rx$enable     byte,  
        auto$enable   byte,  
        tx$char$length byte,  
        tx$enable     byte,  
        dtr           byte,  
        brk           byte,  
        rts           byte;  
  
output(cmd$port)=30H;          /* channel reset */  
  
output(cmd$port)=14H;          /* point to WR4 */  
/* set clock rate, stop bits, and parity information */  
output(cmd$port)=shl(clock$rate,6) or shl(stop$bits,2) or shl(parity$type,1)  
                or parity$enable;  
  
output(cmd$port)=13H;          /* point to WR3 */  
/* set up receiver parameters */  
output(cmd$port)=shl(rx$char$length,6) or rx$enable or shl(auto$enable,5);  
  
output(cmd$port)=15H;          /* point to WR5 */  
/* set up transmitter parameters */  
output(cmd$port)=shl(tx$char$length,5) or shl(tx$enable,3) or shl(dtr,7)  
                or shl(brk,4) or shl(rts,1);  
  
end MPSCSRX$INIT;
```

# APPLICATIONS

MPSC\$POLL\$RCV\$CHARACTER: procedure(data\$port,cmd\$port,character\$ptr) byte;

```
declare data$port      byte,
        cmd$port       byte,
        character$ptr  pointer,
        character      based character$ptr byte,
        status         byte;

declare char$avail     literally '1',
        rcv$error     literally '70H';

/* wait for input character ready */
while (input(cmd$port) and char$avail) <> 0 do; end;

/* check for errors in received character */
output(cmd$port)=1; /* point to RRI */
if (status:=input(cmd$port) and rcv$error)
  then do;
    character=input(data$port); /* read character to clear MPSC */
    call RECEIVE$ERROR(cmd$port,status); /* clear receiver errors */
    return 0; /* error return - no character avail */
  end;
else do;
  character=input(data$port);
  return 0FFH; /* good return - character avail */
end;

end MPSC$POLL$RCV$CHARACTER;
```

MPSC\$POLL\$TRAN\$CHARACTER: procedure(data\$port,cmd\$port,character);

```
declare data$port      byte,
        cmd$port       byte,
        character      byte;

declare tx$buffer$empty literally '4';

/* wait for transmitter buffer empty */
while not (input(cmd$port) and tx$buffer$empty) do; end;

/* output character */
output(data$port)=character;

end MPSC$POLL$TRAN$CHARACTER;
```

RECEIVE\$ERROR: procedure(cmd\$port,status);

```
declare cmd$port      byte,
        status        byte;

output(cmd$port)=30H; /* error reset */

/* *** other application dependent
   error processing should be placed here *** */

end RECEIVE$ERROR;
```

## APPLICATIONS

---

```
TRANSMIT$BUFFER: procedure(buf$ptr,buf$length)

  declare
    buf$ptr      pointer,
    buf$length   byte;

  /* set up transmit buffer pointer and buffer length in global variables for
     interrupt service */
  tx$buffer$ptr=buf$ptr;
  transmit$length=buf$length;

  transmit$status=not$complete;          /* setup status for not complete */
  output(data$port)=transmit$buffer(0); /* transmit first character */
  transmit$index=1;                      /* first character transmitted */

  /* wait until transmission complete or error detected */
  while transmit$status = not$complete do; end;
  if transmit$status <> complete
    then return false;
    else return true;

end TRANSMIT$BUFFER;
```

```
RECEIVE$BUFFER: procedure (buf$ptr,buf$length$ptr);

  declare
    buf$ptr      pointer,
    buf$length$ptr pointer,
    buf$length   based buf$length$ptr byte;

  /* set up receive buffer pointer in global variable for interrupt service */
  rx$buffer$ptr=buf$ptr;
  receive$index=0;

  receive$status=not$complete;          /* set status to not complete */
  /* wait until buffer received */
  while receive$status = not$complete do; end;
  buf$length=receive$length;
  if receive$status = complete
    then return true;
    else return false;

end RECEIVE$BUFFER;
```

# APPLICATIONS

```
MPSC$RECEIVE$CHARACTER$INT: procedure interrupt 22H;

/* ignore input if no open buffer */
if receive$status <> not$complete then return;

/* check for receive buffer overrun */
if receive$index = 128
then receive$status=overrun;
else do;
  /* read character from MPSC and place in buffer - note that the
  parity of the character must be masked off during this step if
  the character is less than 8 bits (e.g., ASCII) */
  receive$buffer(receive$index),character=input(data$port) and 7FH;
  receive$index=receive$index+1; /* update receive buffer index */

  /* check for line feed to end line */
  if character = line$feed
  then do; receive$length=receive$index; receive$status=complete; end;
end;

end MPSC$RECEIVE$CHARACTER$INT;
```

```
MPSC$TRANSMIT$CHARACTER$INT: procedure interrupt 20H;

/* check for more characters to transfer */
if transmit$index < transmit$length
then do;
  /* write next character from buffer to MPSC */
  output(data$port)=transmit$buffer(transmit$index);
  transmit$index=transmit$index+1; /* update transmit buffer index */
end;
else transmit$status=complete;

end MPSC$TRANSMIT$CHARACTER$INT;
```

```
RECEIVE$ERROR$INT: procedure interrupt 23H;

declare
  temp          byte; /* temporary character storage */

output(cmd$port)=1; /* point to RR1 */
receive$status=input(cmd$port);
temp=input(data$port); /* discard character */
output(cmd$port)=error$reset; /* send error reset */

/* *** other application dependent
error processing should be placed here *** */

end RECEIVE$ERROR$INT;
```

```
EXTERNAL$STATUS$CHANGE$INT: procedure interrupt 21H;

transmit$status=input(cmd$port) /* input status change information */
output(cmd$port)=reset$ext$status;

/* *** other application dependent
error processing should be placed here *** */

end EXTERNAL$STATUS$CHANGE$INT;
```

# APPLICATIONS

## APPENDIX C: Interrupt Driven Transmit/Receive Software

```
declare
/* global variables for buffer manipulation */

rx$buffer$ptr      pointer,          /* pointer to receive buffer */
receive$buffer based rx$buffer$ptr(128) byte,
receive$status     byte initial(0), /* indicates receive buffer status */
receive$index      byte,            /* current index into receive buffer */
receive$length     byte,            /* length of final receive buffer */

tx$buffer$ptr      pointer,          /* pointer to transmit buffer */
transmit$buffer based tx$buffer$ptr(128) byte,
transmit$status    byte initial(0), /* indicates transmit buffer status */
transmit$index     byte,            /* current index into transmit buffer */
transmit$length    byte,            /* length of buffer to be transmitted */

cmd$port           literally `43H`,
data$port          literally `41H`,
a$cmd$port         literally `42H`,
b$cmd$port         literally `43H`,
line$feed          literally `0AH`,
not$complete       literally `0`,
complete           literally `0FFH`,
overrun            literally `1`,

channel$reset      literally `18H`,
error$reset        literally `30H`,
reset$ext$status   literally `10H`;
```



## APPLICATIONS

```
MPSC$INT$INIT: procedure (clock$rate, stop$bits, parity$type, parity$enable,
                          rx$char$length, rx$enable, auto$enable,
                          tx$char$length, tx$enable, dtr, brk, rts,
                          ext$en, tx$en, rx$en, stat$affects$vector,
                          config, priority, vector$int$mode, int$vector);

declare
  clock$rate      byte,      /* 2-bit code for clock rate divisor */
  stop$bits       byte,      /* 2-bit code for number of stop bits */
  parity$type     byte,      /* 1-bit parity type */
  parity$enable   byte,      /* 1-bit parity enable */
  rx$char$length  byte,      /* 2-bit receive character length */
  rx$enable       byte,      /* 1-bit receiver enable */
  auto$enable     byte,      /* 1-bit auto enable flag */
  tx$char$length  byte,      /* 2-bit transmit character length */
  tx$enable       byte,      /* 1-bit transmitter enable */
  dtr             byte,      /* 1-bit status of DTR pin */
  brk             byte,      /* 1-bit data link break enable */
  rts             byte,      /* 1-bit status of RTS pin */
  ext$en         byte,      /* 1-bit external/status enable */
  tx$en          byte,      /* 1-bit Tx interrupt enable */
  rx$en          byte,      /* 2-bit Rx interrupt enable/mode */
  stat$affects$vector byte, /* 1-bit status affects vector flag */
  config         byte,      /* 2-bit system config - int/DMA */
  priority       byte,      /* 1-bit priority flag */
  vector$int$mode byte,      /* 3-bit interrupt mode code */
  int$vector     byte;      /* 8-bit interrupt type code */

output(b$cmd$port)=channel$reset; /* channel reset */

output(b$cmd$port)=14H;           /* point to WR4 */
/* set clock rate, stop bits, and parity information */
output(b$cmd$port)=shl(clock$rate,6) or shl(stop$bits,2) or shl(parity$type,1)
                  or parity$enable;

output(b$cmd$port)=13H;           /* point to WR3 */
/* set up receiver parameters */
output(b$cmd$port)=shl(rx$char$length,6) or rx$enable or shl(auto$enable,5);

output(b$cmd$port)=15H;           /* point to WR5 */
/* set up transmitter parameters */
output(b$cmd$port)=shl(tx$char$length,5) or shl(tx$enable,3) or shl(dtr,7)
                  or shl(brk,4) or shl(rts,1);

output(b$cmd$port)=12H;           /* point to WR2 */
/* set up interrupt vector */
output(b$cmd$port)=int$vector;

output(a$cmd$port)=12H;           /* point to WR2, channel A */
/* set up interrupt modes */
output(a$cmd$port)=shl(vector$int$mode,3) or shl(priority,2) or config;

output(b$cmd$port)=11H;           /* point to WR1 */
/* set up interrupt enables */
output(b$cmd$port)=shl(rx$en,3) or shl(stat$affects$vector,2) or shl(tx$en,1)
                  or ext$en;

end MPSC$INT$INIT;
```

## Appendix D

### Application Example Using SDK-86

This application example shows the 8274 in simple iAPX-86/88 system. The 8274 controls two separate asynchronous channels using its internal interrupt controller to request all data transfers. The 8274 driver software is described which transmits and receives data buffers provided by the CPU. Also, status registers are maintained in system memory to allow the CPU to monitor progress of the buffers and error conditions.

#### THE HARDWARE INTERFACE

Nothing could be easier than the hardware design of an interrupt-driven 8274 system. Simply connect the data bus lines, a few bus control lines, supply a timing clock for baud rate and, voila, it's done! For this example, the ubiquitous SDK-86 is used as the host CPU system. The 8274 interface is constructed on the wire-wrap area provided. While discussing the hardware interface, please refer to Diagram 1.

Placing the 8274 on the lower 8-bits of the 8086 data bus allows byte-wide data transfers at even I/O addresses. For simplicity, the 8274's CS/ input is generated by combining the M-I0/ select line with address line A7 via a 7432. This places the 8274 address range in multiple spots within the 8086 I/O address space. (While fine for this example, a more complete address decoding is recommended for actual prototype systems.) The 8086's A1 and A2 address lines are connected to the A0 and A1 8274 register select inputs respectively. Although other port assignments are possible because of the overlapping address spaces, the following I/O port assignments are used in this example:

<u>Port Function</u>	<u>I/O Address</u>
Data channel A	0000H
Command/status A	0002H
Data channel B	0004H
Command/status B	0006H

To connect the 8274's interrupt controller into the system an inverter and pull-up resistor are needed to convert the 8274's active-low interrupt request output, IRQ, into the correct polarity for the 8086's INTR interrupt input. The 8274 recognizes interrupt acknowledge bus cycles by connecting the INTA (INTerrupt Acknowledge) lines of the 8274 and 8086 together.

The 8274 Read and Write lines directly connect to the respective 8086 lines. The RESET line requires an inverter. The system clock for the 8274 is provided by the PCLK (peripheral clock) output of the 8284A clock generator.

On the 8274's serial side, traditional 1488 and 1489 RS-232 drivers and receivers are used for the serial interface. The onboard baud rate generator supplies the channel baud rate timing. In this example, both sides of both channels operate at the same baud rate although this certainly is not a requirement. (On the SDK-86, the baud rate selection is hard-wired thru jumpers. A more flexible approach would be to incorporate a 8253 Programmable Interval Timer to allow software-configurable baud rate selection.)

That's all there is to it. This hardware interface is completely general-purpose and supports all of the 8274 features except the DMA data transfer mode which requires an external DMA controller. Now let's look at the software interface.

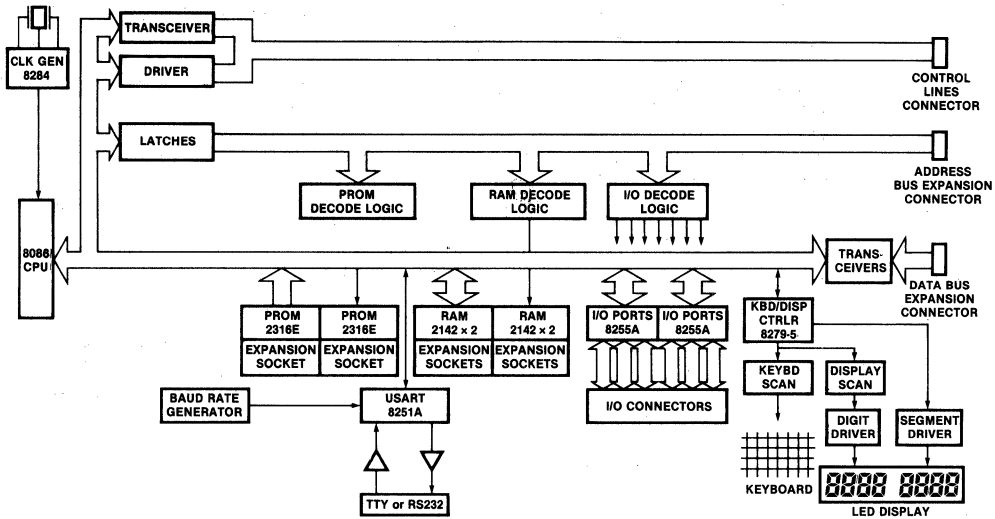
#### SOFTWARE INTERFACE

In this example, it is assumed that the 8086 has better things to do rather than continuously run a serial channel. Presenting the software as a group of callable procedures lets the designer include them in the main body of another program. The interrupt-driven data transfers give the effect that the serial channels are handled in the background while the main program is executing in the foreground. There are five basic procedures: a serial channel initialization routine and buffer handling routines for the transmit and receive data buffers of each channel. Appendix D-1 shows the entire software listing. Listing line numbers are referenced as each major routing is discussed.

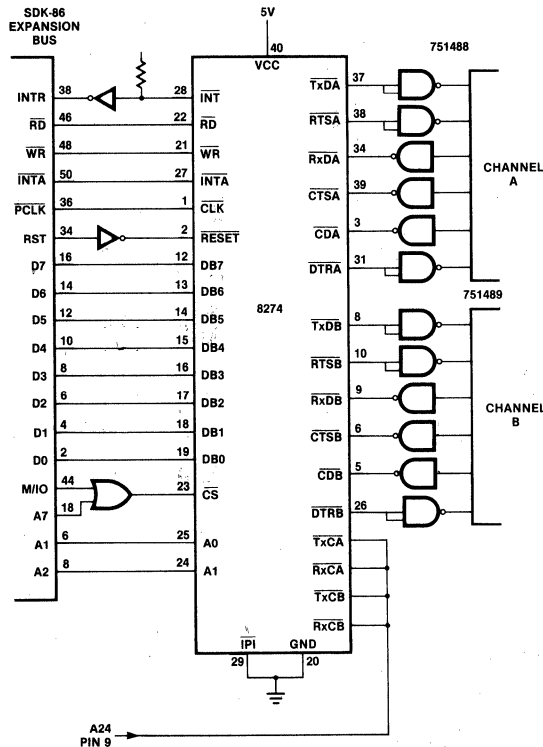
The channel initialization routine (INITIAL 8274), starting with line #203, simply sets each channel into a particular operating mode by loading the command registers of the 8274. In normal operation, once these registers are loaded, they are rarely changed. (Although this example assumes a simple asynchronous operating mode, the concept is easily extended for the byte and bit-synchronous modes.)

The channel operating modes are contained in two tables starting with line #163. As the 8274 has only one command register per channel, the remaining seven registers are

# APPLICATIONS



(For detailed description on SDK-86, refer to SDK-86 MCS-86 System Design Kit Assembly Manual)



8274/SDK-86 Hardware Interface

Diagram 1

## APPLICATIONS

loaded indirectly through the WRO (Write Register 0) register. The first byte of each table entry is the register pointer value which is loaded into WRO and the second byte is the value for that particular register.

The indicated modes set the 8274 for asynchronous operation with data characters 8 bits long, no parity, and 2 stop bits. A X16 baud rate clock is assumed. Also selected is the "interrupt on all RX character" mode with a variable interrupt vector compatible with the 8086/8088. The transmitters are enabled and all mode1 control lines are put in their active state.

In addition to initializing the 8274, this routine also sets up the appropriate interrupt vectors. The 8086 assumes the first 1K bytes of memory contain up to 256 separate interrupt vectors. On the SDK-86 the initial 2K bytes of memory is RAM and therefore must be initialized with the appropriate vectors. (In a prototype system, this initial memory is probably ROM thus the vector set-up is not needed.) The 8274 supplies up to eight different interrupt vectors. These vectors are developed from internal conditions such as data requests, status changes, or error conditions for each channel. The initialization routine arbitrarily assumes that the initial 8274 vector corresponds to 8086 vector location 80H (memory location 200H). This choice is arbitrary since the 8274 initial vector location is programmable.

Finally, the initialization routine sets up the status and flag in RAM. The meaning and use of these locations are discussed later.

Following the initialization routine are those for the transmit commands (starting with line #268). These commands assume that the host CPU has initialized the publically declared variables for the transmit buffer pointer, TX\_POINTER\_CHx, and the buffer length, TX\_LENGTH\_CHx. The transmit command routines simply clear the transmitter empty flag, TX\_EMPTY\_CHx, and load the first character of the buffer into the transmitter. It is necessary to load the first character in this manner since transmitter interrupts are generated only when the 8274's transmit data buffer becomes empty. It is the act of becoming empty which generates the interrupt not simply the buffer being empty, thus the transmitter needs one character to start.

The host CPU can monitor the transmitter empty flag, TX\_EMPTY\_CHx, in order to determine when transmission of the buffer is complete. Obviously, the CPU should only call the command routine after first checking that the empty flag is set.

After returning to the main program, all transmitter data transfers are handled via the transmitter interrupt service routines starting at lines #360 and #443. These routines start by issuing an End-Of-Interrupt command to the 8274. (This command resets the internal interrupt controller logic of the 8274 for this particular vector and opens the logic for other internal interrupt requests. The routines next check the length count. If the buffer is completely transmitted, the transmitter empty flag, TX\_EMPTY\_CHx, is set and a command is issued to the 8274 to reset its interrupt line. Assuming that the buffer is not completely transmitted, the next character is output to the transmitter. In either case, an interrupt return is executed to return to the main CPU program.

The receiver commands start at line #314. Like the transmit commands, it is assumed that the CPU has initialized the receive buffer pointer public variable, RX\_POINTER\_CHx. This variable points to the first location in an empty receive buffer. The command routines clear the receiver ready flag, RX\_READY\_CHx, and then set the receiver enable bit in the 8274 WR3 register. With the receiver now enabled, any received characters are placed in the receive buffer using interrupt-driven data transfers.

The received data service routines, starting at lines #402 and #485, simply place the received character in the buffer after first issuing the EOI command. The character is then compared to a ASCII CR. An ASCII CR causes the routine to set the receiver ready flag, RX\_READY\_CHx, and to disable the receiver. The CPU can interrogate this flag to determine when the buffer contains a new line of data. The receive buffer pointer, RX\_POINTER\_CHx, points to the last received character and the receive counter, RX\_COUNTER\_CHx, contains the length.

That completes our discussion of the command routines and their associated interrupt service routines. Although not used by the commands, two additional service routines are included for completeness. These routines handle the error and status-change interrupt vectors.

# APPLICATIONS

The error service routines, starting at lines #427 and #510, are vectored to if a special receive condition is detected by the 8274. These special receive conditions include parity, receiver overrun, and framing errors. When this vector is generated, the error condition is indicated in RRI (Read Register 1). The error service routine issues an EOI command, reads RRI and places it in the ERROR\_MSG\_CHx variable, and then issues a reset error command to the 8274. The CPU can monitor the error message location to detect error conditions. The designer, of course, can supply his own error service routine.

Similarly, the status-change routines (starting lines #386 and #469) are initiated by a change in the modem control status lines CTS/, CD/, or SYNDET/. (Note that WR2 bit 0 controls whether the 8274

generates interrupts based upon changes in these lines. Our WR2 parameter is such that the 8274 is programmed to ignore changes for these inputs.) The service routines simply read RRO, place its contents in the STATUS\_MSG\_CHx variable and then issue a reset external status command. Read Register 0 contains the state of the modem inputs at the point of the last change.

Well, that's it. This application example has presented useful, albeit very simple, routines showing how the 8274 might be used to transmit and receive buffers using an asynchronous serial format. Extensions for byte or bit-synchronous formats would require no hardware changes due to the highly programmable nature of the 8294's serial formats.

## Appendix D-1

MCS-86 MACRO ASSEMBLER ASYNCR

PAGE 1

ISIS-II MCS-86 MACRO ASSEMBLER V2.1 ASSEMBLY OF MODULE ASYNCR  
OBJECT MODULE PLACED IN :F1:ASYNCR.OBJ  
ASSEMBLER INVOKED BY: ASM86 :F1:ASYNCR.SRC

LOC	OBJ	LINE	SOURCE
		1	;*****
		2	;*
		3	;* 8274 APPLICATION BRIEF PROGRAM *
		4	;*
		5	;*
		6	;*
		7	;* THE 8274 IS INITIALIZED FOR SIMPLE ASYNCHRONOUS SERIAL *
		8	;* FORMAT AND VECTORED INTERRUPT-DRIVEN DATA TRANSFERS. *
		9	;* THE INITIALIZATION ROUTINE ALSO LOADS THE 8086'S INTERRUPT *
		10	;* VECTOR TABLE FROM THE CODE SEGMENT INTO LOW RAM ON THE *
		11	;* SDK-86. THE TRANSMITTER AND RECEIVER ARE LEFT ENABLED. *
		12	;*
		13	;* FOR TRANSMIT, THE CPU PASSES IN MEMORY THE POINTER OF A *
		14	;* BUFFER TO TRANSMIT AND THE BYTE LENGTH OF THE BUFFER. *
		15	;* THE DATA TRANSFER PROCEED USING INTERRUPT-DRIVEN TRANSFERS. *
		16	;* A STATUS BIT IN MEMORY IS SET WHEN IF BUFFERS IS EMPTY. *
		17	;*
		18	;* FOR RECEIVE, THE CPU PASSES THE POINTER OF A BUFFER TO FILL. *
		19	;* THE BUFFER IS FILLED UNTIL A 'CR_LCHR' CHARACTER IS RECEIVED. *
		20	;* A STATUS BIT IS SET AND THE CPU MAY READ THE RX POINTER TO *
		21	;* DETERMINE THE LOCATION OF THE LAST CHARACTER. *
		22	;*
		23	;* ALL ROUTINES ARE ASSUMED TO EXIST IN THE SAME CODE SEGMENT. *
		24	;* CALL'S TO THE SERVICE ROUTINES ARE ASSUMED TO BE "SHORT" OR *
		25	;* INTRASEGMENT (ONLY THE RETURN ADDRESS IP IS ON THE STACK). *
		26	;*
		27	;*
		28	;*
		29	;*
		30	;*****

# APPLICATIONS

```

LOC OBJ          LINE   SOURCE
31
32              NAME   ASYNCB ;MODULE NAME
33
34 ;PUBLIC DECLARATIONS FOR COMMAND ROUTINES
35
36 PUBLIC INITIAL_8274 ;INITIALIZATION ROUTINE
37 PUBLIC TX_COMMAND_CHB ;TX BUFFER COMMAND CHANNEL B
38 PUBLIC TX_COMMAND_CHA ;TX BUFFER COMMAND CHANNEL A
39 PUBLIC RX_COMMAND_CHB ;RX BUFFER COMMAND CHANNEL B
40 PUBLIC RX_COMMAND_CHA ;RX BUFFER COMMAND CHANNEL A
41
42 ;PUBLIC DECLARATIONS FOR STATUS VARIABLES
43
44 PUBLIC RX_READY_CHB ;RX READY FLAG CHB
45 PUBLIC RX_READY_CHA ;RX READY FLAG CHA
46 PUBLIC TX_EMPTY_CHB ;TX EMPTY FLAG CHB
47 PUBLIC TX_EMPTY_CHA ;TX EMPTY FLAG CHA
48 PUBLIC RX_COUNT_CHB ;RX BUFFER COUNTER CHB
49 PUBLIC RX_COUNT_CHA ;RX BUFFER COUNTER CHA
50 PUBLIC ERROR_MSG_CHB ;ERROR FLAG CHB
51 PUBLIC ERROR_MSG_CHA ;ERROR FLAG CHA
52 PUBLIC STATUS_MSG_CHB ;STATUS FLAG CHB
53 PUBLIC STATUS_MSG_CHA ;STATUS FLAG CHA
54
55 ;PUBLIC DECLARATIONS FOR VARIABLES PASSED TO THE TRANSMIT
56 ;AND RECEIVE COMMANDS.
57
58 PUBLIC TX_POINTER_CHB ;TX BUFFER POINTER FOR CHB
59 PUBLIC TX_LENGTH_CHB ;TX LENGTH OF BUFFER FOR CHB
60 PUBLIC TX_POINTER_CHA ;TX BUFFER POINTER FOR CHA
61 PUBLIC TX_LENGTH_CHA ;TX LENGTH OF BUFFER FOR CHA
62 PUBLIC RX_POINTER_CHB ;RX BUFFER POINTER FOR CHB
63 PUBLIC RX_POINTER_CHA ;RX BUFFER POINTER FOR CHA
64
65 ;I/O PORT ASSIGNMENTS
66
67 ;CHANNEL A PORT ASSIGNMENTS
68
0000 69 DATA_PORT_CHA EQU 0 ;DATA I/O PORT
0002 70 COMMAND_PORT_CHA EQU 2 ;COMMAND PORT
0002 71 STATUS_PORT_CHA EQU COMMAND_PORT_CHA ;STATUS PORT
72
73 ;CHANNEL B PORT ASSIGNMENTS
74
0004 75 DATA_PORT_CHB EQU 4 ;DATA I/O PORT
0006 76 COMMAND_PORT_CHB EQU 6 ;COMMAND PORT
0006 77 STATUS_PORT_CHB EQU COMMAND_PORT_CHB ;STATUS PORT
78
79 ;MISC. SYSTEM EQUATES
80
0000 81 CR_CHR EQU 0DH ;ASCII CR CHARACTER CODE
0200 82 INT_TABLE_BASE EQU 200H ;INT. VECTOR BASE ADDRESS
0500 83 CODE_START EQU 500H ;START LOCATION FOR CODE
84
85 +1 $EJECT
86
87 ;RAM ASSIGNMENTS FOR DATA SEGMENT
88
89 DATA SEGMENT
90

```

# APPLICATIONS

```

LOC  OBJ                LINE  SOURCE
                                91  ;VECTOR INTERRUPT TABLE - ASSUME INITIAL 8274 INTERRUPT
                                92  ;VECTOR IS NUMBER 80 (@200H).  FOR EACH VECTOR, THE TABLE
                                93  ;CONTAINS START LOCATION AND CODE SEGMENT REGISTER VALUE.
                                94  ;THE TABLE IS LOADED FROM PROM.
                                95
0200                                96          ORG      INT_TABLE_BASE
                                97
0200 0000                98  TX_VECTOR_CHB  DW      0      ;TX INTERRUPT VECTOR FOR CHB
0202 0000                99  TX_CS_CHB     DW      0
                                100
0204 0000                101  STS_VECTOR_CHB DW      0      ;STATUS INTERRUPT VECTOR FOR CHB
0206 0000                102  STS_CS_CHB    DW      0
                                103
0208 0000                104  RX_VECTOR_CHB DW      0      ;RX INTERRUPT VECTOR FOR CHB
020A 0000                105  RX_CS_CHB    DW      0
                                106
020C 0000                107  ERR_VECTOR_CHB DW      0      ;ERROR INTERRUPT VECTOR FOR CHB
020E 0000                108  ERR_CS_CHB   DW      0
                                109
0210 0000                110  TX_VECTOR_CHA  DW      0      ;TX INTERRUPT VECTOR FOR CHA
0212 0000                111  TX_CS_CHA     DW      0
                                112
0214 0000                113  STS_VECTOR_CHA DW      0      ;STATUS INTERRUPT VECTOR FOR CHA
0216 0000                114  STS_CS_CHA    DW      0
                                115
0218 0000                116  RX_VECTOR_CHA  DW      0      ;RX INTERRUPT VECTOR FOR CHA
021A 0000                117  RX_CS_CHA     DW      0
                                118
021C 0000                119  ERR_VECTOR_CHA DW      0      ;ERROR INTERRUPT VECTOR FOR CHA
021E 0000                120  ERR_CS_CHA    DW      0
                                121
                                122  ;MISC RAM LOCATIONS FOR CHANNEL STATUS AND POINTERS
                                123
                                124  ;CHANNEL B POINTERS AND STATUS
                                125
0220 0000                126  TX_POINTER_CHB DW      0      ;TX BUFFER POINTER FOR CHB
0222 0000                127  TX_LENGTH_CHB DW      0      ;TX BUFFER LENGTH FOR CHB
0224 0000                128  RX_POINTER_CHB DW      0      ;RX BUFFER POINTER FOR CHB
0226 0000                129  RX_COUNT_CHB  DW      0      ;RX LENGTH COUNTER FOR CHB
0228 00                   130  TX_EMPTY_CHB  DB      0      ;TX DONE FLAG
0229 00                   131  RX_READY_CHB  DB      0      ;READY FLAG (1 IF CR_CHR RECEIVED, ELSE 0)
022A 00                   132  STATUS_MSG_CHB DB      0      ;STATUS CHANGE MESSAGE
022B 00                   133  ERROR_MSG_CHB DB      0      ;ERROR STATUS LOCATION (0 IF NO ERROR)
                                134
                                135  ;CHANNEL A POINTERS AND STATUS
                                136
022C 0000                137  TX_POINTER_CHA DW      0      ;TX BUFFER POINTER FOR CHA
022E 0000                138  TX_LENGTH_CHA DW      0      ;TX BUFFER LENGTH FOR CHA
0230 0000                139  RX_POINTER_CHA DW      0      ;RX BUFFER POINTER FOR CHA
0232 0000                140  RX_COUNT_CHA  DW      0      ;RX LENGTH COUNTER FOR CHA
0234 00                   141  TX_EMPTY_CHA  DB      0      ;TX DONE FLAG
0235 00                   142  RX_READY_CHA  DB      0      ;READY FLAG (1 IF CR_CHR RECEIVED, ELSE 0)
0236 00                   143  STATUS_MSG_CHA DB      0      ;STATUS CHANGE MESSAGE
0237 00                   144  ERROR_MSG_CHA DB      0      ;ERROR STATUS LOCATION (0 IF NO ERROR)
                                145
                                146          DATA  ENDS
                                147
                                148  *1 $EJECT

```

# APPLICATIONS

```

LOC  OBJ          LINE   SOURCE
-----
                                149
                                150          ABC   SEGMENT
                                151      ASSUME  CS:ABC, DS:DATA, SS:DATA
0500   152          ORG   CODE_START
                                153
                                154      ;*****
                                155      ;*
                                156          PARAMETERS FOR CHANNEL INITIALIZATION      *
                                157      ;*
                                158      ;*****
                                159
                                160      ; CHANNEL B PARAMETERS
                                161
                                162          ; WR1 - INTERRUPT ON ALL RX CHR, VARIABLE INT VECTOR, TX INT ENABLE
0500  01          163      CMDSTRB DB   1,16H
0501  16
                                164          ; WR2 - INTERRUPT VECTOR
0502  02          165      DB   2,(INT_TABLE_BASE/4)
0503  00
                                166          ; WR3 - RX 8 BITS/CHR, RX DISABLE
0504  03          167      DB   3,0C0H
0505  C0
                                168          ; WR4 - X16 CLOCK, 2 STOP BITS, NO PARITY
0506  04          169      DB   4,4CH
0507  4C
                                170          ; WR5 - DTR ACTIVE, TX 8 BITS/CHR, TX ENABLE, RTS ACTIVE
0508  05          171      DB   5,0EAH
0509  EA
                                172          ; WR6 AND WR7 NOT REQUIRED FOR ASYNC
050A  00          173      DB   0,0
050B  00
                                174
                                175      ; CHANNEL A PARAMETERS
                                176
                                177          ; WR1 - INTERRUPT ON ALL RX CHR, TX INT ENABLE
050C  01          178      CMDSTRA DB   1,12H
050D  12
                                179          ; WR2 - VECTORED INTERRUPT FOR 8086
050E  02          180      DB   2,30H
050F  30
                                181          ; WR3 - RX 8 BITS/CHR, RX DISABLE
0510  03          182      DB   3,0C0H
0511  C0
                                183          ; WR4 - X16 CLOCK, 2 STOP BITS, NO PARITY
0512  04          184      DB   4,4CH
0513  4C
                                185          ; WR5 - DTR ACTIVE, TX 8 BITS/CHR, TX ENABLE, RTS ACTIVE
0514  05          186      DB   5,0EAH
0515  EA
                                187          ; WR6 AND WR7 NOT REQUIRED FOR ASYNC
0516  00          188      DB   0,0
0517  00
                                189
                                190  +1  $EJECT

```



# APPLICATIONS

```

LOC OBJ          LINE   SOURCE
191
192      ;START OF COMMAND ROUTINES
193
194      ;*****
195      ;*
196      ;*   INITIALIZATION COMMAND FOR THE 8274 - THE 8274   *
197      ;*   IS SETUP ACCORDING TO THE PARAMETERS STORED IN   *
198      ;*   PROM ABOVE STARTING AT CMDSTRB FOR CHANNEL B AND   *
199      ;*   CMSTRA FOR CHANNEL A.                             *
200      ;*
201      ;*****
202
0518          203      INITIAL_8274:
204              ;COPY INTERRUPT VECTOR IP AND CS VALUES FROM PROM TO RAM
0518 C70600020806 205      MOV     TX_VECTOR_CHB, OFFSET XMTINB ;TX DATA VECTOR CHB
051E 8C0E0202     206      MOV     TX_CS_CHB, CS
0522 C70604023506 207      MOV     STS_VECTOR_CHB, OFFSET STAINB ;STATUS VECTOR CHB
0528 8C0E0602     208      MOV     STS_CS_CHB, CS
052C C70608024906 209      MOV     RX_VECTOR_CHB, OFFSET RCVINB ;RX DATA VECTOR CHB
0532 8C0E0A02     210      MOV     RX_CS_CHB, CS
0536 C7060C027706 211      MOV     ERR_VECTOR_CHB, OFFSET ERRINB ;ERROR VECTOR CHB
053C 8C0E0A02     212      MOV     RX_CS_CHB, CS
0540 C70610028C06 213      MOV     TX_VECTOR_CHA, OFFSET XMTINA ;TX DATA VECTOR CHA
0546 8C0E1202     214      MOV     TX_CS_CHA, CS
054A C7061402B906 215      MOV     STS_VECTOR_CHA, OFFSET STAINA ;STATUS VECTOR CHA
0550 8C0E1602     216      MOV     STS_CS_CHA, CS
0554 C7061802CD06 217      MOV     RX_VECTOR_CHA, OFFSET RCVINA ;RX DATA VECTOR CHA
055A 8C0E1A02     218      MOV     RX_CS_CHA, CS
055E C7061C02FB06 219      MOV     ERR_VECTOR_CHA, OFFSET ERRINA ;ERROR VECTOR CHA
0564 8C0E1E02     220      MOV     ERR_CS_CHA, CS
221
222      ;COPY SETUP TABLE PARAMETERS INTO 8274
223
0568 BF0005       224      MOV     DI, OFFSET CMDSTRB ;INITIALIZE CHB
056E BA0600       225      MOV     DX, COMMAND_PORT_CHB
056E E82E00       226      CALL    SETUP ;COPY CHB PARAMETERS
0571 BF0C05       227      MOV     DI, OFFSET CMDSTRA ;INITIALIZE CHA
0574 BA0200       228      MOV     DX, COMMAND_PORT_CHA
0577 E82500       229      CALL    SETUP ;COPY CHA PARAMETERS
230
231      INITIALIZE STATUS BYTES AND FLAGS
232
057A B80000       233      MOV     AX, 0
057D A22B02       234      MOV     ERROR_MSG_CHB, AL ;CLEAR ERROR FLAG CHB
0580 A23702       235      MOV     ERROR_MSG_CHA, AL ;CLEAR ERROR FLAG CHA
0583 A22A02       236      MOV     STATUS_MSG_CHB, AL ;CLEAR STATUS FLAG CHB
0586 A23602       237      MOV     STATUS_MSG_CHA, AL ;CLEAR STATUS FLAG CHA
0589 A32602       238      MOV     RX_COUNT_CHB, AX ;CLEAR RX COUNTER CHB
058C A33202       239      MOV     RX_COUNT_CHA, AX ;CLEAR RX COUNTER CHA
058F B001         240      MOV     AL, 1
0591 A22902       241      MOV     RX_READY_CHB, AL ;SET RX DONE FLAG CHB
0594 A23502       242      MOV     RX_READY_CHA, AL ;SET RX DONE FLAG CHA
0597 A22802       243      MOV     TX_EMPTY_CHB, AL ;SET TX DONE FLAG CHB
059A A23402       244      MOV     TX_EMPTY_CHA, AL ;SET TX DONE FLAG CHA
059D FB         245      STI ;ENABLE INTERRUPTS
059E C3         246      RET ;RETURN - DONE WITH SETUP
247
059F 8A05         248      SETUP: MOV     AL, [DI] ;PARAMETER COPYING ROUTINE
05A1 3C00         249      CMP     AL, 0
05A3 7404         250      JE     DONE

```

# APPLICATIONS

```

LOC OBJ          LINE   SOURCE
05A5 EE          251       OUT   DX, AL           ; OUTPUT PARAMETER
05A6 47          252       INC   DI               ; POINT AT NEXT PARAMETER
05A7 EBF6        253       JMP   SETUP           ; GO LOAD IT
05A9 C3          254   DONE:  RET           ; DONE - SO RETURN
                255
                256 +1 $EJECT
                257
                258 ;*****
                259 ;*
                260 ;* TX CHANNEL B COMMAND ROUTINE - ROUTINE IS CALLED TO
                261 ;* TRANSMIT A BUFFER. THE BUFFER STARTING ADDRESS,
                262 ;* TX_POINTER_CHB, AND THE BUFFER LENGTH, TX_LENGTH_CHB,
                263 ;* MUST BE INITIALIZED BY THE CALLING PROGRAM.
                264 ;* BOTH ITEMS ARE WORD VARIABLES.
                265 ;*
                266 ;*****
                267
05AA             268   TX_COMMAND_CHB:
05AA 50          269       PUSH  AX           ; SAVE REGISTERS
05AB 57          270       PUSH  DI
05AC 52          271       PUSH  DX
05AD C606280200  272       MOV   TX_EMPTY_CHB, 0 ; CLEAR EMPTY FLAG
05B2 BA0400      273       MOV   DX, DATA_PORT_CHB ; SETUP PORT POINTER
05B5 8B3E2002    274       MOV   DI, TX_POINTER_CHB ; GET TX BUFFER POINTER CHB
05B9 8A05        275       MOV   AL, [DI]        ; GET FIRST CHARACTER TO TX
05BB EE          276       OUT   DX, AL       ; OUTPUT IT TO 8274 TO GET IT STARTED
05BC 5A          277       POP   DX
05BD 5F          278       POP   DI
05BE 58          279       POP   AX
05BF C3          280       RET           ; RETURN
                281
                282 ;*****
                283 ;*
                284 ;* TX CHANNEL A COMMAND ROUTINE - ROUTINE IS CALLED TO
                285 ;* TRANSMIT A BUFFER. THE BUFFER STARTING ADDRESS,
                286 ;* TX_POINTER_CHA, AND THE BUFFER LENGTH, TX_LENGTH_CHA,
                287 ;* MUST BE INITIALIZED BY THE CALLING PROGRAM.
                288 ;* BOTH ITEMS ARE WORD VARIABLES.
                289 ;*
                290 ;*****
                291
05C0             292   TX_COMMAND_CHA:
05C0 50          293       PUSH  AX           ; SAVE REGISTERS
05C1 57          294       PUSH  DI
05C2 52          295       PUSH  DX
05C3 C606340200  296       MOV   TX_EMPTY_CHA, 0 ; CLEAR EMPTY FLAG
05C8 BA0000      297       MOV   DX, DATA_PORT_CHA ; SETUP PORT POINTER
05CB 8B3E2C02    298       MOV   DI, TX_POINTER_CHA ; GET TX BUFFER POINTER CHA
05CF 8A05        299       MOV   AL, [DI]        ; GET FIRST CHARACTER TO TX
05D1 EE          300       OUT   DX, AL       ; OUTPUT IT TO 8274 TO GET IT STARTED
05D2 5A          301       POP   DX
05D3 5F          302       POP   DI
05D4 58          303       POP   AX
05D5 C3          304       RET           ; RETURN
                305
                306 ;*****
                307 ;*
                308 ;* RX COMMAND FOR CHANNEL B - THE CALLING ROUTINE MUST
                309 ;* INITIALIZE RX_POINTER_CHB TO POINT AT THE RECEIVE
                310 ;* BUFFER BEFORE CALLING THIS ROUTINE.

```

# APPLICATIONS

MCS-86 MACRO ASSEMBLER    ASYNCR

PAGE 7

```

LOC  OBJ          LINE  SOURCE
                                     311  ;*
                                     312  ;*****
                                     313
05D6          314  RX_COMMAND_CHB:
05D6 50         315          PUSH  AX           ;SAVE REGISTERS
05D7 52         316          PUSH  DX
05D8 C606290200 317          MOV   RX_READY_CHB, 0 ;CLEAR RX READY FLAG
05D9 C70626020000 318          MOV   RX_COUNT_CHB, 0 ;CLEAR RX COUNTER
05E3 B00600     319          MOV   DX, COMMAND_PORT_CHB ;POINT AT COMMAND PORT
05E6 B003      320          MOV   AL, 3           ;SET UP FOR WR3
05E8 EE        321          OUT   DX, AL
05E9 B0C1     322          MOV   AL, 0C1H       ;WR3 - 8 BITS/CHR, ENABLE RX
05EB EE        323          OUT   DX, AL
05EC 5A       324          POP  DX
05ED 58       325          POP  AX
05EE C3       326          RET           ;RETURN
                                     327
                                     328  ;*****
                                     329  ;*
                                     330  ;*  RX COMMAND FOR CHANNEL A - THE CALLING ROUTINE MUST
                                     331  ;*  INITIALIZE RX_POINTER_CHA TO POINT AT THE RECEIVE
                                     332  ;*  BUFFER BEFORE CALLING THIS ROUTINE.
                                     333  ;*
                                     334  ;*****
05EF          335  RX_COMMAND_CHA:
05EF 50         336          PUSH  AX           ;SAVE REGISTERS
05F0 52         337          PUSH  DX
05F1 C606350200 338          MOV   RX_READY_CHA, 0 ;CLEAR RX READY FLAG
05F6 C70632020000 339          MOV   RX_COUNT_CHA, 0 ;CLEAR RX COUNTER
05FC B00200    340          MOV   DX, COMMAND_PORT_CHA ;POINT AT COMMAND PORT
05FF B003      341          MOV   AL, 3           ;SET UP FOR WR3
0601 EE        342          OUT   DX, AL
0602 B0C1     343          MOV   AL, 0C1H       ;WR3 - 8 BITS/CHR, ENABLE RX
0604 EE        344          OUT   DX, AL
0605 5A       345          POP  DX
0606 58       346          POP  AX
0607 C3       347          RET           ;RETURN
0608          348
0609          349
060A          350  +1 $EJECT
060B          351
060C          352  ;*****
060D          353  ;*
060E          354  ;*  START OF INTERRUPT SERVICE ROUTINES
060F          355  ;*
0610          356  ;*****
0611          357
0612          358  ;CHANNEL B TRANSMIT DATA SERVICE ROUTINE
0613          359
0614          360  XMTINB: PUSH  DX           ;SAVE REGISTERS
0615          361          PUSH  DI
0616          362          PUSH  AX
0617          363          CALL  EOI           ;SEND EOI COMMAND TO 8274
0618          364          INC  TX_POINTER_CHB ;POINT TO NEXT CHARACTER
0619          365          DEC  TX_LENGTH_CHB ;DEC LENGTH COUNTER
061A          366          JE   XIB           ;TEST IF DONE
061B          367          MOV  DX, DATA_PORT_CHB ;NOT DONE - GET NEXT CHARACTER
061C          368          MOV  DI, TX_POINTER_CHB
061D          369          MOV  AL, [DI]         ;PUT CHARACTER IN AL
061E          370          OUT  DX, AL       ;OUTPUT IT TO 8274

```

# APPLICATIONS

```

LOC  OBJ          LINE  SOURCE
0622 58          371      POP  AX          ;RESTORE REGISTERS
0623 5F          372      POP  DI
0624 5A          373      POP  DX
0625 CF          374      IRET              ;RETURN TO FOREGROUND
0626 BA0600      375      XIB:  MOV  DX, COMMAND_PORT_CHB ;ALL CHARACTERS HAVE BEEN SEND
0629 B028      376      MOV  AL, 28H      ;RESET TRANSMITTER INTERRUPT PENDING
062B EE          377      OUT  DX, AL
062C C006280201 378      MOV  TX_EMPTY_CHB, 1 ;DONE - SO SET TX EMPTY FLAG CHB
0631 58          379      POP  AX          ;RESTORE REGISTERS
0632 5F          380      POP  DI
0633 5A          381      POP  DX
0634 CF          382      IRET              ;RETURN TO FOREGROUND
383
384      ;CHANNEL B STATUS CHANGE SERVICE ROUTINE
385
0635 52          386      STAINB: PUSH DX          ;SAVE REGISTERS
0636 57          387      PUSH DI
0637 50          388      PUSH AX
0638 E00500      389      CALL EDI          ;SEND EOI COMMAND TO 8274
0638 BA0600      390      MOV  DX, COMMAND_PORT_CHB
063E EC          391      IN   AL, DX      ;READ RR0
063F A22A02      392      MOV  STATUS_MSG_CHB, AL ;PUT RR0 IN STATUS MESSAGE
0642 B010      393      MOV  AL, 10H     ;SEND RESET STATUS INT COMMAND TO 8274
0644 EE          394      OUT  DX, AL
0645 58          395      POP  AX          ;RESTORE REGISTERS
0646 5F          396      POP  DI
0647 5A          397      POP  DX
0648 CF          398      IRET
399
400      ;CHANNEL B RECEIVED DATA SERVICE ROUTINE
401
0649 52          402      RCVINB: PUSH DX          ;SAVE REGISTERS
064A 57          403      PUSH DI
064B 50          404      PUSH AX
064C E0C100      405      CALL EDI          ;SEND EOI COMMAND TO 8274
064F 0B3E2402    406      MOV  DI, RX_POINTER_CHB ;GET RX CHB BUFFER POINTER
0653 BA0400      407      MOV  DX, DATA_PORT_CHB
0656 EC          408      IN   AL, DX      ;READ CHARACTER
0657 8805      409      MOV  [DI], AL    ;STORE IN BUFFER
0659 FF062402    410      INC  RX_POINTER_CHB ;BUMP THE BUFFER POINTER
065D FF062602    411      INC  RX_COUNT_CHB  ;BUMP THE COUNTER
0661 3C00      412      CMP  AL, CR_CHR   ;TEST IF LAST CHARACTER TO BE RECEIVED?
0663 750E      413      JNE  RIB
0665 C006290201 414      MOV  RX_READY_CHB, 1 ;YES, SET READY FLAG
066A BA0600      415      MOV  DX, COMMAND_PORT_CHB ;POINT AT COMMAND PORT
066D B003      416      MOV  AL, 3        ;POINT AT WR3
066F EE          417      OUT  DX, AL
0670 B0C0      418      MOV  AL, 0C0H    ;DISABLE RX
0672 EE          419      OUT  DX, AL
0673 58          420      RIB:  POP  AX          ;EITHER WAY, RESTORE REGISTERS
0674 5F          421      POP  DI
0675 5A          422      POP  DX
0676 CF          423      IRET              ;RETURN TO FOREGROUND
424
425      ;CHANNEL B ERROR SERVICE ROUTINE
426
0677 52          427      ERRINB: PUSH DX          ;SAVE REGISTERS
0678 50          428      PUSH AX
0679 E89400      429      CALL EDI          ;SEND EOI COMMAND TO 8274
067C BA0600      430      MOV  DX, COMMAND_PORT_CHB

```

# APPLICATIONS

```

LOC OBJ          LINE    SOURCE
067F B001        431      MOV     AL, 1          ;POINT AT RR1
0681 EE          432      OUT     DX, AL
0682 EC          433      IN      AL, DX        ;READ RR1
0683 A22B02      434      MOV     ERROR_MSG_CHB, AL ;SAVE IT IN ERROR FLAG
0686 B030        435      MOV     AL, 30H       ;SEND RESET ERROR COMMAND TO 8274
0688 EE          436      OUT     DX, AL
0689 58          437      POP     AX            ;RESTORE REGISTERS
068A 5A          438      POP     DX
068B CF          439      IRET    ;RETURN TO FOREGROUND
440
441      ;CHANNEL A TRANSMIT DATA SERVICE ROUTINE
442
068C 52          443      XNTINA: PUSH  DX        ;SAVE REGISTERS
068D 57          444      PUSH  DI
068E 50          445      PUSH  AX
068F E87E00      446      CALL  EDI            ;SEND EOI COMMAND TO 8274
0692 FF062C02    447      INC   TX_POINTER_CHA ;POINT TO NEXT CHARACTER
0696 FF0E2E02    448      DEC   TX_LENGTH_CHA ;DEC LENGTH COUNTER
069A 740E        449      JE    XIA            ;TEST IF DONE
069C BA0000      450      MOV   DX, DATA_PORT_CHA ;NOT DONE - GET NEXT CHARACTER
069F 8B3E2C02    451      MOV   DI, TX_POINTER_CHA
06A3 8A05        452      MOV   AL, [DI]       ;PUT CHARACTER IN AL
06A5 EE          453      OUT   DX, AL         ;OUTPUT IT TO 8274
06A6 58          454      POP   AX            ;RESTORE REGISTERS
06A7 5F          455      POP   DI
06A8 5A          456      POP   DX
06A9 CF          457      IRET    ;RETURN TO FOREGROUND
06AA BA0200      458      XIA:  MOV   DX, COMMAND_PORT_CHA ;ALL CHARACTERS HAVE BEEN SEND
06AD B028        459      MOV   AL, 28H       ;RESET TRANSMITTER INTERRUPT PENDING
06AF EE          460      OUT   DX, AL
06B0 C60E340201  461      MOV   TX_EMPTY_CHA, 1 ;DONE - SO SET TX EMPTY FLAG CHB
06B5 58          462      POP   AX            ;RESTORE REGISTERS
06B6 5F          463      POP   DI
06B7 5A          464      POP   DX
06B8 CF          465      IRET    ;RETURN TO FOREGROUND
466
467      ;CHANNEL A STATUS CHANGE SERVICE ROUTINE
468
06B9 52          469      STAINA: PUSH  DX        ;SAVE REGISTERS
06BA 57          470      PUSH  DI
06BB 50          471      PUSH  AX
06BC E85100      472      CALL  EDI            ;SEND EOI COMMAND TO 8274
06BF BA0200      473      MOV   DX, COMMAND_PORT_CHA
06C2 EC          474      IN    AL, DX         ;READ RR0
06C3 A23602      475      MOV   STATUS_MSG_CHA, AL ;PUT RR0 IN STATUS MESSAGE
06C6 B010        476      MOV   AL, 10H       ;SEND RESET STATUS INT COMMAND TO 8274
06C8 EE          477      OUT   DX, AL
06C9 58          478      POP   AX            ;RESTORE REGISTERS
06CA 5F          479      POP   DI
06CB 5A          480      POP   DX
06CC CF          481      IRET
482
483      ;CHANNEL A RECEIVED DATA SERVICE ROUTINE
484
06CD 52          485      RCVINA: PUSH  DX        ;SAVE REGISTERS
06CE 57          486      PUSH  DI
06CF 50          487      PUSH  AX
06D0 E83D00      488      CALL  EDI            ;SEND EOI COMMAND TO 8274
06D3 8B3E3002    489      MOV   DI, RX_POINTER_CHA ;GET RX CHA BUFFER POINTER
06D7 BA0000      490      MOV   DX, DATA_PORT_CHA

```

# APPLICATIONS

```

LOC OBJ          LINE    SOURCE
06DA EC          491      IN    AL, DX          ;READ CHARACTER
06DB 8805        492      MOV   [DI], AL      ;STORE IN BUFFER
06DD FF063002    493      INC   RX_POINTER_CHA ;BUMP THE BUFFER POINTER
06E1 FF063202    494      INC   RX_COUNT_CHA  ;BUMP THE COUNTER
06E5 3C0D        495      CMP   AL, CR_CHR    ;TEST IF LAST CHARACTER TO BE RECEIVED?
06E7 750E        496      JNE   RIA
06E9 C606350201  497      MOV   RX_READY_CHA, 1 ;YES, SET READY FLAG
06EE BA0200      498      MOV   DX, COMMAND_PORT_CHA ;POINT AT COMMAND PORT
06F1 B003        499      MOV   AL, 3         ;POINT AT WR3
06F3 EE         500      OUT   DX, AL
06F4 B0C0      501      MOV   AL, 0C0H      ;DISABLE RX
06F6 EE         502      OUT   DX, AL
06F7 58         503      RIA: POP   AX        ;EITHER WAY, RESTORE REGISTERS
06F8 5F         504      POP   DI
06F9 5A         505      POP   DX
06FA CF         506      IRET                ;RETURN TO FOREGROUND
                    507
                    508      ;CHANNEL A ERROR SERVICE ROUTINE
                    509
06FB 52         510      ERRINA: PUSH  DX      ;SAVE REGISTERS
06FC 50         511      PUSH  AX
06FD E81000      512      CALL  EOI            ;SEND EOI COMMAND TO 8274
0700 BA0200      513      MOV   DX, COMMAND_PORT_CHA
0703 8001        514      MOV   AL, 1         ;POINT AT RR1
0705 EE         515      OUT   DX, AL
0706 EC         516      IN    AL, DX        ;READ RR1
0707 A23702      517      MOV   ERROR_MSG_CHA, AL ;SAVE IT IN ERROR FLAG
070A B030      518      MOV   AL, 30H       ;SEND RESET ERROR COMMAND TO 8274
070C EE         519      OUT   DX, AL
070D 58         520      POP   AX            ;RESTORE REGISTERS
070E 5A         521      POP   DX
070F CF         522      IRET                ;RETURN TO FOREGROUND
                    523
                    524      ;END-OF-INTERRUPT ROUTINE - SENDS EOI COMMAND TO 8274.
                    525      ; THIS COMMAND MUST ALWAYS TO ISSUED ON CHANNEL A.
                    526
0710 50         527      EOI:  PUSH  AX      ;SAVE REGISTERS
0711 52         528      PUSH  DX
0712 BA0200      529      MOV   DX, COMMAND_PORT_CHA ;ALWAYS FOR CHANNEL A !!!
0715 B038      530      MOV   AL, 38H
0717 EE         531      OUT   DX, AL
0718 5A         532      POP   DX
0719 58         533      POP   AX
071A C3         534      RET
                    535
                    536      ;END OF CODE ROUTINE
                    537
----          538      ABC    ENDS
                    539      END

```

ASSEMBLY COMPLETE, NO ERRORS FOUND

# APPLICATIONS

---

## References

1. 8274 Multi-Protocol Serial Controller (MPSC) Data Sheet, Intel Corporation, California, 1980.
2. Basics of Data Communication, Electronics Book Series, McGraw-Hill, New York, 1976.
3. Telecommunications and the Computer, J. Martin, Prentice-Hall, New Jersey, 1976.
4. Technical Aspects of Data Communications, J. McNamara, DEC Press, Massachusetts, 1977.
5. Miscellaneous Data Communications Standards—EIA RS-232-C, EIA RS-422, EIA RS-423, EIA Standard Sales, Washington, D.C.

# Using the 8292 GPIB Controller

## Contents

<b>INTRODUCTION</b>	2-357
<b>GPIB/IEEE 488 OVERVIEW</b>	2-357
<b>HARDWARE ASPECTS OF THE SYSTEM</b>	2-362
8291 Talker/Listener	
8292 Controller	
8293 Bus Transceivers	
ZT7488/18 GPIB Controller	
<b>8292 COMMAND DESCRIPTION</b>	2-367
<b>SOFTWARE DRIVER OUTLINE</b>	2-371
Initialization	
Talker/Listener	
Send Data	
Receive Data	
Transfer Data	
Controller	
Trigger	
Device Clear	
Serial Poll	
Parallel Poll	
Pass Control	
Receive Control	
Service Request	
System Controller	
Remote	
Local	
Interface Clear/Abort	
<b>INTERRUPT AND DMA CONSIDERATIONS</b>	2-383
<b>APPLICATION EXAMPLE</b>	2-384
<b>CONCLUSION</b>	2-385
<b>APPENDIX A</b>	2-385
Source Listings	
<b>APPENDIX B</b>	2-403
Test Cases for the Software Drivers	
<b>APPENDIX C</b>	2-407
Remote Message Coding	



## INTRODUCTION

The Intel® 8292 is a preprogrammed UPI™-41A that implements the Controller function of the IEEE Std 488-1978 (GPIB, HP-IB, IEC Bus, etc.). In order to function the 8292 must be used with the 8291 Talker/Listener and suitable interface and transceiver logic such as a pair of Intel 8293s. In this configuration the system has the potential to be a complete GPIB Controller when driven by the appropriate software. It has the following capabilities: System Controller, send IFC and Take Charge, send REN, Respond to SRQ, send Interface messages, Receive Control, Pass Control, Parallel Poll and Take Control Synchronously.

This application note will explain the 8292 only in the system context of an 8292, 8291, two 8293s and the driver software. If the reader wishes to learn more about the UPI-41A aspects of the 8292, Intel's Application Note AP-41 describes the hardware features and programming characteristics of the device. Additional information on the 8291 may be obtained in the data sheet. The 8293 is detailed in its data sheet. Both chips will be covered here in the details that relate to the GPIB controller.

The next section of this application note presents an overview of the GPIB in a tutorial, but comprehensive nature. The knowledgeable reader may wish to skip this section; however, certain basic semantic concepts introduced there will be used throughout this note.

Additional sections cover the view of the 8292 from the CPU's data bus, the interaction of the 3 chip types (8291, 8292, 8293), the 8292's software protocol and the system level hardware/software protocol. A brief description of interrupts and DMA will be followed by an application example. Appendix A contains the source code for the system driver software.

## GPIB/IEEE 488 OVERVIEW

### DESIGN OBJECTIVES

#### What is the IEEE 488 (GPIB)?

The experience of designing systems for a variety of applications in the early 1970's caused Hewlett-Packard to define a standard intercommunication mechanism which would allow them to easily assemble instrumentation systems of varying degrees of complexity. In a typical situation each instrument designer designed his/her own interface from scratch. Each one was inconsistent in terms of electrical levels, pin-outs on a connector, and types of connectors. Every time they built a system they had to invent new cables and new documentation just to specify the cabling and interconnection procedures.

Based on this experience, Hewlett-Packard began to define a new interconnection scheme. They went further than that, however, for they wanted to specify the typical communication protocol for systems of instruments. So in 1972, Hewlett-Packard came out with the first version of the bus which since has been modified and standardized by a committee of several manufacturers, coordinated through the IEEE, to perfect what is now known as the IEEE 488 Interface Bus (also known as the HP-IB, the GPIB and the IEC bus). While this bus specification may not be perfect, it is a good compromise of the various desires and goals of instrumentation and computer peripheral manufacturers to produce a common interconnection mechanism. It fits most instrumentation systems in use today and also fits very well the microcomputer I/O bus requirements. The basic design objectives for the GPIB were to:

1. Specify a system that is easy to use, but has all of the terminology and the definitions related to that system precisely spelled out so that everyone uses the same language when discussing the GPIB.
2. Define all of the mechanical, electrical, and functional interface requirements of a system, yet not define any of the device aspects (they are left up to the instrument designer).
3. Permit a wide range of capabilities of instruments and computer peripherals to use a system simultaneously and not degrade each other's performance.
4. Allow different manufacturers' equipment to be connected together and work together on the same bus.
5. Define a system that is good for limited distance interconnections.
6. Define a system with minimum restrictions on performance of the devices.
7. Define a bus that allows asynchronous communication with a wide range of data rates.
8. Define a low cost system that does not require extensive and elaborate interface logic for the low cost instruments, yet provides higher capability for the higher cost instruments if desired.
9. Allow systems to exist that do not need a central controller; that is, communication directly from one instrument to another is possible.

Although the GPIB was originally designed for instrumentation systems, it became obvious that most of these systems would be controlled by a calculator or computer. With this in mind several modifications were made to the original proposal before its final adoption as an international standard. Figure 1 lists the salient characteristics of the

GPIB as both an instrumentation bus and as a computer I/O bus.

- Data Rate
  - 1M bytes/s, max
  - 250k bytes/s, typ
- Multiple Devices
  - 15 devices, max (electrical limit)
  - 8 devices, typ (interrupt flexibility)
- Bus Length
  - 20 m, max
  - 2 m/device, typ
- Byte Oriented
  - 8-bit commands
  - 8-bit data
- Block Multiplexed
  - Optimum strategy on GPIB due to setup overhead for commands
- Interrupt Driven
  - Serial poll (slower devices)
  - Parallel poll (faster devices)
- Direct Memory Access
  - One DMA facility at controller serves all devices on bus
- Asynchronous
 

One talker	}	3-wire handshake
Multiple listeners		
- I/O to I/O Transfers
  - Talker and listeners need not include microcomputer/controller

**Figure 1. Major Characteristics of GPIB as Microcomputer I/O Bus**

The bus can be best understood by examining each of these characteristics from the viewpoint of a general microcomputer I/O bus.

**Data Rate** — Most microcomputer systems utilize peripherals of differing operational rates, such as floppy discs at 31k or 62k bytes/s (single or double density), tape cassettes at 5k to 10k bytes/s, and cartridge tapes at 40k to 80k bytes/s. In general, the only devices that need high speed I/O are 0.5" (1.3-cm) magnetic tapes and hard discs, operational at 30k to 781k bytes/s, respectively. Certainly, the 250k-bytes/s data rate that can be easily achieved by the IEEE 488 bus is sufficient for microcomputers and their peripherals, and is more than needed for typical analog instruments that take only a few readings per second. The 1M-byte/s maximum data rate is not easily achieved on the GPIB and requires special attention to considerations beyond the scope of this note. Although not required, data buffering in each device will improve the overall bus per-

formance and allow utilization of more of the bus bandwidth.

**Multiple Devices** — Many microcomputer systems used as computers (not as components) service from three to seven peripherals. With the GPIB, up to 8 devices can be handled easily by 1 controller; with some slowdown in interrupt handling, up to 15 devices can work together. The limit of 8 is imposed by the number of unique parallel poll responses available; the limit of 15 is set by the electrical drive characteristics of the bus. Logically, the IEEE 488 Standard is capable of accommodating more device addresses (31 primary, each potentially with 31 secondaries).

**Bus Length** — Physically, the majority of microcomputer systems fit easily on a desk top or in a standard 19" (48-cm) rack, eliminating the need for extra long cables. The GPIB is designed typically to have 2 m of length per device, which accommodates most systems. A line printer might require greater cable lengths, but this can be handled at the lower speeds involved by using extra dummy terminations.

**Byte Oriented** — The 8-bit byte is almost universal in I/O applications; even 16-bit and 32-bit computers use byte transfers for most peripherals. The 8-bit byte matches the ASCII code for characters and is an integral submultiple of most computer word sizes. The GPIB has an 8-bit wide data path that may be used to transfer ASCII or binary data, as well as the necessary status and control bytes.

**Block Multiplexed** — Many peripherals are block oriented or are used in a block mode. Bytes are transferred in a fixed or variable length group; then there is a wait before another group is sent to that device, e.g., one sector of a floppy disc, one line on a printer or tape punch, etc. The GPIB is, by nature, a block multiplexed bus due to the overhead involved in addressing various devices to talk and listen. This overhead is less bothersome if it only occurs once for a large number of data bytes (once per block). This mode of operation matches the needs of microcomputers and most of their peripherals. Because of block multiplexing, the bus works best with buffered memory devices.

**Interrupt Driven** — Many types of interrupt systems exist, ranging from complex, fast, vectored/priority networks to simple polling schemes. The main tradeoff is usually cost versus speed of response. The GPIB has two interrupt protocols to help span the range of applications. The first is a single service request (SRQ) line that may be asserted by all interrupting devices. The controller then polls all devices to find out which wants service. The polling mechanism is well defined and can be easily

automated. For higher performance, the parallel poll capability in the IEEE 488 allows up to eight devices to be polled at once — each device is assigned to one bit of the data bus. This mechanism provides fast recognition of an interrupting device. A drawback is the frequent need for the controller to explicitly conduct a parallel poll, since there is no equivalent of the SRQ line for this mode.

**Direct Memory Access (DMA)** — In many applications, no immediate processing of I/O data on a byte-by-byte basis is needed or wanted. In fact, programmed transfers slow down the data transfer rate unnecessarily in these cases, and higher speed can be obtained using DMA. With the GPIB, one DMA facility at the controller serves all devices. There is no need to incorporate complex logic in each device.

**Asynchronous Transfers** — An asynchronous bus is desirable so that each device can transfer at its own rate. However, there is still a strong motivation to buffer the data at each device when used in large systems in order to speed up the aggregate data rate on the bus by allowing each device to transfer at top speed. The GPIB is asynchronous and uses a special

3-wire handshake that allows data transfers from one talker to many listeners.

**I/O To I/O Transfers** — In practice, I/O to I/O transfers are seldom done due to the need for processing data and changing formats or due to mismatched data rates. However, the GPIB can support this mode of operation where the micro-computer is neither the talker nor one of the listeners.

## GPIB SIGNAL LINES

### Data Bus

The lines DI01 through DI08 are used to transfer addresses, control information and data. The formats for addresses and control bytes are defined by the IEEE 488 standard (see Appendix C). Data formats are undefined and may be ASCII (with or without parity) or binary. DI01 is the Least Significant Bit (note that this will correspond to bit 0 on most computers).

### Management Bus

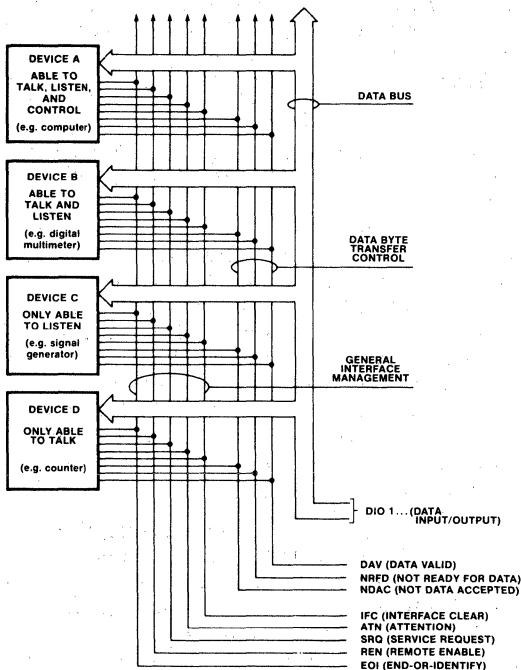
**ATN** — Attention This signal is asserted by the Controller to indicate that it is placing an address or control byte on the Data Bus. ATN is de-asserted to allow the assigned Talker to place status or data on the Data Bus. The Controller regains control by re-asserting ATN; this is normally done synchronously with the handshake to avoid confusion between control and data bytes.

**EOI** — End or Identify This signal has two uses as its name implies. A talker may assert EOI simultaneously with the last byte of data to indicate end of data. The Controller may assert EOI along with ATN to initiate a Parallel Poll. Although many devices do not use Parallel Poll, all devices *should* use EOI to end transfers (many currently available ones do not).

**SRQ** — Service Request This line is like an interrupt: it may be asserted by any device to request the Controller to take some action. The Controller must determine which device is asserting SRQ by conducting a Serial Poll at its earliest convenience. The device deasserts SRQ when polled.

**IFC** — Interface Clear This signal is asserted *only* by the System Controller in order to initialize all device interfaces to a known state. After deasserting IFC, the System Controller is the active controller of the system.

**REN** — Remote Enable This signal is asserted *only* by the System Controller. Its assertion does not place devices into Remote Control mode; REN *enables* a device to go remote when addressed to listen. When in Remote, a device should ignore its front panel controls.



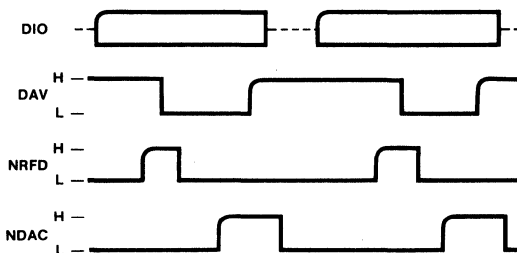
**Figure 2. Interface Capabilities and Bus Structure**

## Transfer Bus

**NRFD** — Not Ready For Data This handshake line is asserted by a listener to indicate it is not yet ready for the next data or control byte. Note that the Controller will not see NRFD deasserted (i.e., ready for data) until all devices have deasserted NRFD.

**NDAC** — Not Data Accepted This handshake line is asserted by a Listener to indicate it has not yet accepted the data or control byte on the DIO lines. Note that the Controller will not see NDAC deasserted (i.e., data accepted) until all devices have deasserted NDAC.

**DAV** — Data Valid This handshake line is asserted by the Talker to indicate that a data or control byte has been placed on the DIO lines and has had the minimum specified settling time.



**Figure 3. GPIB Handshake Sequence**

## GPIB INTERFACE FUNCTIONS

There are ten (10) interface functions specified by the IEEE 488 standard. Not all devices will have all functions and some may only have partial subsets. The ten functions are summarized below with the relevant section number from the IEEE document given at the beginning of each paragraph. For further information please see the IEEE standard.

1. **SH** — Source Handshake (section 2.3) This function provides a device with the ability to properly transfer data from a Talker to one or more Listeners using the three handshake lines.
2. **AH** — Acceptor Handshake (section 2.4) This function provides a device with the ability to properly receive data from the Talker using the three handshake lines. The AH function may also delay the beginning (NRFD) or end (NDAC) of any transfer.
3. **T** — Talker (section 2.5) This function allows a device to send status and data bytes when addressed to talk. An address consists of one (Primary) or two (Primary and Secondary)

bytes. The latter is called an extended Talker.

4. **L** — Listener (section 2.6) This function allows a device to receive data when addressed to listen. There can be extended Listeners (analogous to extended Talkers above).
5. **SR** — Service Request (section 2.7) This function allows a device to request service (interrupt) the Controller. The SRQ line may be asserted asynchronously.
6. **RL** — Remote Local (section 2.8) This function allows a device to be operated in two modes: Remote via the GPIB or Local via the manual front panel controls.
7. **PP** — Parallel Poll (section 2.9) This function allows a device to present one bit of status to the Controller-in-charge. The device need not be addressed to talk and no handshake is required.
8. **DC** — Device Clear (section 2.10) This function allows a device to be cleared (initialized) by the Controller. Note that there is a difference between DC (*device clear*) and the IFC line (*interface clear*).
9. **DT** — Device Trigger (section 2.11) This function allows a device to have its basic operation started either individually or as part of a group. This capability is often used to synchronize several instruments.
10. **C** — Controller (section 2.12) This function allows a device to send addresses, as well as universal and addressed commands to other devices. There may be more than one controller on a system, but only one may be the controller-in-charge at any one time.

At power-on time the controller that is hardwired to be the System Controller becomes the active controller-in-charge. The System Controller has several unique capabilities including the ability to send Interface Clear (IFC — clears all device interfaces and returns control to the System Controller) and to send Remote Enable (REN — allows devices to respond to bus data once they are addressed to listen). The System Controller may optionally Pass Control to another controller, if the system software has the capability to do so.

## GPIB CONNECTOR

The GPIB connector is a standard 24-pin industrial connector such as Cinch or Amphenol series 57 Micro-Ribbon. The IEEE standard specifies this connector, as well as the signal connections and the mounting hardware.

The cable has 16 signal lines and 8 ground lines. The maximum length is 20 meters with no more than two meters per device.

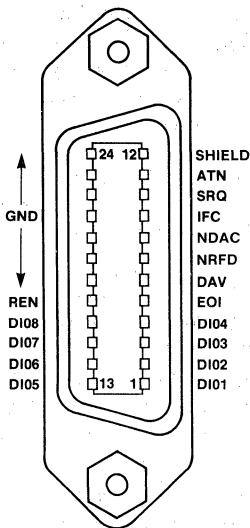


Figure 4. GPIB Connector

**GPIB SIGNAL LEVELS**

The GPIB signals are all TTL compatible, low true signals. A signal is asserted (true) when its electrical voltage is less than 0.5 volts and is deasserted (false) when it is greater than 2.4 volts. Be careful not to become confused with the two handshake signals, NRFD and NDAC which are also low true (i.e. > 0.5 volts implies the device is Not Ready For Data).

The Intel 8293 GPIB transceiver chips ensure that all relevant bus driver/receiver specifications are met. Detailed bus electrical specifications may be found in Section 3 of the IEEE Std 488-1978. The Standard is the ultimate reference for all GPIB questions.

**GPIB MESSAGE PROTOCOLS**

The GPIB is a very flexible communications medium and as such has many possible variations of protocols. To bring some order to the situation, this section will discuss a protocol similar to the one used by Ziatech's ZT80 GPIB controller for Intel's MULTIBUS™ computers. The ZT80 is a complete high-level interface processor that executes a set of high level instructions that map directly into GPIB actions. The sequences of commands, addresses and data for these instructions provide a good example of how to use the GPIB (additional information is available in the ZT80 Instruction Manual). The 'null' at the end of each instruction is for cosmetic use to remove previous information from the DIO lines.

*DATA* — Transfer a block of data from device A to devices B, C...

1. Device A Primary (Talk) Address  
Device A Secondary Address (if any)
2. Universal Unlisten
3. Device B Primary (Listen) Address  
Device B Secondary Address (if any)  
Device C Primary (Listen) Address  
etc.
4. First Data Byte  
Second Data Byte

Last Data Byte (EOI)

5. Null

*TRIGR* — Trigger devices A, B,... to take action

1. Universal Unlisten
2. Device A Primary (Listen) Address  
Device A Secondary Address (if any)  
Device B Primary (Listen) Address  
Device B Secondary Address (if any)  
etc.

3. Group Execute Trigger
4. Null

*PSCTL* — Pass control to device A

1. Device A Primary (Talk) Address  
Device A Secondary Address (if any)
2. Take Control
3. Null

*CLEAR* — Clear all devices

1. Device Clear
2. Null

*REMAI* — Remote Enable

1. Assert REN continuously

*GOREM* — Put devices A, B,... into Remote

1. Assert REN continuously
2. Device A Primary (Listen) Address  
Device A Secondary Address (if any)  
Device B Primary (Listen) Address  
Device B Secondary Address (if any)  
etc.
3. Null

*GOLOC* — Put devices A, B,... into Local

1. Device A Primary (Listen) Address  
Device A Secondary Address (if any)  
Device B Primary (Listen) Address  
Device B Secondary Address (if any)  
etc.
2. Go To Local
3. Null

*LOCAL* — Reset all devices to Local

1. Stop asserting REN

*LLKAL* — Prevent all devices from returning to Local

1. Local Lock Out
2. Null

*SPOLL* — Conduct a serial poll of devices A, B,...

1. Serial Poll Enable
2. Universal Unlisten
3. ZT 80 Primary (Listen) Address  
ZT 80 Secondary Address
4. Device Primary (Talk) Address  
Device Secondary Address (if any)
5. Status byte from device
6. Go to Step 4 until all devices on list have been polled
7. Serial Poll Disable
8. Null

*PPUAL* — Unconfigure and disable Parallel Poll response from all devices

1. Parallel Poll Unconfigure
2. Null

*ENAPP* — Enable Parallel Poll response in devices A, B,...

1. Universal Unlisten
2. Device Primary (Listen) Address  
Device Secondary Address (if any)
3. Parallel Poll Configure
4. Parallel Poll Enable
5. Go to Step 2 until all devices on list have been configured.
6. Null

*DISPP* — Disable Parallel Poll response from devices A, B,...

1. Universal Unlisten
2. Device A Primary (Listen) Address  
Device A Secondary Address (if any)  
Device B Primary (Listen) Address  
Device B Secondary Address (if any)  
etc.
3. Disable Parallel Poll
4. Null

This Ap Note will detail how to implement a useful subset of these controller instructions.

## HARDWARE ASPECTS OF THE SYSTEM

### 8291 GPIB TALKER/LISTENER

The 8291 is a custom designed chip that implements many of the non-controller GPIB functions. It provides hooks so the user's software can implement additional features to complete the set. This chip is discussed in detail in its data sheet. The major features are summarized here:

- Designed to interface microprocessors to the GPIB
- Complete Source and Acceptor Handshake
- Complete Talker and Listener Functions with extended addressing

- Service Request, Parallel Poll, Device Clear, Device Trigger, Remote/Local functions
- Programmable data transfer rate
- Maskable interrupts
- On-chip primary and secondary address recognition
- 1-8 MHz clock range
- 16 registers (8 read, 8 write) for CPU interface
- DMA handshake provision
- Trigger output pin
- On-chip EOS (End of Sequence) recognition

The pinouts and block diagram are shown in Fig. 5. One of eight read registers is for data transfer to the CPU; the other seven allow the microprocessor to monitor the GPIB states and various bus and device conditions. One of the eight write registers is for data transfer from the CPU; the other seven control various features of the 8291.

The 8291 interface functions will be software configured in this application example to the following subsets for use with the 8292 as a controller that does not pass control. The 8291 is used only to provide the handshake logic and to send and receive data bytes. It is not acting as a normal device in this mode, as it never sees ATN asserted.

SH1	Source Handshake
AH1	Acceptor Handshake
T3	Basic Talk-only
L1	Basic Listen-only
SR0	No Service Requests
RL0	No Remote/Local
PP0	No Parallel Poll response
DC0	No Device Clear
DT0	No Device Trigger

If control is passed to another controller, the 8291 must be reconfigured to act as a talker/listener with the following subsets:

SH1	Source Handshake
AH1	Acceptor Handshake
T5	Basic Talker and Serial Poll
L3	Basic Listener
SR1	Service Requests
RL1	Remote/Local with Lockout
PP2	Preconfigured Parallel Poll
DC1	Device Clear
DT1	Device Trigger
C0	Not a Controller

Most applications do not pass control and the controller is always the system controller (see 8292 commands below).

### 8292 GPIB CONTROLLER

The 8292 is a preprogrammed Intel® 8041A that provides the additional functions necessary to

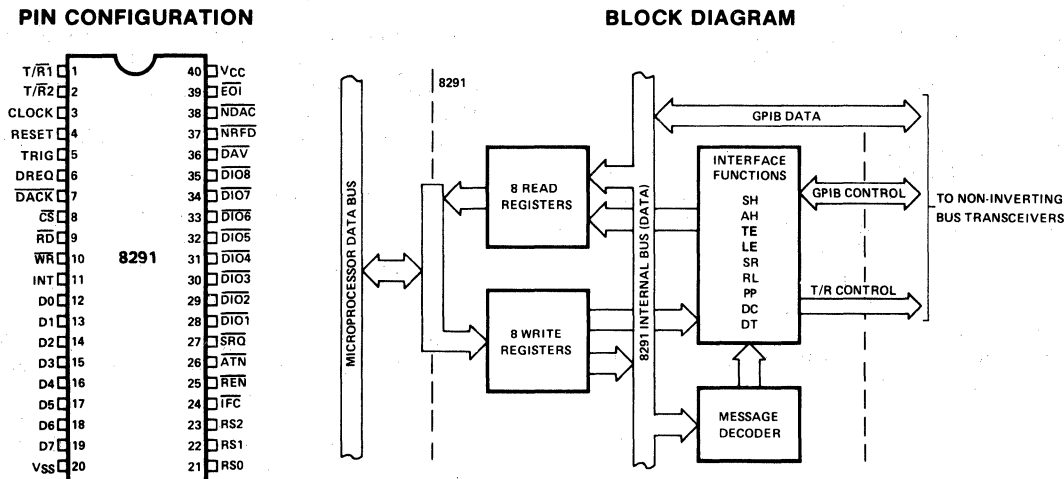


Figure 5. 8291 Pin Configuration and Block Diagram

implement a GPIB controller when used with an 8291 Talker/Listener. The 8041A is documented in both a user's manual and in AP-41. The following description will serve only as an outline to guide the later discussion.

The 8292 acts as an intelligent slave processor to the main system CPU. It contains a processor, memory, I/O and is programmed to perform a variety of tasks associated with GPIB controller operation. The on-chip RAM is used to store information about the state of the Controller function, as well as a variety of local variables, the stack and certain user status information. The timer/counter may be optionally used for several time-out functions or for counting data bytes transferred. The I/O ports provide the GPIB control signals, as well as the ancillary lines necessary to make the 8291, 2, 3 work together.

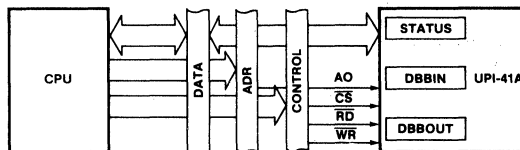
The 8292 is closely coupled to the main CPU through three on-chip registers that may be independently accessed by both the master and the 8292 (UPI-41A). Figure 6 shows this Register Interface. Also refer to Figure 12.

The status register is used to pass Interrupt Status information to the master CPU (A0 = 1 on a read).

The DBBOUT register is used to pass one of five other status words to the master based on the last command written into DBBIN. DBBOUT is accessed when A0 = 0 on a Read. The five status words are Error Flag, Controller Status, GPIB Status, Event Counter Status or Time Out Status.

DBBIN receives either commands (A0 = 1 on a Write) or command related data (A0 = 0 on a write) from the master. These command related data are

Interrupt Mask, Error Mask, Event Counter or Time Out.



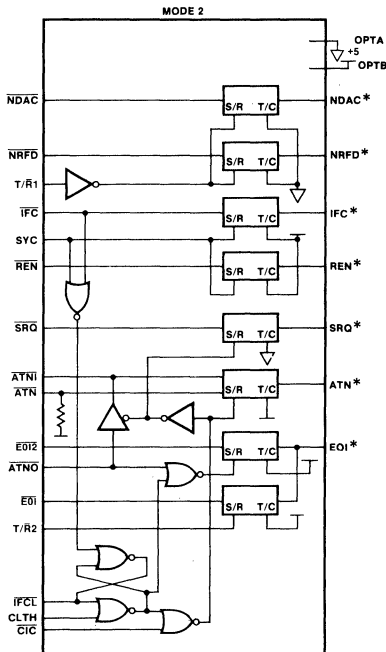
CS	AO	RD	WR	REGISTER
0	0	0	1	READ DBBOUT
0	1	0	1	READ STATUS
0	0	1	0	WRITE DBBIN (DATA)
0	1	1	0	WRITE DBBIN (COMMAND)
1	X	X	X	NO ACTION

Figure 6. UPI-41A Registers

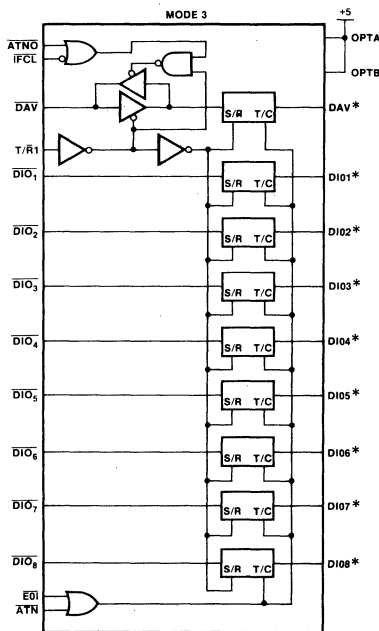
### 8293 GPIB TRANSCEIVERS

The 8293 is a multi-use HMOS chip that implements the IEEE 488 bus transceivers and contains the additional logic required to make the 8291 and 8292 work together. The two option strapping pins are used to internally configure the chip to perform the specialized gating required for use with 8291 as a device or with 8291/92 as a controller.

In this application example the two configurations used are shown in Fig. 7a and 7b. The drivers are set to open collector or three state mode as required and the special logic is enabled as required in the two modes.



**Figure 7a. 8293 Mode 2**



**Figure 7b. 8293 Mode 3**

## 8291/2/3 CHIP SET

Figure 8 shows the four chips interconnected with the special logic explicitly shown.

The 8291 acts only as the mechanism to put commands and addresses on the bus while the 8292 is asserting ATN. The 8291 is tricked into believing that the ATN line is not asserted by the ATN2 output of the ATN transceiver and is placed in Talk-only mode by the CPU. The 8291 then acts as though it is sending data, when in reality it is sending addresses and/or commands. When the 8292 deasserts ATN, the CPU software must place the 8291 in Talk-only, Listen-only or Idle based on the implicit knowledge of how the controller is going to participate in the data transfer. In other words, the 8291 does not respond directly to addresses or commands that it sends on the bus on behalf of the Controller. The user software, through the use of Listen-only or Talk-only, makes the 8291 behave as though it were addressed.

Although it is not a common occurrence, the GPIB specification allows the Controller to set up a data transfer between two devices and not directly participate in the exchange. The controller must know when to go active again and regain control. The chip set accomplishes this through use of the "Continuous Acceptor Handshake cycling mode" and the ability to detect EO1 or EOS at the end of the transfer. See XFER in the Software Driver Outline below.

If the 8292 is not the System Controller as determined by the signal on its SYNC pin, then it must be able to respond to an IFC within 100 usec. This is accomplished by the cross-coupled NORs in Fig. 7a which deassert the 8293's internal version of  $\overline{CIC}$  (Not Controller-in-Charge). This condition is latched until the 8292's firmware has received the IFCL (interface clear received latch) signal by testing the IFCL input. The firmware then sets its signals to reflect the inactive condition and clears the 8293's latch.

In order for the 8292 to conduct a Parallel Poll the 8291 must be able to capture the PP response on the DIO lines. The only way to do this is to fool the 8291 by putting it into Listen-only mode and generating a DAV condition. However, the bus spec does not allow a DAV during Parallel Poll, so the back-to-back 3-state buffers (see Fig. 7b) in the 8293 isolate the bus and allow the 8292 to generate a local DAV for this purpose. Note that the 8291 cannot assert a Parallel Poll response. When the 8292 is not the controller-in-charge the 8291 may respond to PPs and the 8293 guarantees that the DIO drivers are in "open collector" mode through the OR gate (Fig. 7b).



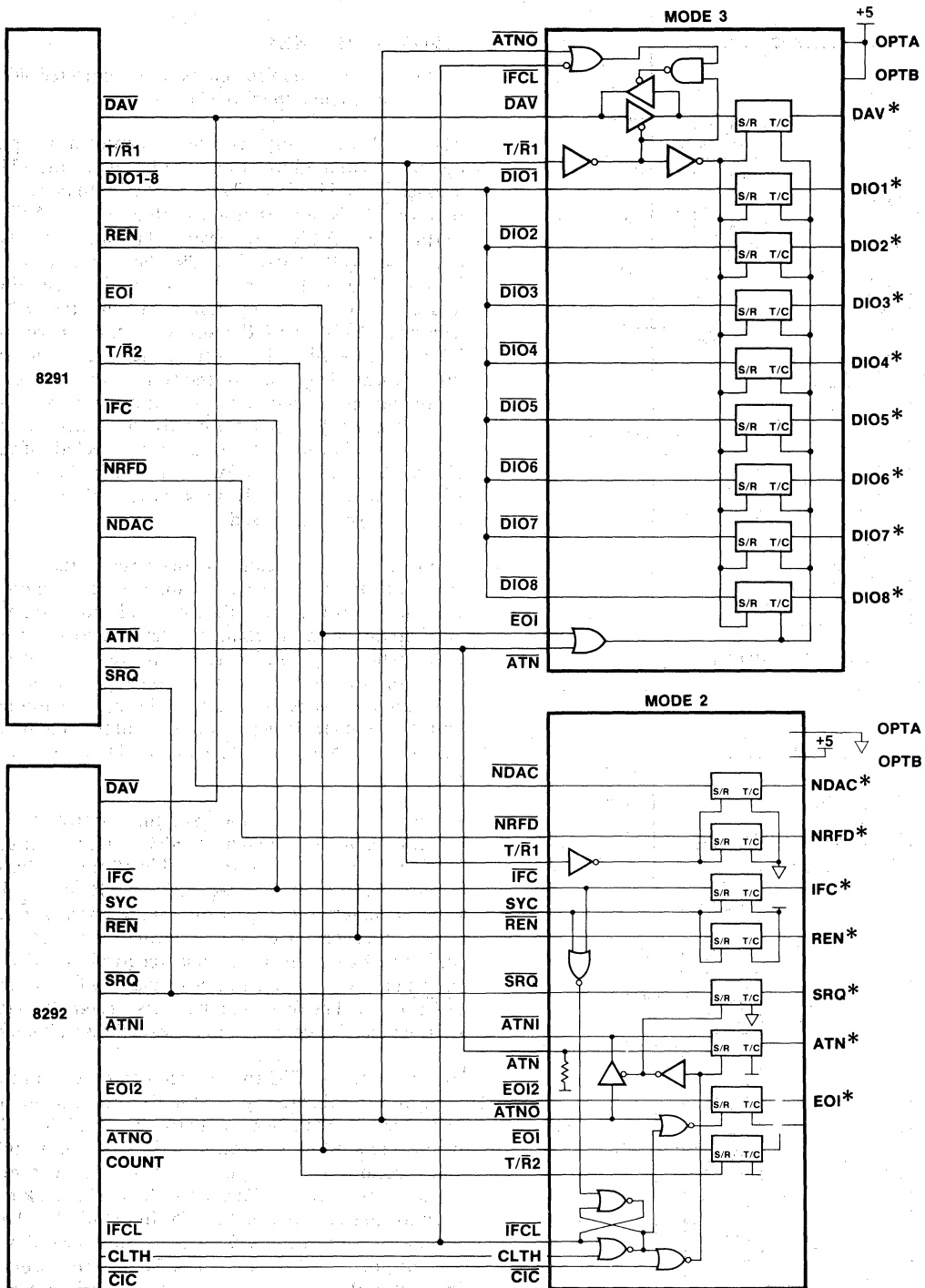


Figure 8. Talker/Listener/Controller

# APPLICATIONS

## ZT7488/18 GPIB CONTROLLER

Ziatech's GPIB Controller, the ZT7488/18 will be used as the controller hardware in this Application Note. The controller consists of an 8291, 8292, an 8 bit input port and TTL logic equivalent to that shown in Figure 8. Figure 9 shows the card's block diagram. The ZT7488/18 plugs into the STD bus, a 56 pin 8 bit microprocessor oriented bus. An 8085 CPU card is also available on the STD bus and will be used to execute the driver software.

The 8291 uses I/O Ports 60H to 67H and the 8292 uses I/O Ports 68H and 69H. The five interrupt lines are connected to a three-state buffer at I/O Port

6FH to facilitate polling operation. This is required for the TCI, as it cannot be read internally in the 8292. The other three 8292 lines (SPI, IBF, OBF) and the 8291's INT line are also connected to minimize the number of I/O reads necessary to poll the devices.

$\overline{NDAC}$  is connected to  $\overline{COUNT}$  on the 8292 to allow byte counting on data transfers. The example driver software will not use this feature, as the software is simpler and faster if an internal 8085 register is used for counting in software.

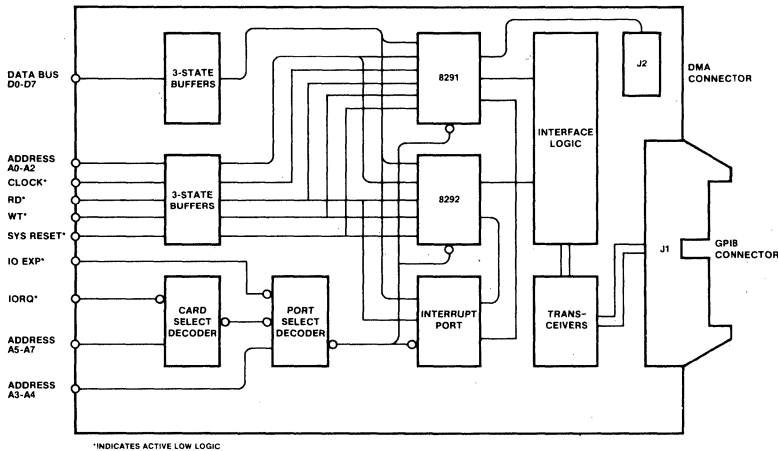


Figure 9. ZT7488/18 GPIB Controller

READ REGISTERS								PORT =	WRITE REGISTERS							
D17	D16	D15	D14	D13	D12	D11	D10	60H	DO7	DO6	DO5	DO4	DO3	DO2	DO1	DO0
DATA IN									DATA OUT							
CPT	APT	GET	END	DEC	ERR	BO	BI	61H	CPT	APT	GET	END	DEC	ERR	BO	BI
INTERRUPT STATUS 1									INTERRUPT MASK 1							
INT	SPAS	LL0	REM	SPASC	LLOC	REMC	ADSC	62H	0	0	DMA0	DMA1	SPASC	LLOC	REMC	ADSC
INTERRUPT STATUS 2									INTERRUPT MASK 2							
S8	SRQS	S6	S5	S4	S3	S2	S1	63H	S8	rsv	S6	S5	S4	S3	S2	S1
SERIAL POLL STATUS									SERIAL POLL MODE							
Ion	Ion	EQI	LPAS	TPAS	LA	TA	MJMN	64H	TO	LO	0	0	0	0	ADM1	ADM0
ADDRESS STATUS									ADDRESS MODE							
CPT7	CPT6	CPT5	CPT4	CPT3	CPT2	CPT1	CPT0	65H	CNT2	CNT1	CNT0	COM4	COM3	COM2	COM1	COM0
COMMAND PASS THROUGH									AUX MODE							
X	DT0	DL0	AD5:0	AD4:0	AD3:0	AD2:0	AD1:0	66H	ARS	DT	DL	AD5	AD4	AD3	AD2	AD1
ADDRESS 0									ADDRESS 0:1							
X	DT1	DL1	AD5:1	AD4:1	AD3:1	AD2:1	AD1:1	67H	EC7	EC6	EC5	EC4	EC3	EC2	EC1	EC0
ADDRESS 1									EOS							

Figure 10. 8291 Registers

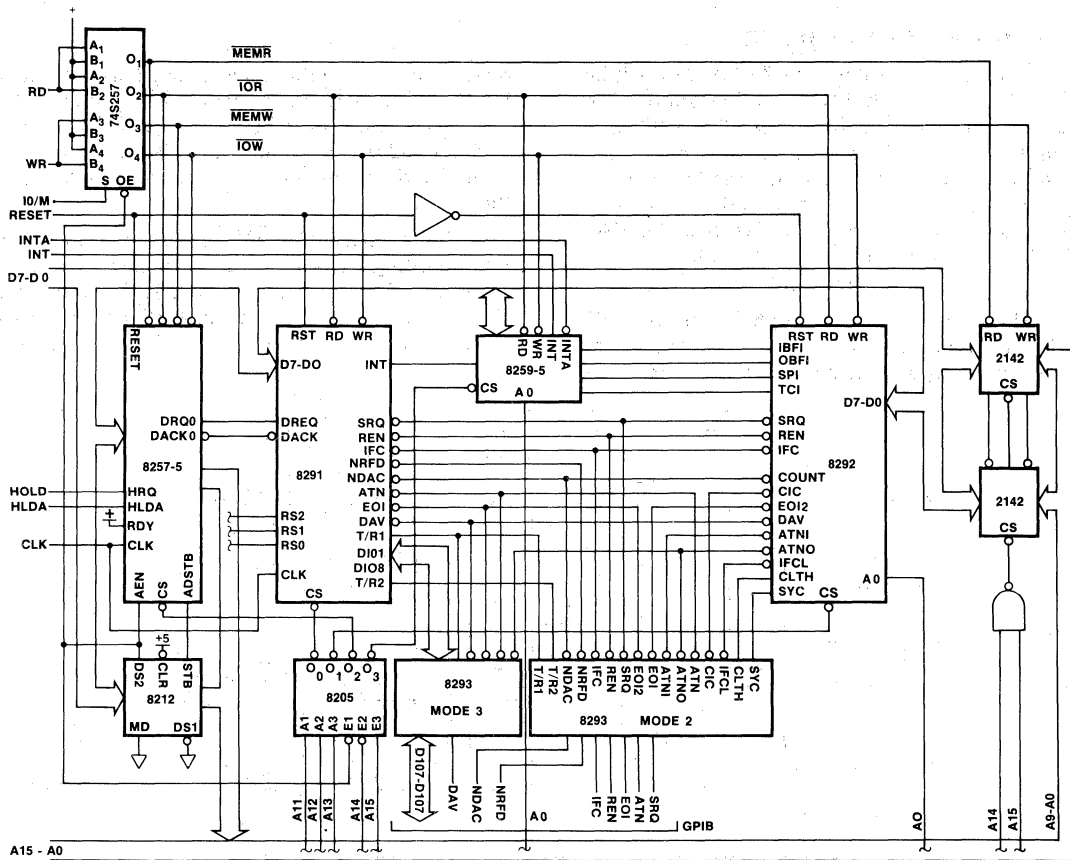


Figure 11. DMA/Interrupt GPIB Controller Block Diagram

The application example will not use DMA or interrupts; however, the Figure 11 block diagram includes these features for completeness.

The 8257-5 DMA chip can be used to transfer data between the RAM and the 8291 Talker/Listener. This mode allows a faster data rate on the GPIB and typically will depend on the 8291's EOS or EO1 detection to terminate the transfer. The 8259-5 interrupt controller is used to vector the five possible interrupts for rapid software handling of the various conditions.

### 8292 COMMAND DESCRIPTION

This section discusses each command in detail and relates them to a particular GPIB activity. Recall that although the 8041A has only two read registers and one write register, through the magic of on-chip firmware the 8292 appears to have six read registers and five write registers. These are listed in Figure 12. Please see the 8292 data sheet for detailed definitions

of each register. Note the two letter mnemonics to be used in later discussions. The CPU must not write into the 8292 while IBF (Input Buffer Full) is a one, as information will be lost.

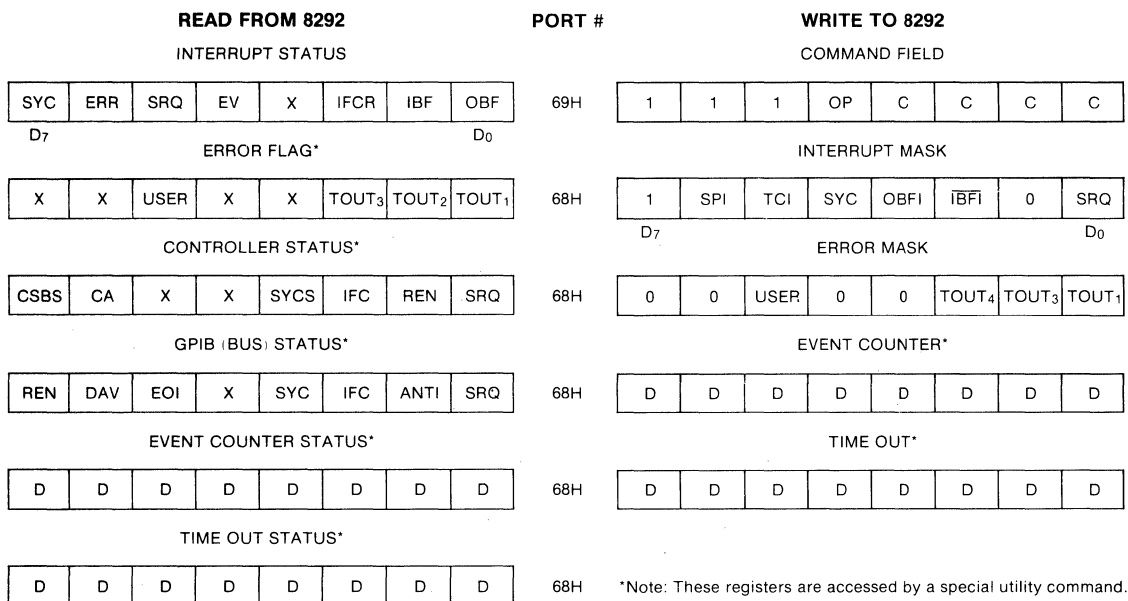
### DIRECT COMMANDS

Both the Interrupt Mask (IM) and the Error Mask (EM) register may be directly written with the LSB of the address bus (A0) a "0". The firmware uses the MSB of the data written to differentiate between IM and EM.

### Load Interrupt Mask

This command loads the Interrupt Mask with D7-D0. Note that D7 must be a "1" and that interrupts are enabled by a corresponding "1" bit in this register. IFC interrupt cannot be masked off; however, when the 8292 is the System Controller, sending an ABORT command will not cause an IFC interrupt.

# APPLICATIONS



**Figure 12. 8292 Registers**

## Load Error Mask

This command loads the Error Mask with D7–D0. Note that D7 must be a zero and that interrupts are enabled by a corresponding “1” bit in this register.

## UTILITY COMMANDS

These commands are used to read or write the 8292 registers that are not directly accessible. All utility commands are written with A0 = 1, D7 = D6 = D5 = 1, D4 = 0. D3–D0 specify the particular command. For writing into registers the general sequence is:

1. wait for IBF = 0 in Interrupt Status Register
2. write the appropriate command to the 8292,
3. write the desired register value to the 8292 with A0 = 1 with no other writes intervening,
4. wait for indication of completion from 8292 (IBF = 0).

For reading a register the general sequence is:

1. wait for IBF = 0 in Interrupt Status Register
2. write the appropriate command to the 8292
3. wait for a TCI (Task Complete Interrupt)
4. Read the value of the accessed register from the 8292 with A0 = 0.

**WEVC** — Write to Event Counter  
(Command = 0E2H)

The byte written following this command will be loaded into the event counter register and event counter status for byte counting. The internal

counter is incremented on a high to low transition of the COUNT (T1) input. In this application example NDAC is connected to count. The counter is an 8 bit register and therefore can count up to 256 bytes (writing 0 to the EC implies a count of 256). If longer blocks are desired, the main CPU must handle the interrupts every 256 counts and carefully observe the timing constraints.

Because the counter has a frequency range from 0 to 133 kHz when using a 6 MHz crystal, this feature may not be usable with all devices on the GPIB. The 8291 can easily transfer data at rates up to 250 kHz and even faster with some tuning of the system. There is also a 500 ns minimum high time requirement for COUNT which can potentially be violated by the 8291 in continuous acceptor handshake mode (i.e., TNDDV1 + TDVND2–C = 350 + 350 = 700 max). When cable delays are taken into consideration, this problem will probably never occur.

When the 8292 has completed the command, IBF will become a “0” and will cause an interrupt if masked on.

**WTOUT** — Write to Time Out Register  
(Command = 0E1H)

The byte written following this command will be used to determine the number of increments used for the time out functions. Because the register is 8 bits, the maximum time out is 256 time increments. This

is probably enough for most instruments on the GPIB but is not enough for a manually stepped operation using a GPIB logic analyzer like Ziotech's ZT488. Also, the 488 Standard does not set a lower limit on how long a device may take to do each action. Therefore, any use of a time out must be able to be overridden (this is a good general design rule for service and debugging considerations).

The time out function is implemented in the 8292's firmware and will not be an accurate time. The counter counts backwards to zero from its initial value. The function may be enabled/disabled by a bit in the Error mask register. When the command is complete IBF will be set to a "0" and will cause an interrupt if masked on.

**REVC** — Read Event Counter Status  
(Command = 0E3H)

This command transfers the content of the Event Counter to the DBBOUT register. The firmware then sets TCI = 1 and will cause an interrupt if masked on. The CPU may then read the value from the 8292 with A0 = 0.

**RINM** — Read Interrupt Mask Register  
(Command = 0E5H)

This command transfers the content of the Interrupt Mask register to the DBBOUT register. The firmware sets TCI = 1 and will cause an interrupt if masked on. The CPU may then read the value.

**RERM** — Read Error Mask Register  
(Command = 0EAH)

This command transfers the content of the Error Mask register to the DBBOUT register. The firmware sets TCI = 1 and will cause an interrupt if masked on. The CPU may then read the value.

**RCST** — Read Controller Status Register  
(Command = 0E6H)

This command transfers the content of the Controller Status register to the DBBOUT register. The firmware sets TCI = 1 and will cause an interrupt if masked on. The CPU may then read the value.

**RTOUT** — Read Time Out Status Register  
(Command = 0E9H)

This command transfers the content of the Time Out Status register to the DBBOUT register. The firmware sets TCI = 1 and will cause an interrupt if masked on. The CPU may then read the value.

If this register is read while a time-out function is in process, the value will be the time remaining before time-out occurs. If it is read after a time-out, it will be zero. If it is read when no time-out is in process, it will be the last value reached when the previous timing occurred.

**RBST** — Read Bus Status Register  
(Command = 0E7H)

This command causes the firmware to read the GPIB management lines, DAV and the SYC pin and place a copy in DBBOUT. TCI is set to "1" and will cause an interrupt if masked on. The CPU may read the value.

**RERF** — Read Error Flag Register  
(Command = 0E4H)

This command transfers the content of the Error Flag register to the DBBOUT register. The firmware sets TCI = 1 and will cause an interrupt if masked on. The CPU may then read the value.

This register is also placed in DBBOUT by an IACK command if ERR remains set. TCI is set to "1" in this case also.

**IACK** — Interrupt Acknowledge  
(Command = A1 A2 A3 A4 1 A5 1 1)

This command is used to acknowledge any combinations of the five SPI interrupts (A1–A5): SYC, ERR, SRQ, EV, and IFCR. Each bit A1–A5 is an individual acknowledgement to the corresponding bit in the Interrupt Status Register. The command clears SPI but it will be set again if all of the pending interrupts were not acknowledged.

If A2 (ERR) is "1", the Error Flag register is placed in DBBOUT and TCI is set. The CPU may then read the Error Flag without issuing an RERF command.

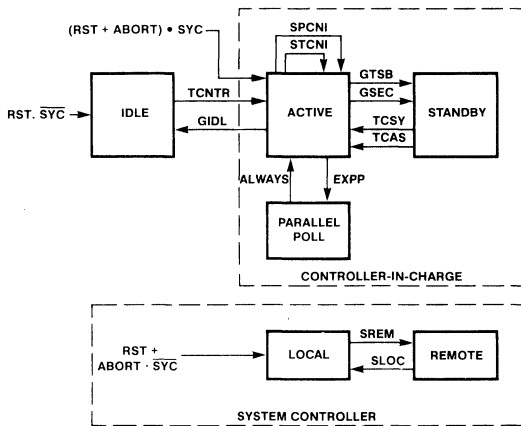
## OPERATION COMMANDS

The following diagram (Fig. 13) is an attempt to show the interrelationships among the various 8292 Operation Commands. It is not meant to replace the complete controller state diagram in the IEEE Standard.

**RST** — Reset (Command = 0F2H)

This command has the same effect as an external reset applied to the chip's pin #4. The 8292's actions are:

1. All outputs go to their electrical high state. This means that SPI, TCI, OBF1, IBF1, CLTH will be TRUE and all other GPIB signals will be FALSE.
2. The 8292's firmware will cause the above mentioned five signals to go FALSE after approximately 17.5 usec. (at 6 MHz).
3. These registers will be cleared: Interrupt Status, Interrupt Mask, Error Mask, Time Out, Event Counter, Error Flag.
4. If the 8292 is the System Controller (SYC is TRUE), then IFC will be sent TRUE for approximately 100 usec and the Controller function will end up in charge of the bus. If the 8292 is not the



**Figure 13. 8292 Command Flowchart**

System Controller then it will end up in an Idle state.

5. TCI will not be set.

**RSTI** — Reset Interrupts (Command = 0F3)

This command clears all pending interrupts and error flags. The 8292 will stop waiting for actions to occur (e.g., waiting for ATN to go FALSE in a TCNTR command or waiting for the proper handshake state in a TCSY command). TCI will not be set.

**ABORT** — Abort all operations and Clear Interface (Command = 0F9H)

If the 8292 is not the System Controller this command acts like a NOP and flags a USER ERROR in the Error Flag Register. No TCI will occur.

If the 8292 is the System Controller then IFC is set TRUE for approximately 100  $\mu$ sec and the 8292 becomes the Controller-in-Charge and asserts ATN. TCI will be set, only if the 8292 was NOT the CIC.

**STCNI** — Start Counter Interrupts (Command = 0FEH)

Enables the EV Counter Interrupt. TCI will not be set. Note that the counter must be enabled by a GSEC command.

**SPCNI** — Stop Counter Interrupts (Command = 0F0H)

The 8292 will not generate an EV interrupt when the counter reaches 0. Note that the counter will continue counting. TCI will not be set.

**SREM** — Set Interface to Remote Control (Command = 0F8H)

If the 8292 is the System Controller, it will set REN

and TCI TRUE. Otherwise it only sets the User Error Flag.

**SLOC** — Set Interface to Local Mode (Command = 0F7H)

If the 8292 is the System Controller, it will set REN FALSE and TCI TRUE. Otherwise, it only sets the User Error Flag.

**EXPP** — Execute Parallel Poll (Command = 0F5H)

If not Controller-in-Charge, the 8292 will treat this as a NOP and does not set TCI. If it is the Controller-in-Charge then it sets IDY (EOI & ATN) TRUE and generates a local DAV pulse (that never reaches the GPIB because of gates in the 8293). If the 8291 is configured as a listener, it will capture the Parallel Poll Response byte in its data register. TCI is not generated, the CPU must detect the BI (Byte In) from the 8291. The 8292 will be ready to accept another command before the BI occurs; therefore the 8291's BI serves as a task complete indication.

**GTSB** — Go To Standby (Command = 0F6H)

If the 8292 is not the Controller-in-Charge, it will treat this command as a NOP and does not set TCI TRUE. Otherwise, it goes to Controller Standby State (CSBS), sets ATN FALSE and TCI TRUE. This command is used as part of the Send, Receive, Transfer and Serial Poll System commands (see next section) to allow the addressed talker to send data/status.

If the data transfer does not start within the specified Time-Out, the 8292 sets TOUT2 TRUE in the Error Flag Register and sets SPI (if enabled). The controller continues waiting for a new command. The CPU must decide to wait longer or to regain control and take corrective action.

**GSEC** — Go to Standby and Enable Counting (Command = 0F4H)

This command does the same things as GTSB but also initializes the event counter to the value previously stored in the Event Counter Register (default value is 256) and enables the counter. One may wire the count input to NDAC to count bytes. When the counter reaches zero, it sets EV (and SPI if enabled) in Interrupt Status and will set EV every 256 bytes thereafter. Note that there is a potential loss of count information if the CPU does not respond to the EV/SPI before another 256 bytes have been transferred. TCI will be set at the end of the command.

**TCSY** — Take Control Synchronously (Command = 0FDH)

If the 8292 is not in Standby, it treats this command as a NOP and does not set TCI. Otherwise, it waits

for the proper handshake state and sets ATN TRUE. The 8292 will set TOUT3 if the handshake never assumes the correct state and will remain in this command until the handshake is proper or a RSTI command is issued. If the 8292 successfully takes control, it sets TCI TRUE.

This is the normal way to regain control at the end of a Send, Receive, Transfer or Serial Poll System Command. If TCSY is not successful, then the controller must try TCAS (see warning below).

**TCAS** — Take Control Asynchronously  
(Command = 0FCH)

If the 8292 is not in Standby, it treats this command as a NOP and does not set TCI. Otherwise, it arbitrarily sets ATN TRUE and TCI TRUE. Note that this action may cause devices on the bus to lose a data byte or cause them to interpret a data byte as a command byte. Both Actions can result in anomalous behavior. TCAS should be used only in emergencies. If TCAS fails, then the System Controller will have to issue an ABORT to clean things up.

**GIDL** — Go to Idle (Command = 0F1H)

If the 8292 is not the Controller in Charge and Active, then it treats this command as a NOP and does not set TCI. Otherwise, it sets ATN FALSE, becomes Not Controller in Charge, and sets TCI TRUE. This command is used as part of the Pass Control System Command.

**TCNTR** — Take (Receive) Control  
(Command = 0FAH)

If the 8292 is not Idle, then it treats this command as a NOP and does not set TCI. Otherwise, it waits for the current Controller-in-Charge to set ATN FALSE. If this does not occur within the specified Time Out, the 8292 sets TOUT1 in the Error Flag Register and sets SPI (if enabled). It will not proceed until ATN goes false or it receives an RSTI command. Note that the Controller in Charge must previously have sent this controller (via the 8291's command pass through register) a Pass Control message. When ATN goes FALSE, the 8292 sets CIC, ATN and TCI TRUE and becomes Active.

**SOFTWARE DRIVER OUTLINE**

The set of system commands discussed below is shown in Figure 14. These commands are implemented in software routines executed by the main CPU.

The following section assumes that the Controller is the System Controller and will not Pass Control. This is a valid assumption for 99+% of all controllers. It also assumes that no DMA or Interrupts will be used. SYC (System Control Input)

should not be changed after Power-on in any system — it adds unnecessary complexity to the CPU's software.

In order to use polling with the 8292 one must enable TCI but not connect the pin to the CPU's interrupt pin. TCI must be readable by some means. In this application example it is connected to bit 1 port 6FH on the ZT7488/18. In addition, the other three 8292 interrupt lines and the 8291 interrupt are also on that port (SPI-Bit 2, IBF1-Bit 4, OBF1-Bit 3, 8291 INT-Bit 0).

These drivers assume that only primary addresses will be used on the GPIB. To use secondary addresses, one must modify the test for valid talk/listen addresses (range macro) to include secondaries.

INIT	INITIALIZATION
<b>Talker/Listener</b>	
SEND	SEND DATA
REC V	RECEIVE DATA
XFER	TRANSFER DATA
<b>Controller</b>	
TRIG	GROUP EXECUTE TRIGGER
DCLR	DEVICE CLEAR
SPOL	SERIAL POLL
PPEN	PARALLEL POLL ENABLE
PPDS	PARALLEL POLL DISABLE
PPUN	PARALLEL POLL UNCONFIGURE
PPOL	PARALLEL POLL
PCTL	PASS CONTROL
RCTL	RECEIVE CONTROL
SRQD	SERVICE REQUESTED
<b>System Controller</b>	
REME	REMOTE ENABLE
LOCL	LOCAL
IFCL	ABORT/INTERFACE CLEAR

**Figure 14. Software Driver Routines**

**INITIALIZATION**

**8292** — Comes up in Controller Active State when SYC is TRUE. The only initialization needed is to enable the TCI interrupt mask. This is done by writing 0A0H to Port 68H.

**8291** — Disable both the major and minor addresses because the 8291 will never see the 8292's commands/addresses (refer to earlier hardware discussion). This is done by writing 60H and 0E0H to Port 66H.

# APPLICATIONS

Set Address Mode to Talk-only by writing 80H to Port 64H.

Set internal counter to 3 MHz to match the clock input coming from the 8085 by writing 23H to Port 65H. High speed mode for the handshakes will not be used here even though the hardware uses three-state drivers.

No interrupts will be enabled now. Each routine will enable the ones it needs for ease of polling operation. The INT bit may be read through Port 6FH. Clear both interrupt mask registers.

Release the chip's initialization state by writing 0 to Port 65H.

## INIT:

Enable-8292	;Set up Int. pins for Port 6FH
Enable TCI	;Task complete must be on
Enable-8291	
Disable major address	;In controller usage, the 8291
Disable minor address	;Is set to talk only and/or listen only
ton	;Talk only is our rest state
Clock frequency	;3 MHz in this ap note example
All interrupts off	
Immediate execute pon	;Releases 8291 from init. state

## TALKER/LISTENER ROUTINES

### Send Data

*SEND* <listener list pointer> <count> <EOS> <data buffer pointer>

This system command sends data from the CPU to one or more devices. The data is usually a string of ASCII characters, but may be binary or other forms as well. The data is device-specific.

My Talk Address (MTA) must be output to satisfy the GPIB requirement of only one talker at a time (any other talker will stop when MTA goes out). The MTA is not needed as far as the 8291 is concerned — it will be put into talk-only mode (ton).

This routine assumes a non-null listener list in that it

always sends Universal Unlisten. If it is desired to send data to the listeners previously addressed, one could add a check for a null list and not send UNL. Count must be 255 or less due to an 8 bit register. This routine also always uses an EOS character to terminate the string output; this could easily be eliminated and rely on the count. Items in brackets ( ) are optional and will not be included in the actual code in Appendix A.

## SEND:

Output-to-8291 MTA, UNL	;We will talk, nobody listen
Put EOS into 8291	;End of string compare character
While $20H \leq \text{listener} \leq 3EH$	;GPIB listen addresses are
output-to-8291 listener	;"space" thru ">" ASCII
Increment listen list pointer	;Address all listeners
Output-to-8292 GTSB	;8292 stops asserting ATN, go to standby
Enable-8291	
Output EO1 on EOS sent	;Send EO1 along with EOS character
If count < > 0 then	
While not (end or count = 0)	;Wait for EOS or end of count
(could check tout 2 here)	;Optionally check for stuck bus-tout 2
Output-to-8291 data	;Output all data, one byte at a time
Increment data buffer pointer	;8085 CREG will count for us
Decrement count	
Output-to-8292 TCSY	;8292 asserts ATN, take control sync.
(If tout3 then take control async)	;If unable to take control sync.
Enable 8291	;Restore 8291 to standard condition
No output EO1 on EOS sent	
Return	



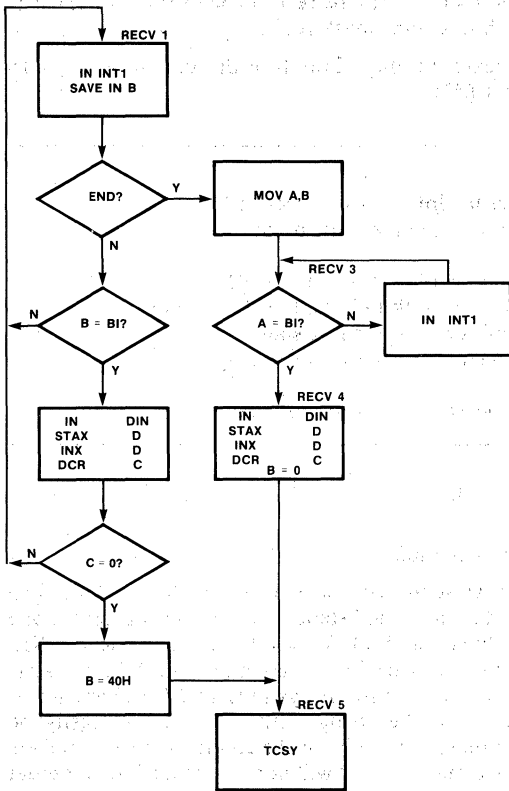


Figure 15. Flowchart For Receive Ending Conditions

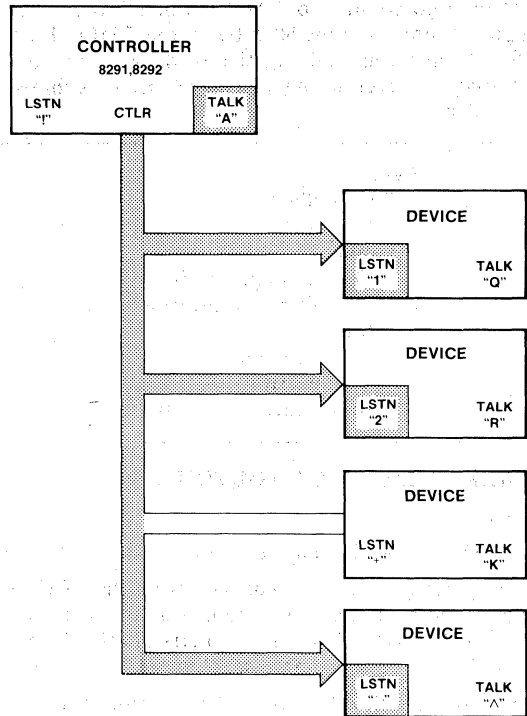


Figure 16. SEND to "1", "2", ">"; "ABCD"; EOS = "D"

**Receive Data**

*RECV* <talker> <count> <EOS> <data buffer pointer>

This system command is used to input data from a device. The data is typically a string of ASCII characters.

This routine is the dual of SEND. It assumes a new talker will be specified, a count of less than 257, and an EOS character to terminate the input. EOI received will also terminate the input. Figure 15 shows the flow chart for the RECV ending conditions. My Listen Address (MLA) is sent to keep the GPIB transactions totally regular to

facilitate analysis by a GPIB logic analyzer like the Ziotech ZT488. Otherwise, the bus would appear to have no listener even though the 8291 will be listening.

Note that although the count may go to zero before the transmission ends, the talker will probably be left in a strange state and may have to be cleared by the controller. The count ending of RECV is therefore used as an error condition in most situations.

# APPLICATIONS

## RECV:

```

Put EOS into 8291
If 40H ≤ talker ≤ 5EH then
  Output-to-8291 talker
  Increment talker pointer
Output-to-8291 UNL, MLA
Enable-8291
  Holdoff on end
  End on EOS received
  lon, reset ton
  Immediate execute pon
Output-to-8292 GTSB
While not (end or count = 0 (or tout2))

  Input-from-8291 data
  Increment data buffer pointer
  Decrement count
  (If count = 0 then error)
Output-to-8292 TCSY
  (If Tout3 then take control async.)
Enable-8291
  No holdoff on end
  No end on EOS received
  ton, reset lon
  Finish handshake
  Immediate execute pon
Return error-indicator
  
```

```

;End of string compare character
;GPIB talk addresses are
;“@” thru “^” ASCII
;Do this for consistency’s sake
;Everyone except us stop listening

;Stop when EOS character is
;Detected by 8291
;Listen only (no talk)

;8292 stops asserting ATN, go to standby
;wait for EOS or EOI or end of count
;optionally check for stuck bus-tout2
;input data, one byte at a time

;Use 8085 C register as counter
;Count should not occur before end
;8292 asserts ATN take control
;If unable to take control sync.
;Put 8291 back as needed for
;Controller activity and
;Clear holdoff due to end

;Complete holdoff due to end, if any
;Needed to reset lon
  
```

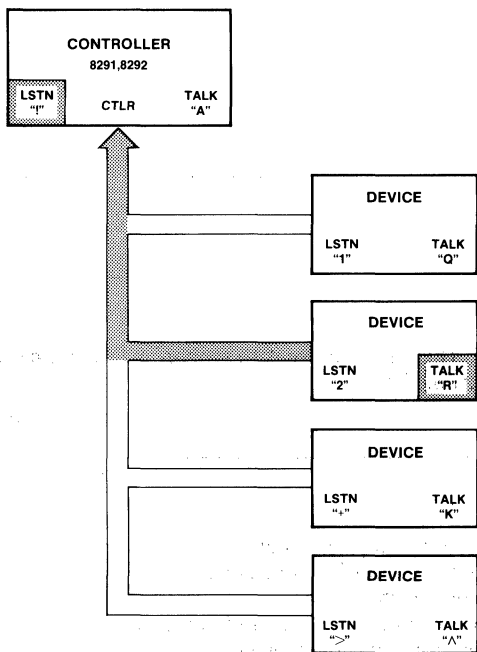


Figure 17. RECV from “R”; EOS = 0DH

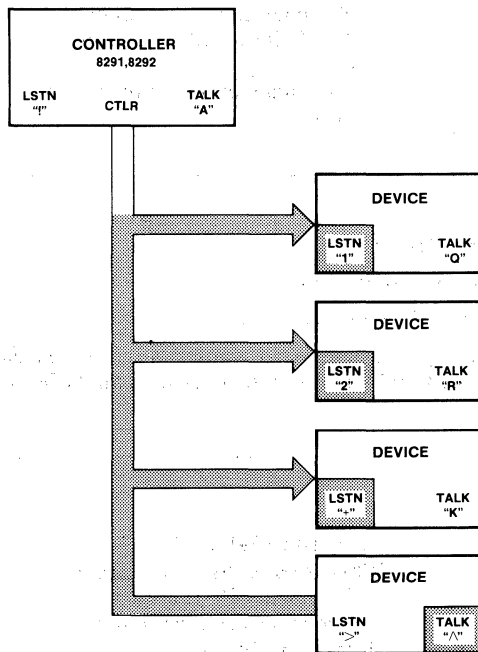


Figure 18. XFER from “^” to “1”, “2”, “+”; EOS = 0DH

# APPLICATIONS

---

## Transfer Data

*XFER* <Talker> <Listener list> <EOS>

This system command is used to transfer data from a talker to one or more listeners where the controller does not participate in the transfer of the ASCII data. This is accomplished through the use of the 8291's continuous acceptor handshake mode while in listen-only.

This routine assumes a device list that has the ASCII talker address as the first byte and the string (one or more) of ASCII listener addresses following. The EOS character or an EOI will cause the controller to take control synchronously and thereby terminate the transfer.

---

*XFER*:

Output-to-8291: Talker, UNL

While 20H ≤ listen ≤ 3EH

Output-to-8291: Listener

Increment listen list pointer

Enable-8291

lon, no ton

Continuous AH mode

End on EOS received

Immediate execute PON

Put EOS into 8291

Output-to-8292: GTSB

Upon end (or tout2) then

Take control synchronously

Enable-8291

Finish handshake

Not continuous AH mode

Not END on EOS received

ton

Immediate execute pon

Return

;Send talk address and unlisten

;Send listen address

;Controller is pseudo listener

;Handshake but don't capture data

;Capture EOS as well as EOI

;Initialize the 8291

;Set up EOS character

;Go to standby

;8292 waits for EOS or EOI and then

;Regains control

;Go to Ready for Data

---

## CONTROLLER

### Group Execute Trigger

*TRIG* <Listener list>

This system command causes a group execute trigger (GET) to be sent to all devices on the listener

list. The intended use is to synchronize a number of instruments.

---

*TRIG*:

Output-to-8291 UNL

While 20H ≤ listener ≤ 3EH

Output-to-8291 Listener

Increment listen list pointer

Output-to-8291 GET

Return

;Everybody stop listening

;Check for valid listen address

;Address each listener

;Terminate on any non-valid character

;Issue group execute trigger

---

# APPLICATIONS

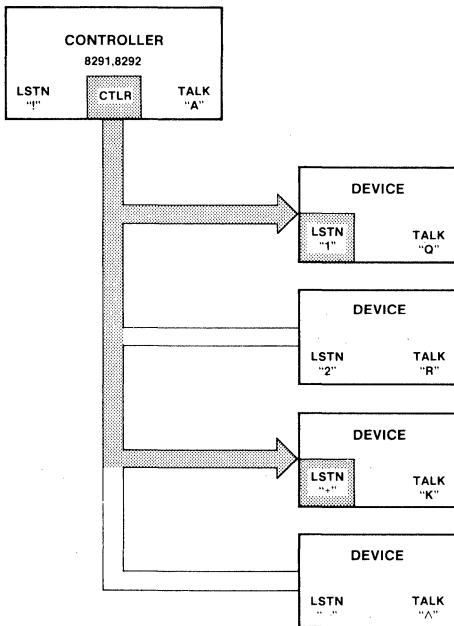


Figure 19. TRIG "1", "+"

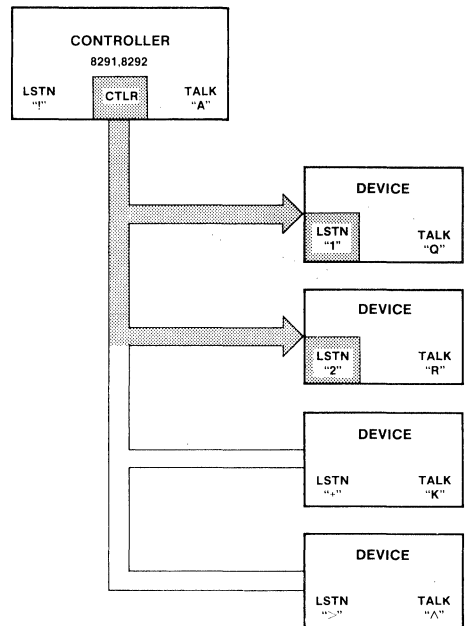


Figure 20. DCLR "1", "2"

## Device Clear

*DCLR* <Listener list >

This system command causes a device clear (SDC) to be sent to all devices on the listener list. Note that this is not intended to clear the GPIB interface

of the device, but should clear the device-specific logic.

### *DCLR*:

```
Output-to-8291 UNL
While 20H ≤ Listener ≤ 3EH
  Output-to-8291 listener
  Increment listen list pointer
Output-to-8291 SDC
Return
```

```
;Everybody stop listening
;Check for valid listen address
;Address each listener
;Terminate on any non-valid character
;Selective device clear
```

## Serial Poll

*SPOL* <Talker list> <status buffer pointer>

This system command sequentially addresses the designated devices and receives one byte of status from each. The bytes are stored in the buffer in the

same order as the devices appear on the talker list. MLA is output for completeness.

# APPLICATIONS

**SPOL:**

Output-to-8291 UNL, MLA, SPE

While 40H ≤ talker ≤ 5EH  
 Output-to-8291 talker  
 Increment talker list pointer  
 Enable-8291  
     lon, reset ton  
     Immediate execute pon  
 Output-to-8292 GTSB  
 Wait for data in (BI)  
 Output-to-8292 TCSY  
 Input-from-8291 data  
 Increment buffer pointer  
 Enable 8291  
     ton, reset lon  
     Immediate execute pon  
 Output-to-8291 SPD  
 Return

```

;Unlisten, we listen, serial poll enable
;Only one byte of serial poll
;Status wanted from each talker
;Check for valid transfer
;Address each device to talk
;One at a time

;Listen only to get status
;This resets ton
;Go to standby
;Serial poll status byte into 8291
;Take control synchronously
;Actually get data from 8291

;Resets lon
;Send serial poll disable after all devices polled
    
```

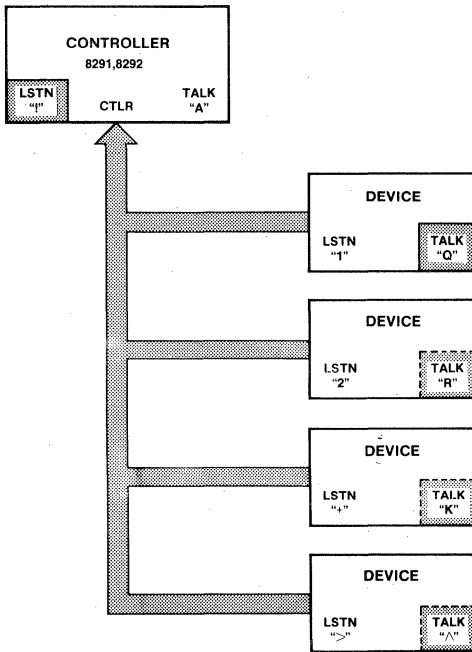


Figure 21. SPOL "Q", "R", "K", "^"

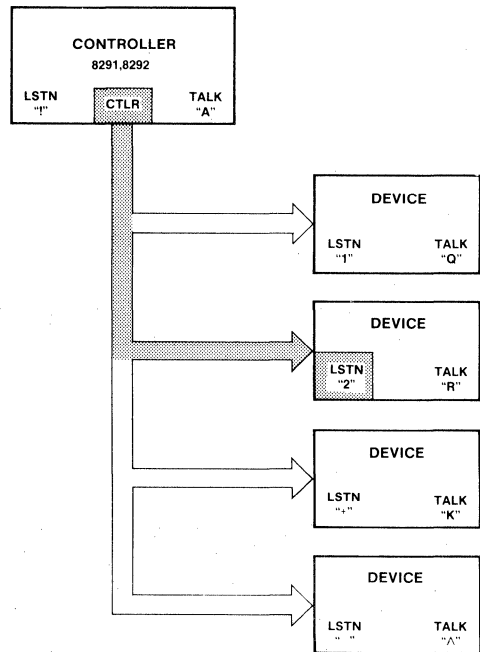


Figure 22. PPEN "2"; iP<sub>3</sub>P<sub>2</sub>P<sub>1</sub> = 0111B

**Parallel Poll Enable**

*PPEN* <Listener list> <Configuration Buffer pointer >

This system command configures one or more devices to respond to Parallel Poll, assuming they implement subset PPI. The configuration information is stored in a buffer with one byte per device in the same order as devices appear on the listener

list. The configuration byte has the format XXXXIP3P2P1 as defined by the IEEE Std. P3P2P1 indicates the bit # to be used for a response and I indicates the assertion value. See Sec. 2.9.3.3 of the Std. for more details.

# APPLICATIONS

*PPEN:*

```

Output-to-8291 UNL
While 20H ≤ Listener ≤ 3EH
  Output-to-8291 listener
  Output-to-8291 PPC, (PPE or data)
  Increment listener list pointer
  Increment buffer pointer
Return
;Universal unlisten
;Check for valid listener
;Stop old listener, address new
;Send parallel poll info
;Point to next listener
;One configuration byte per listener
    
```

## Parallel Poll Disable

*PPDS* <listener list>

This system command disables one or more devices from responding to a Parallel Poll by issuing a

Parallel Poll Disable (PPD). It does not deconfigure the devices.

*PPDS:*

```

Output-to-8291 UNL
While 20H ≤ Listener ≤ 3EH
  Output-to-8291 listener
  Increment listener list pointer
Output-to-8291 PPC, PPD
Return
;Universal Unlisten
;Check for valid listener
;Address listener
;Disable PP on all listeners
    
```

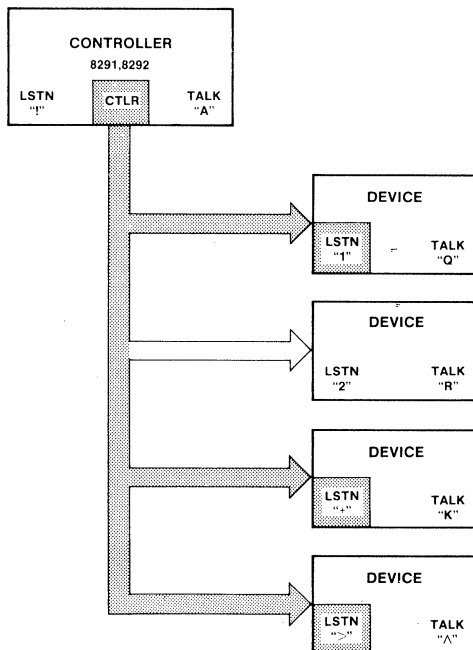


Figure 23. PPDS "1", "4", "5"

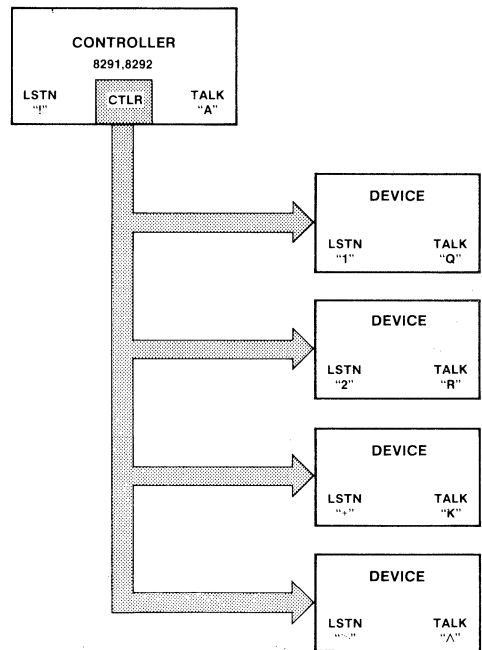


Figure 24. PPUN

# APPLICATIONS

## Parallel Poll Unconfigure

### PPUN

This system command deconfigures the Parallel Poll response of all devices by issuing a Parallel Poll Unconfigure message.

---

```
PPUN:
  Output-to-8291 PPU          ;Unconfigure all parallel poll
  Return
```

---

## Conduct a Parallel Poll

### PPOL

This system command causes the controller to conduct a Parallel Poll on the GPIB for approximately 12.5 usec (at 6 MHz). Note that a parallel poll does not use the handshake; therefore, to ensure that the device knows whether or not its positive response

was observed by the controller, the CPU should explicitly acknowledge each device by a device-dependent data string. Otherwise, the response bit will still be set when the next Parallel Poll occurs. This command returns one byte of status.

---

```
PPOL:
  Enable-8291
  lon                          ;Listen only
  Immediate execute pon       ;This resets ton
  Output-to-8292 EXPP         ;Execute parallel poll
  Upon BI                      ;When byte is input
  Input-from-8291 data        ;Read it
  Enable-8291
  ton                          ;Talk only
  Immediate execute pon       ;This resets lon
  Return Data (status byte)
```

---

## Pass Control

### PCTL <talker>

This system command allows the controller to relinquish active control of the GPIB to another controller. Normally some software protocol should already have informed the controller to expect this, and under what conditions to return control. The

8291 must be set up to become a normal device and the CPU must handle all commands passed through, otherwise control cannot be returned (see Receive Control below). The controller will go idle.

---

```
PCTL:
  If 40H ≤ talker ≤ 5EH then
    if talker < > MTA then
      output-to-8291 talker, TCT
      Enable-8291
      not ton, not lon
      Immediate execute pon
      My device address, mode 1
      Undefined command pass through
      (Parallel Poll Configuration)
      Output-to-8292 GIDL
      Return
    ;Cannot pass control to myself
    ;Take control message to talker
    ;Set up 8291 as normal device
    ;Reset ton and lon
    ;Put device number in Register 6
    ;Required to receive control
    ;Optional use of PP
    ;Put controller in idle
```

---

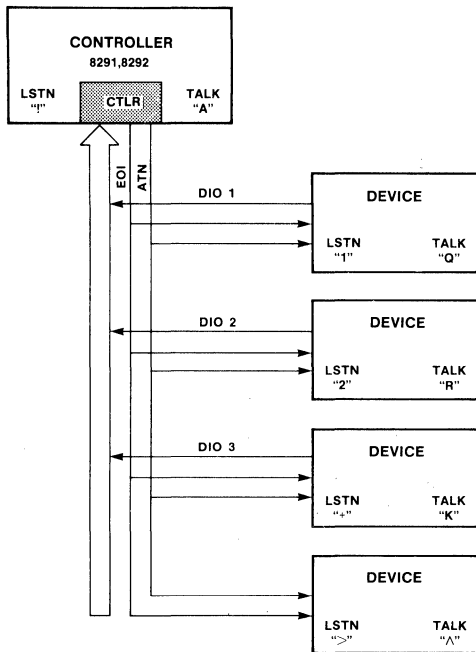


Figure 25. PPOL

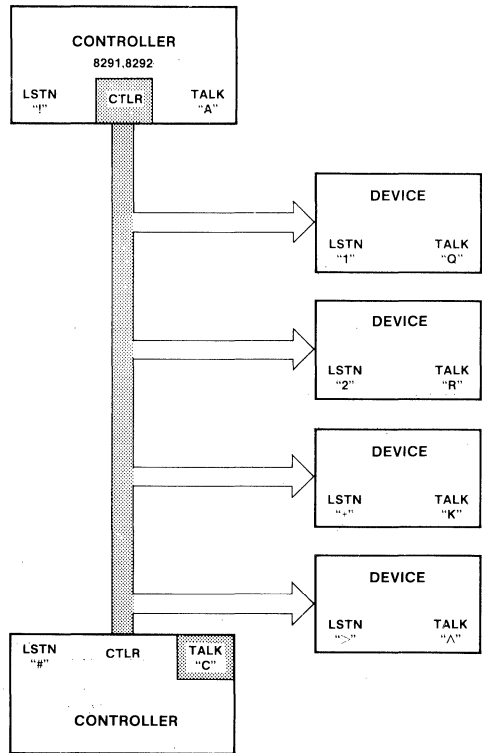


Figure 26. PCTL "C"

## Receive Control

### RCTL

This system command is used to get control back from the current controller-in-charge if it has passed control to this inactive controller. Most GPIB systems do not use more than one controller and therefore would not need this routine.

To make passing and receiving control a manageable event, the system designer should specify a

protocol whereby the controller-in-charge sends a data message to the soon-to-be-active controller. This message should give the current state of the system, why control is being passed, what to do, and when to pass control back. Most of these issues are beyond the scope of this Ap Note.

### RCTL:

```

Upon CPT
  If (command=TCT) then
    If TA then
      Enable-8291
      Disable major device number
      ton
      Mask off interrupts
      Immediate execute pon
  
```

```

;Wait for command pass through bit in 8291
;If command is take control and
;We are talker addressed
;Controller will use ton and lon
;Talk only mode
  
```



# APPLICATIONS

```

Output-to-8292 TCNTR           ;Take (receive) control
Enable-8291                     ;Release handshake
    Valid command
    Return valid
Else
    Enable-8291
    Invalid command           ;Not talker addr. so TCT not for us
Else
    Enable-8291
    Invalid command           ;Not TCT, so we don't care
Return invalid
    
```

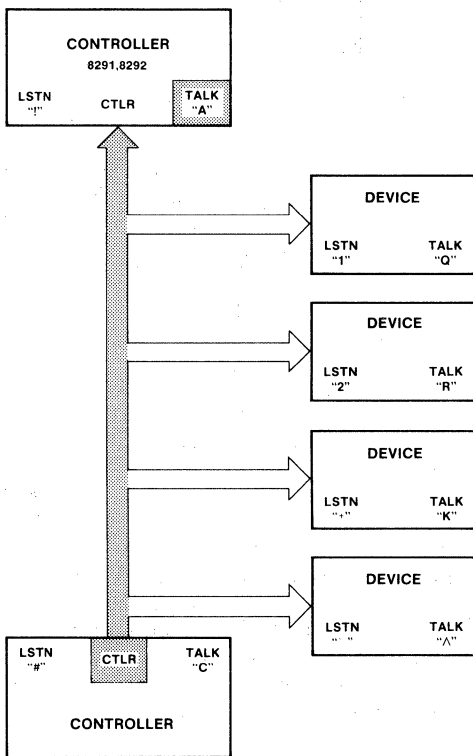


Figure 27. RCTL

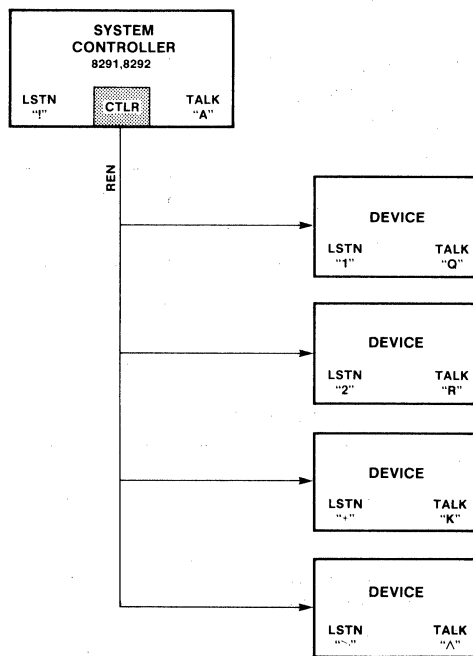


Figure 28. REME

## Service Request

### SRQD

This system command is used to detect the occurrence of a Service Request on the GPIB. One or more devices may assert SRQ simultaneously, and

the CPU would normally conduct a Serial Poll after calling this routine to determine which devices are SRQing.

# APPLICATIONS

```

SRQD:
  If SRQ then                               ;Test 92 status bit
    Output-to-8292 IACK.SRQ                 ;Acknowledge it
    Return SRQ
  Else return no SRQ
  
```

## SYSTEM CONTROLLER

### Remote Enable

#### REME

This system command asserts the Remote Enable line (REN) on the GPIB. The devices will not go

remote until they are later addressed to listen by some other system command.

```

REME:
  Output-to-8292 SREM                       ;8292 asserts remote enable line
  Return
  
```

### Local

#### LOCL

This system command deasserts the REN line on the GPIB. The devices will go local immediately.

```

LOCL:
  Output-to-8292 SLOC                       ;8292 stops asserting remote enable
  Return
  
```

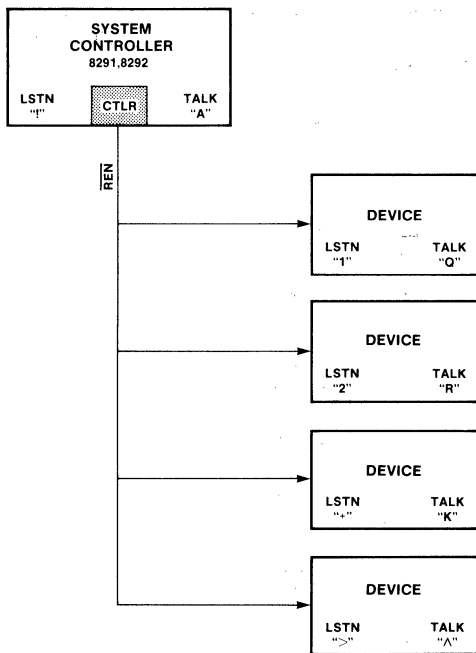


Figure 29. LOCL

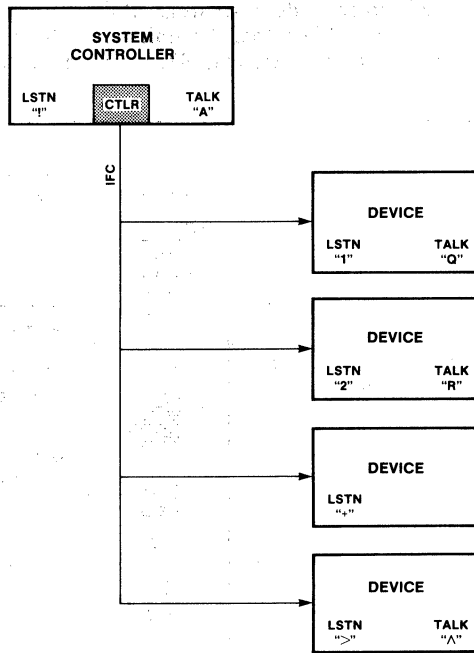


Figure 30. IFCL

**Interface Clear/Abort**

*IFCL*

This system command asserts the GPIB's Interface Clear (IFC) line for at least 100 microseconds. This causes all interface logic in all devices to go to a known state. Note that the device itself may or

may not be reset, too. Most instruments do totally reset upon IFC. Some devices may require a DCLR as well as an IFCL to be completely reset. The (system) controller becomes Controller-in-Charge.

*IFCL:*

Output-to-8292 ABORT  
Return

;8292 asserts Interface Clear  
;For 100 microseconds

**INTERRUPTS AND DMA CONSIDERATIONS**

The previous sections have discussed in detail how to use the 8291, 8292, 8293 chip set as a GPIB controller with the software operating in a polling mode and using programmed transfer of the data. This is the simplest mode of use, but it ties up the microprocessor for the duration of a GPIB transaction. If system design constraints do not allow this, then either Interrupts and/or DMA may be used to free up processor cycles.

The 8291 and 8292 provide sufficient interrupts that one may return to do other work while waiting for such things as 8292 Task Completion, 8291 Next Byte In, 8291 Last Byte Out, 8292 Service Request

In, etc. The only difficulty lies in integrating these various interrupt sources and their matching routines into the overall system's interrupt structure. This is highly situation-specific and is beyond the scope of this Ap Note.

The strategy to follow is to replace each of the WAIT routines (see Appendix A) with a return to the main code and provide for the corresponding interrupt to bring the control back to the next section of GPIB code. For example WAITO (Wait for Byte Out of 8291) would be replaced by having the BO interrupt enabled and storing the (return) address of the next instruction in a known place. This co-routine structure will then be activated by a BO interrupt. Fig. 31 shows an example of the flow of control.

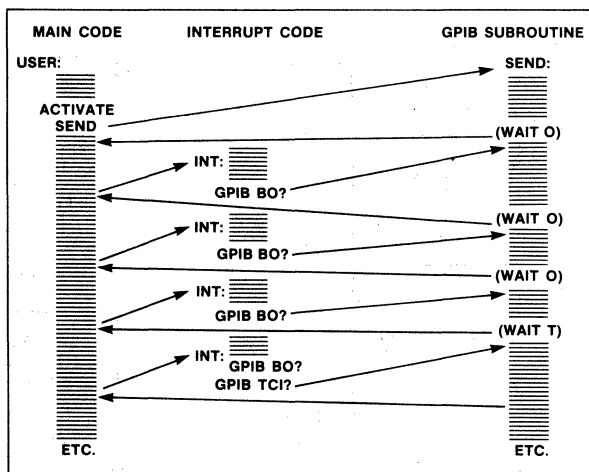


Figure 31. GPIB Interrupt & Co-Routine Flow of Control

DMA is also useful in relieving the processor if the average length of a data buffer is long enough to overcome the extra time used to set up a DMA chip. This decision will also be a function of the data rate of the instrument. The best strategy is to use the DMA to handle only the data buffer transfers on SEND and RECV and to do all the addressing and control just as shown in the driver descriptions.

Another major reason for using a DMA chip is to increase the data rate and therefore increase the overall transaction rate. In this case the limiting factor becomes the time used to do the addressing and control of the GPIB using software like that in Appendix A. The data transmission time becomes insignificant at DMA speeds unless extremely long buffers are used.

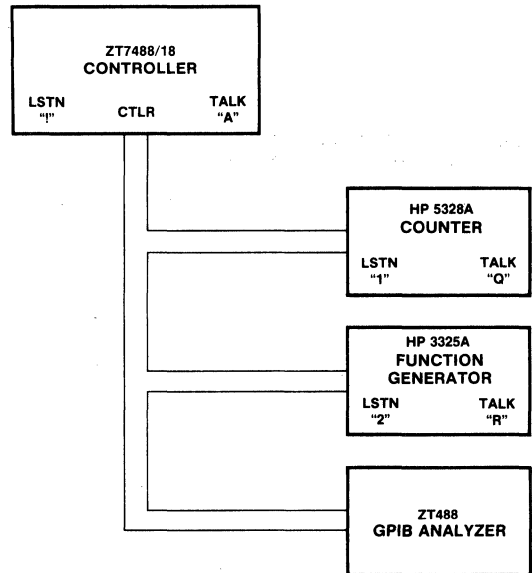
Refer to Figure 11 for a typical DMA and interrupt based design using the 8291, 8292, 8293. A system like this can achieve a 250K byte transfer rate while under DMA control.

### APPLICATION EXAMPLE

This section will present the code required to operate a typical GPIB instrument set up as shown in Fig. 32. The HP5328A universal counter and the HP3325 function generator are typical of many GPIB devices; however, there are a wide variety of software protocols to be found on the GPIB. The Ziatech ZT488 GPIB analyzer is used to single step the bus to facilitate debugging the system. It also serves as a training/familiarization aid for newcomers to the bus.

This example will set up the function generator to output a specific waveform, frequency and ampli-

tude. It will then tell the counter to measure the frequency and Request Service (SRQ) when complete. The program will then read in the data. The assembled source code will be found at the end of Appendix A.



**Figure 32. GPIB Example Configuration**

SEND

```
LSTN: "2", COUNT: 15, EOS: 0DH, DATA: "FU1FR37KHAM2VO (CR)"
;SETS UP FUNCTION GEN. TO 37 KHZ SINE, 2 VOLTS PP
;COUNT EQUAL TO # CHAR IN BUFFER
;EOS CHARACTER IS (CR) = 0DH = CARRIAGE RETURN
```

SEND

```
LSTN: "1", COUNT: 6, EOS: "T" DATA: "PR4G7T"
;SETS UP COUNTER FOR P:INITIALIZE, F4: FREQ CHAN A
;      G7:0.1 HZ RESOLUTION, T:TRIGGER AND SRQ
;COUNT IS EQUAL TO # CHAR
```

WAIT FOR SRQ

```
SPOL TALK: "Q", DATA: STATUS 1
;CLEARS THE SRO — IN THIS EXAMPLE ONLY FREQ CTR ASSERTS SRQ
```

```
RECV TALK: "Q", COUNT: 17, EOS: 0AH,
DATA: "+ 37000.0E+0" (CR) (LF)
;GETS 17 BYTES OF DATA FROM COUNTER
;COUNT IS EXACT BUFFER LENGTH
;DATA SHOWN IS TYPICAL HP5328A READING THAT WOULD BE RECEIVED
```

## CONCLUSION

This Application Note has shown a structured way to view the IEEE 488 bus and has given typical code sequences to make the Intel 8291, 8292, and 8293's behave as a controller of the GPIB. There are other ways to use the chip set, but whatever solution is chosen, it must be integrated into the overall system software.

The ultimate reference for GPIB questions is the IEEE Std 488, -1978 which is available from IEEE, 345 East 47th St., New York, NY, 10017. The ultimate reference for the 8292 is the source listing for it (remember it's a pre-programmed UPI-41A) which is available from INSITE, Intel Corp., 3065 Bowers Ave., Santa Clara, CA 95051.

## APPENDIX A

ISIS-II 8080/8085 MACRO ASSEMBLER, V3.0  
GPIB CONTROLLER SUBROUTINES

LOC	OBJ	LINE	SOURCE STATEMENT
		1	STITLE('GPIB CONTROLLER SUBROUTINES')
		2	;
		3	; GPIB CONTROLLER SUBROUTINES
		4	;
		5	;
		6	; for Intel 8291, 8292 on ZT 7488/18
		7	Bert Forbes, Ziatech Corporation
		8	2410 Broad Street
		9	San Luis Obispo, CA, USA 93401
		10	;
		11	;
		12	General Definitions & Equates
		13	8291 Control Values
		14	;
1000		15	ORG 1000H ; For ZT7488/18 w/8085
		16	;
0060		17	PRT91 EQU 60H ;8291 Base Port #
		18	;
		19	Reg #0 Data in & Data out
0060		20	DIN EQU PRT91+0 ;91 Data in reg
0060		21	DOUT EQU PRT91+0 ;91 Data out reg
		22	;
		23	Reg # 1 Interrupt 1 Constants
0061		24	INT1 EQU PRT91+1 ;INT Reg 1
0061		25	INTM1 EQU PRT91+1 ;INT Mask Reg. 1
0002		26	BOM EQU 02 ;91 BO INTRP Mask
0001		27	BIM EQU 01 ;91 BI INTRP Mask
0010		28	ENDMK EQU 10H ;91 END INTRP Mask
0080		29	CPT EQU 80H ;91 command pass thru int bit
		30	;
		31	Reg #2 Interrupt 2
0062		32	INT2 EQU PRT91+2
		33	;
		34	Reg #4 Address Mode Constants
0064		35	ADRMD EQU PRT91+4 ;91 address mode register #
0080		36	TON EQU 80H ;91 talk only mode & not listen only
0040		37	LON EQU 40H ;91 listen only & not ton
00C0		38	TLOH EQU 0C0H ;91 talk & listen only
0001		39	MODE1 EQU 01 ;mode 1 addressing for device
		40	;
		41	Reg #4 (Read) Address Status Register
0064		42	ADRST EQU PRT91+4 ;reg #4
0020		43	EOIST EQU 20H
0002		44	TA EQU 2
0001		45	LA EQU 1 ;listener active
		46	;
		47	Reg #5 (Write) Auxillary Mode Register
0065		48	AUXMD EQU PRT91+5 ;91 auxillary mode register #
0023		49	CLKRT EQU 23H ;91 3 Mhz clock input

# APPLICATIONS

```

0003      50 FNHSK   EQU      03      ;91 finish handshake command
0006      51 SDEOI   EQU      06      ;91 send EOI with next byte
0008      52 AXRA    EQU      80H     ;91 aux reg A pattern
0001      53 HOHSK   EQU      1       ;91 hold off handshake on all bytes
0002      54 HOEND   EQU      2       ;91 hold off handshake on end
0003      55 CAHCY   EQU      3       ;91 continuous AH cycling
0004      56 EDEOS   EQU      4       ;91 end on EOS received
0008      57 EOIS    EQU      8       ;91 output EOI on EOS sent
000F      58 VSCMD   EQU      0FH     ;91 valid command pass through
0007      59 NVCMD   EQU      07H     ;91 invalid command pass through
00A0      60 AXRB    EQU      0A0H    ;Aux. reg. B pattern
0001      61 CPTEN   EQU      01H     ;command pass thru enable
62 ;
63 ;
0065      64 CPTRG   EQU      PRT91+5  ;
65 ;
66 ;      Reg #5 Address 0/1 reg. constants
0066      67 ADR01   EQU      PRT91+6  ;
0060      68 DTDL1   EQU      60H     ;Disable major talker & listener
00E0      69 DTDL2   EQU      0E0H    ;Disable minor talker & listener
70 ;
71 ;      Reg #7 EOS Character Register
0067      72 EOSR    EQU      PRT91+7  ;
73 ;
74 ;
75 ;      8292 CONTROL VALUES
76 ;
77 ;
78 ;
0068      79 PRT92   EQU      PRT91+8  ;8292 Base Port # (CS7)
80 ;
0068      81 INTMR   EQU      PRT92+0  ;92 INTRP Mask Reg
00A0      82 INTM    EQU      0A0H     ;TCI
83 ;
0068      84 ERRM    EQU      PRT92+0  ;92 Error Mask Reg
0001      85 TOUT1   EQU      01       ;92 Time Out for Pass Control
0002      86 TOUT2   EQU      02       ;92 Time Out for Standby
0004      87 TOUT3   EQU      04       ;92 Time Out for Take Control Sync
0068      88 EVREG   EQU      PRT92+0  ;92 Event Counter Pseudo Reg
0058      89 TOREG   EQU      PRT92+0  ;92 Time Out Pseudo Reg
90 ;
0069      91 CMD92   EQU      PRT92+1  ;92 Command Register
92 ;
0069      93 INTST   EQU      PRT92+1  ;92 Interrupt Status Reg
0010      94 EVBIT   EQU      10H     ;Event Counter Bit
0002      95 IBFBT   EQU      02       ;Input Buffer Full Bit
0020      96 SR0BT   EQU      20H     ;Seq bit
97 ;
0068      98 ERFLG   EQU      PRT92+0  ;92 Error Flag Pseudo Reg
0068      99 CLRST   EQU      PRT92+0  ;92 Controller Status Pseudo Reg
0068      100 BUSST  EQU      PRT92+0  ;92 GPIB (Bus) Status Pseudo Reg
0068      101 EVCST   EQU      PRT92+0  ;92 Event Counter Status Pseudo Reg
0068      102 TOST    EQU      PRT92+0  ;92 Time Out Status Pseudo Reg
103 ;
104 ;      8292 OPERATION COMMANDS
105 ;
106 ;
00F0      107 SPCNI   EQU      0F0H    ;Stop Counter Interrupts
00F1      108 GIDL   EQU      0F1H    ;Go to idle
00F2      109 RSET   EQU      0F2H    ;Reset
00F3      110 RSTI   EQU      0F3H    ;Reset Interrupts
00F4      111 GSEC   EQU      0F4H    ;Goto standby, enable counting
00F5      112 EXPP   EQU      0F5H    ;Execute parallel poll
00F6      113 GTSB   EQU      0F6H    ;Goto standby
00F7      114 SLOC   EQU      0F7H    ;Set local mode
00F8      115 SREM   EQU      0F8H    ;Set interface to remote
00F9      116 ABORT   EQU      0F9H    ;Abort all operation, clear interface
00FA      117 TCNTR   EQU      0FAH    ;Take control (Receive control)
00FC      118 TCASY   EQU      0FCH    ;Take control asynchronously
00FD      119 TCSY   EQU      0FDH    ;Take control synchronously
00FE      120 STCNI   EQU      0FEH    ;Start counter interrupts
121 ;
122 ;

```

# APPLICATIONS

```

123 ;          8292  UTILITY COMMANDS
124 ;
125 ;
00E1 126 WOUT EQU 0E1H ;Write to timeout req
00E2 127 WEVC EQU 0E2H ;Write to event counter
00E3 128 REVC EQU 0E3H ;Read event counter status
00E4 129 RERF EQU 0E4H ;Read error flag reg
00E5 130 RINM EQU 0E5H ;Read interrupt mask reg
00E6 131 RCST EQU 0E6H ;Read controller status reg
00E7 132 RBST EQU 0E7H ;Read GPIB Bus status reg
00E9 133 RTOUT EQU 0E9H ;Read timeout status reg
00EA 134 RERM EQU 0EAH ;Read error mask reg
00EB 135 IACK EQU 0BH ;Interrupt Acknowledge
136 ;
137 ;
138 ;          PORT F BIT ASSIGNMENTS
139 ;
140 ;
141 ;
006F 142 PRTF EQU PRT91+0FH ;ZT7488 port 6F for interrupts
0002 143 TCIF EQU 02H ;Task complete interrupt
0004 144 SPIF EQU 04H ;Special interrupt
0008 145 OBFF EQU 08H ;92 Output (to CPU) Buffer full
0010 146 IBFF EQU 10H ;92 Input (from CPU) Buffer empty
0001 147 BOF EQU 01H ;91 Int line (BO in this case)
148 ;
149 ;          GPIB MESSAGES (COMMANDS)
150 ;
0001 151 MDA EQU 1 ;My device address is 1
0041 152 MTA EQU MDA+40H ;My talk address is 1 ("A")
0021 153 MLA EQU MDA+20H ;My listen address is 1 ("!")
003F 154 UNL EQU 3FH ;Universal unlisten
0008 155 GET EQU 08 ;Group Execute Triqger
0004 156 SDC EQU 04H ;Device Clear
0018 157 SPE EQU 18H ;Serial poll enable
0019 158 SPD EQU 19H ;Serial poll disable
0005 159 PPC EQU 05 ;Parallel poll configure
0070 160 PPD EQU 70H ;Parallel poll disable
0060 161 PPE EQU 60H ;Parallel poll disable
0015 162 PPU EQU 15H ;Parallel poll unconfigured
0009 163 TCT EQU 09 ;Take control (pass control)
164 ;
165 ;          MACRO DEFINITIONS
166 ;
167 ;
168 ;
169 SETF MACRO ;Sets flags on A register
170 ORA A
171 ENDM
172 ;
173 WAITO MACRO ;Wait for last 91 byte to be done
174 LOCAL WAITL
175 WAITL: IN INT1 ;Get Int1 status
176 ANI BOM ;Check for byte out
177 JZ WAITL ;If not, try again
178 ENDM ;until it is
179 ;
180 ;
181 WAITI MACRO ;Wait for 91 byte to be input
182 LOCAL WAITL
183 WAITL: IN INT1 ;Get INT1 status
184 MOV B,A ;Save status in B
185 ANI BIM ;Check for byte in
186 JZ WAITL ;If not, just try again
187 ENDM ;until it is
188 ;
189 WAITX MACRO ;Wait for 92's TCI to go false
190 LOCAL WAITL
191 WAITL: IN PRTF
192 ANI TCIF
193 JNZ WAITL
194 ENDM
195 ;

```

# APPLICATIONS

```

196 WAITT  MACRO
197         LOCAL   WAITL
- 198 WAITL: IN      PRTF   ;Get task complete int,etc.
- 199         ANI    TCIF   ;Mask it
- 200         JZ     WAITL  ;Wait for task to be complete
201         ENDM
202
203 RANGE   MACRO   LOWER,UPPER,LABEL
204         ;Checks for value in range
205         ;branches to label if not
206         ;in range. Falls through if
207         ;lower <= ( H ) ( L ) <= upper.
208         ;Get next byte.
- 209         MOV    A,M
- 210         CPI    LOWER
- 211         JM     LABEL
- 212         CPI    UPPER+1
- 213         JP     LABEL
214         ENDM
215 ;
216 CLRA    MACRO
- 217         XRA    A      ;A XOR A =0
218         ENDM
219 ;
220 ;
221 ;       All of the following routines have these common
222 ;       assumptions about the state of the 8291 & 8292 upon entry
223 ;       to the routine and will exit the routine in an identical state.
224 ;
225 ;       8291:  BO is or has been set,
226 ;             All interrupts are masked off
227 ;             TON mode, not LA
228 ;             No holdoffs in effect or enabled
229 ;             No holdoffs waiting for finish command
230 ;
231 ;       8292:  ATN asserted (active controller)
232 ;             note: RCTL is an exception--- it expects
233 ;             to not be active controller
234 ;             Any previous task is complete & 92 is
235 ;             ready to receive next command.
236 ;       8085:  Pointer registers (DE,HL) end one
237 ;             beyond last legal entry
238 ;*****
239 ;
240 ;
241 ;       INITIALIZATION ROUTINE
242 ;
243 ;INPUTS:      None
244 ;OUTPUTS:     None
245 ;CALLS:       None
246 ;DESTROYS:    A,F
247 ;
1000 3EA0 248 INIT:  MVI    A,INTM ;Enable TCI
1002 D368 249         OUT    INTMR ;Output to 92's intr. mask reg
1004 3E60 250         MVI    A,DTDL1 ;Disable major talker/listener
1006 D366 251         OUT    ADR01
1008 3EE0 252         MVI    A,DTDL2 ;Disable minor talker/listener
100A D366 253         OUT    ADR01
100C 3E80 254         MVI    A,TON   ;Talk only mode
100E D364 255         OUT    ADRMD
1010 3E23 256         MVI    A,CLKRT ;3 MHZ for delay timer
1012 D355 257         OUT    AUXMD
258         CLRA
1014 AF   259+        XRA    A      ;A XOR A =0
1015 D361 260         OUT    INT1
1017 D362 261         OUT    INT2   ;Disable all 91 mask bits
1019 D365 262         OUT    AUXMD  ;Immediate execute PON
101B C9   263         RET
264 ;
265 ;*****
266 ;
267 ;
268 ;       SEND ROUTINE
269 ;

```



# APPLICATIONS

```

270 ;
271 ;
272 ;      INPUTS:      HL listener list pointer
273 ;      DE data buffer pointer
274 ;      C count-- 0 will cause no data to be sent
275 ;      b EOS character-- software detected
276 ;      OUTPUTS:      none
277 ;      CALLS:      none
278 ;      DESTROYS:     A, C, DE, HL, F
279 ;
280 ;
281 ;
101C 3E41 282 SEND:  MVI  A,MTA ;Send MTA to turn off any
101E D360 283      OUT  DOUT ;previous talker
284      WAITO
1020 DB61 285+??0001: IN  INT1 ;Get Intl status
1022 E602 286+      ANI  BOM  ;Check for byte out
1024 CA2010 287+      JZ   ??0001 ;If not, try again
1027 3E3F 288      MVI  A,UNL ;Send universal unlisten
1029 D360 289      OUT  DOUT ;to stop previous listeners
102B 78 290      MOV  A,B  ;Get EOS character
102C D367 291      OUT  EOSR  ;Output it to 8291
292      ;while listener.....
293 SEND1: RANGE 20H,3EH,SEND2 ;Check next listen address
294+      ;Checks for value in range
295+      ;branches to label if not
296+      ;in range. Falls through if
297+      ;lower <= ( (H)(L) ) <= upper.
298+      ;Get next byte.
102E 7E 299+      MOV  A,M
102F FE20 300+      CPI  20H
1031 FA4710 301+      JM  SEND2
1034 FE3F 302+      CPI  3EH+1
1036 F24710 303+      JP  SEND2
304      WAITO ;Wait for previous listener sent
1039 DB61 305+??0002: IN  INT1 ;Get Intl status
103B E602 306+      ANI  BOM  ;Check for byte out
103D CA3910 307+      JZ   ??0002 ;If not, try again
1040 7E 308      MOV  A,M  ;Get this listener
1041 D360 309      OUT  DOUT ;Output to GPIB
1043 23 310      INX  H  ;Increment listener list pointer
1044 C32E10 311      JMP  SEND1 ;Loop till non-valid listener
312      ;Enable 91 ending conditions
313 SEND2: WAITO ;Wait for lstn addr accepted
314+??0003: IN  INT1 ;Get Intl status
1049 E602 315+      ANI  BOM  ;Check for byte out
104B CA4710 316+      JZ   ??0003 ;If not, try again
317      ;WAITO required for early versions
318      ;of 8292 to avoid GTSB before DAC
104E 3E66 319      MVI  A,GTSB ;Goto standby
1050 D369 320      OUT  CMD92 ;
1052 3E88 321      MVI  A,AXRA+EOIS ;Send EOI with EOS character
1054 D365 322      OUT  AUXMD
323      WAITX ;Wait for TCI to go false
1055 DB6F 324+??0004: IN  PRTF
1058 E602 325+      ANI  TCIF
105A C25610 326+      JNZ  ??0004
327      WAITT ;Wait for TCI on GTSB
105D DB6F 328+??0005: IN  PRTF ;Get task complete int,etc.
105F E602 329+      ANI  TCIF ;Mask it
1061 CA5D10 330+      JZ   ??0005 ;Wait for task to be complete
331
332 ;      delete next 3 instructions to make count of 0=256
333 ;
1064 79 334      MOV  A,C ;Get count
335      SETF ;Set flags
336+      ORA  A
1065 B7 337      JZ   SEND6 ;If count=0, send no data
1066 CA8810 338 SEND3: LDAX D ;Get data byte
1069 1A 339      OUT  DOUT ;Output to GPIB
106A D360 340      CMP  B ;Test EOS ...this is faster
106C B8 341      ;and uses less code than using
342      ;91's END or EOI bits

```

# APPLICATIONS

```

106D CA7F10      343      JZ      SEND5      ;If char = EOS , go finish
                 344 SEND4: WAITO
1070 DB61        345+??0006: IN      INT1      ;Get Intl status
1072 E602        346+      ANI      BOM       ;Check for byte out
1074 CA7010      347+      JZ      ??0006 ;If not, try again
1077 13          348      INX      D         ;Increment buffer pointer
1078 0D          349      DCR      C         ;Decrement count
1079 C26910      350      JNZ     SEND3     ;If count < > 0, go send
107C C38810      351      JMP      SEND6     ;Else go finish
107F 13          352 SEND5: INX      D         ;for consistency
1080 0D          353      DCR      C         ; " "
                 354      WAITO
                                     ;This ensures that the standard entry
1081 DB61        355+??0007: IN      INT1      ;Get Intl status
1083 E602        356+      ANI      BOM       ;Check for byte out
1085 CA8110      357+      JZ      ??0007 ;If not, try again
                 358      ;assumptions for the next subroutine are met
1088 3EFD        359 SEND6: MVI      A,TCSY ;Take control synchronously
108A D369        360      OUT      CMD92
108C 3E80        361      MVI      A,AXRA ;Reset send EOI on EOS
108E D365        362      OUT      AUXMD
                 363      WAITX     ;Wait for TCI false
1090 DB6F        364+??0008: IN      PRTF     ;
1092 E602        365+      ANI      TCIF     ;
1094 C29010      366+      JNZ     ??0008 ;
                 367      WAITX     ;Wait for TCI
1097 DB6F        368+??0009: IN      PRTF     ;Get task complete int,etc.
1099 E602        369+      ANI      TCIF     ;Mask it
109B CA9710      370+      JZ      ??0009 ;Wait for task to be complete
109E C9          371      RET
372 ;*****
373 ;
374 ;      RECEIVE ROUTINE
375 ;
376 ;
377 ;INPUT:      HL talker pointer
378 ;           DE data buffer pointer
379 ;           C count (max buffer size) 0 implies 256
380 ;           B EOS character
381 ;OUTPUT:    Fills buffer pointed at by DE
382 ;CALLS:     None
383 ;DESTROYS:  A, BC, DE, HL, F
384 ;
385 ;RETURNS:    A=0 normal termination--EOS detected
386 ;           A=40 Error--- count overrun
387 ;           A<40 or A>5EH Error--- bad talk address
388 ;
389 ;
109F 78          390 RECV:  MOV      A,B      ;Get EOS character
10A0 D367        391      OUT      EOSR     ;Output it to 91
                 392      RANGE    40H,5EH,RECV6
                 393+      ;Checks for value in range
                 394+      ;branches to label if not
                 395+      ;in range. Falls through if
                 396+      ;lower <= ( (H)(L) ) <= upper.
                 397+      ;Get next byte.
10A2 7E          398+      MOV      A,M
10A3 FE40        399+      CPI      40H
10A5 FA3911      400+      JM      RECV6
10A8 FE5F        401+      CPI      5EH+1
10AA F23911      402+      JP      RECV6
                 403      ;valid if 40H<= talk <=5EH
10AD D360        404      OUT      DOUT    ;Output talker to GPIB
10AF 23          405      INX      H         ;Incr pointer for consistency
                 406      WAITO
10B0 DB61        407+??0010: IN      INT1      ;Get Intl status
10B2 E602        408+      ANI      BOM       ;Check for byte out
10B4 CAB010      409+      JZ      ??0010 ;If not, try again
10B7 3E3F        410      MVI      A,UNL    ;Stop other listeners
10B9 D360        411      OUT      DOUT
                 412      WAITO
10BB DB61        413+??0011: IN      INT1      ;Get Intl status
10BD E602        414+      ANI      BOM       ;Check for byte out
10BF CABB10      415+      JZ      ??0011 ;If not, try again

```

# APPLICATIONS

```

10C2 3E21      416      MVI      A,MLA      ;For completeness
10C4 D350      417      OUT      DOUT
10C6 3E86      418      MVI      A,AXRA+HOEND+EDEOS      ;End when
10C8 D355      419      OUT      AUXMD      ;EOS or EOI & Holdoff
420      WAITO
10CA DB61      421+??0012: IN      INT1      ;Get Intl status
10CC E602      422+      ANI      BOM      ;Check for byte out
10CE CACA10    423+      JZ      ??0012    ;If not, try again
10D1 3E40      424      MVI      A,LON      ;Listen only
10D3 D364      425      OUT      ADRMD
426      CLRA      ;Immediate XEQ PON
10D5 AF        427+      XRA      A      ;A XOR A =0
10D6 D365      428      OUT      AUXMD
10D8 3EF6      429      MVI      A,GTSB    ;Goto standby
10DA D359      430      OUT      CMD92
431      WAITX      ;Wait for TCI=0
10DC DB6F      432+??0013: IN      PRPF      ;Wait for TCI=1
10DE E602      433+      ANI      TCIF      ;Get task complete int,etc.
10E0 C2DC10    434+      JNZ      ??0013    ;Mask it
435      WAITT
10E3 DB6F      436+??0014: IN      PRPF      ;Get 91 Int status (END &/or BI)
10E5 E602      437+      ANI      TCIF      ;Save it in B for BI check later
10EA DB51      439 RECV1: IN      INT1      ;Check for EOS or EOI
10EC 47        440      MOV      B,A      ;Yes end--- go wait for BI
10ED E610      441      ANI      ENDMK     ;NO, retrieve status &
10EF C20511    442      JNZ      RECV2     ;check for BI
10F2 78        443      MOV      A,B      ;NO, go wait for either END or BI
10F3 E601      444      ANI      BIM      ;YES, BI--- get data
10F5 CAEA10    445      JZ      RECV1     ;Store it in buffer
10F8 DB50      446      IN      DIN      ;Increment buffer pointer
10FA 12        447      STAX     D      ;Decrement counter
10FB 13        448      INX     D      ;If count < > 0 go back & wait
10FC 0D        449      DCR     C      ;Else set error indicator
10FD C2EA10    450      JNZ      RECV1     ;And go take control
1100 0640      451      MVI      B,40H
1102 C31711    452      JMP      RECV5
453 ;
1105 78        454 RECV2: MOV      A,B      ;Retreive status
1106 E601      455 RECV3: ANI      BIM      ;Check for BI
1108 C21011    456      JNZ      RECV4     ;If BI then go input data
110B DB61      457      IN      INT1      ;Else wait for last BI
110D C30611    458      JMP      RECV3     ;In loop
1110 DB60      459 RECV4: IN      DIN      ;Get data byte
1112 12        460      STAX     D      ;Store it in buffer
1113 13        461      INX     D      ;Incr data pointer
1114 0D        462      DCR     C      ;Decrement count, but ignore it
1115 0600      463      MVI      B,0      ;Set normal completion indicators
464 ;
1117 3EFD      465 RECV5: MVI      A,TCSY    ;Take control synchronously
1119 D369      466      OUT      CMD92
467      WAITX      ;Wait for TCI=0 (7 tcy)
111B DB6F      468+??0015: IN      PRPF      ;Wait for TCI=1
111D E602      469+      ANI      TCIF      ;Get task complete int,etc.
111F C21B11    470+      JNZ      ??0015    ;Mask it
471      WAITT
1122 DB6F      472+??0016: IN      PRPF      ;Wait for task to be complete
1124 E602      473+      ANI      TCIF      ;Get task complete int,etc.
1126 CA2211    474+      JZ      ??0016    ;Mask it
475 ;
476 ;if timeout 3 is to be checked, the above WAITT should
477 ;be omitted & the appropriate code to look for TCI or
478 ;TOUF3 inserted here.
479 ;
1129 3E80      480      MVI      A,AXRA    ;Pattern to clear 91 END conditions
112B D365      481      OUT      AUXMD
112D 3E80      482      MVI      A,TON      ;This bit pattern already in "A"
112F D364      483      OUT      ADRMD     ;Output TON
1131 3E03      484      MVI      A,FNHSK    ;Finish handshake
1133 D365      485      OUT      AUXMD
486      CLRA
1135 AF        487+      XRA      A      ;A XOR A =0
1136 D365      488      OUT      AUXMD     ;Immediate execute PON-Reset LON
1138 78        489      MOV      A,B      ;Get completion character
1139 C9        490 RECV6: RET

```

# APPLICATIONS

```

491 ;
492 ;*****
493 ;       XFER ROUTINE
494 ;
495 ;
496 ;INPUTS:       HL device list pointer
497 ;              B EOS character
498 ;OUTPUTS:     None
499 ;CALLS:       None
500 ;DESTROYS:    A, HL, F
501 ;RETURNS:     A=0 normal, A < > 0 bad talker
502 ;
503 ;
504 ;NOTE:         XFER will not work if the talker
505 ;              uses EOI to terminate the transfer.
506 ;              Intel will be making hardware
507 ;              modifications to the 8291 that will
508 ;              correct this problem. Until that time,
509 ;              only EOS may be used without possible
510 ;              loss of the last data byte transferred.
511 XFER:  RANGE  40H,5EH,XFER4 ;Check for valid talker
512+                ;Checks for value in range
513+                ;branches to label if not
514+                ;in range. Falls through if
515+                ;lower <= ( (H)(L) ) <= upper.
516+                ;Get next byte.
113A 7E          517+      MOV     A,M
113B FE40        518+      CPI     40H
113D FABB11     519+      JM     XFER4
1140 FE5F        520+      CPI     5EH+1
1142 F2BB11     521+      JP     XFER4
1145 D360        522      OUT     DOUT ;Send it to GPIB
1147 23          523      INX     H ;Incr pointer
                    524      WAITO
1148 DB61        525+??0017: IN     INT1 ;Get Intl status
114A E602        526+      ANI     BOM ;Check for byte out
114C CA4811     527+      JZ     ??0017 ;If not, try again
114F 3E3F        528      MVI     A,UNL ;Universal unlisten
1151 D360        529      OUT     DOUT
                    530 XFER1: RANGE  20H,3EH,XFER2 ;Check for valid listener
                    531+                ;Checks for value in range
                    532+                ;branches to label if not
                    533+                ;in range. Falls through if
                    534+                ;lower <= ( (H)(L) ) <= upper.
                    535+                ;Get next byte.
1153 7E          536+      MOV     A,M
1154 FE20        537+      CPI     20H
1156 FA6C11     538+      JM     XFER2
1159 FE3F        539+      CPI     3EH+1
115B F26C11     540+      JP     XFER2
                    541      WAITO
115E DB61        542+??0018: IN     INT1 ;Get Intl status
1160 E602        543+      ANI     BOM ;Check for byte out
1162 CA5E11     544+      JZ     ??0018 ;If not, try again
1165 7E          545      MOV     A,M ;Get listener
1166 D360        546      OUT     DOUT
1168 23          547      INX     H ;Incr pointer
1169 C35311     548      JMP     XFER1 ;Loop until non-valid listener
                    549 XFER2: WAITO
116C DB61        550+??0019: IN     INT1 ;Get Intl status
116E E602        551+      ANI     BOM ;Check for byte out
1170 CA6C11     552+      JZ     ??0019 ;If not, try again
1173 3E87        553      MVI     A,AXRA+CAHCY+EDEOS ;Invisible handshake
1175 D365        554      OUT     AUXMD ;Continuous AH mode
1177 3E40        555      MVI     A,LON ;Listen only
1179 D364        556      OUT     ADRMD
                    557      CLRA
117B AF          558+      XRA     A ;A XOR A = 0
117C D365        559      OUT     AUXMD ;Immed. XEQ PON
117E 78          560      MOV     A,B ;Get EOS
117F D367        561      OUT     EOSR ;Output it to 91
1181 3EF6        562      MVI     A,GTSB ;Go to standby
1183 D369        563      OUT     CMD92

```

# APPLICATIONS

```

1185 DB6F      564      WAITX
1187 E602      565+??0020: IN      PRTF
1189 C28511    566+      ANI      TCIF
                    567+      JNZ      ??0020
                    568      WAITT      ;Wait for TCS
118C DB6F      569+??0021: IN      PRTF      ;Get task complete int,etc.
118E E602      570+      ANI      TCIF      ;Mask it
1190 CA8C11    571+      JZ      ??0021 ;Wait for task to be complete
1193 DB61      572 XFER3: IN      INT1      ;Get END status hit
1195 E610      573      ANI      ENDMK      ;Mask it
1197 CA9311    574      JZ      XFER3
119A 3EFD      575      MVI      A,TCSY      ;Take control synchronously
119C D369      576      OUT      CMD92
                    577      WAITX
119E DB6F      578+??0022: IN      PRTF
11A0 E602      579+      ANI      TCIF
11A2 C29E11    580+      JNZ      ??0022
                    581      WAITT      ;Wait for TCI
11A5 DB6F      582+??0023: IN      PRTF      ;Get task complete int,etc.
11A7 E602      583+      ANI      TCIF      ;Mask it
11A9 CAA511    584+      JZ      ??0023 ;Wait for task to be complete
11AC 3E80      585      MVI      A,AXRA      ;Not cont AH or END on EOS
11AE D365      586      OUT      AUXMD
11B0 3E03      587      MVI      A,FNHSK      ;Finish handshake
11B2 D365      588      OUT      AUXMD
11B4 3E80      589      MVI      A,TON      ;Talk only
11B6 D364      590      OUT      ADRMD
                    591      CLRA      ;Normal return A=0
11B8 AF        592+      XRA      A      ;A XOR A =0
11B9 D365      593      OUT      AUXMD      ;Immediate XEQ PON
11BB C9        594 XFER4: RET
                    595 ;
                    596 ;*****
                    597 ;
                    598 ;
                    599 ;          TRIGGER ROUTINE
                    600 ;
                    601 ;
                    602 ;INPUTS:          HL listener list pointer
                    603 ;OUTPUTS:         None
                    604 ;CALLS:           None
                    605 ;DESTROYS:        A, HL, F
                    606 ;
                    607 ;
11BC 3E3F      608 TRIG:  MVI      A,UNL      ;
11BE D360      609      OUT      DOUT      ;Send universal unlisten
                    610 TRIG1: RANGE  20H,3EH,TRIG2 ;Check for valid listen
                    611+      ;Checks for value in range
                    612+      ;branches to label if not
                    613+      ;in range. Falls through if
                    614+      ;lower <= ( (H)(L) ) <= upper.
                    615+      ;Get next byte.
11C0 7E        616+      MOV      A,M
11C1 FE20      617+      CPI      20H
11C3 FAD911    618+      JM      TRIG2
11C6 FE3F      619+      CPI      3EH+1
11C8 F2D911    620+      JP      TRIG2
                    621      WAITO      ;Wait for UNL to finish
11CB DB61      622+??0024: IN      INT1      ;Get Intl status
11CD E602      623+      ANI      BOM      ;Check for byte out
11CF CACB11    624+      JZ      ??0024 ;If not, try again
11D2 7E        625      MOV      A,M      ;Get listener
11D3 D360      626      OUT      DOUT      ;Send listener to GPIB
11D5 23        627      INX      H      ;Incr. pointer
11D6 C3C011    628      JMP      TRIG1      ;Loop until non-valid char
                    629 TRIG2: WAITO      ;Wait for last listen to finish
11D9 DB61      630+??0025: IN      INT1      ;Get Intl status
11DB E602      631+      ANI      BOM      ;Check for byte out
11DD CAD911    632+      JZ      ??0025 ;If not, try again
11E0 3E08      633      MVI      A,GET      ;Send group execute trigger
11E2 D360      634      OUT      DOUT      ;to all addressed listeners
                    635      WAITO
11E4 DB61      636+??0026: IN      INT1      ;Get Intl status
11E6 E602      637+      ANI      BOM      ;Check for byte out

```

# APPLICATIONS

```

11E8 CAE411      638+      JZ      ??0026 ;If not, try again
11EB C9          639      RET
640 ;
641 ;*****
642 ;
643 ;DEVICE CLEAR ROUTINE
644 ;
645 ;
646 ;
647 ;INPUTS:      HL listener pointer
648 ;OUTPUT:      None
649 ;CALLS:       None
650 ;DESTROYS:    A, HL, F
651 ;
11EC 3E3F        652 DCLR:  MVI  A,UNL
11EE D360        653      OUT  DOUT
654 DCLR1:  RANGE 20H,3EH,DCLR2
655+          ;Checks for value in range
656+          ;branches to label if not
657+          ;in range. Falls through if
658+          ;lower <= ( (H)(L) ) <= upper.
659+          ;Get next byte.
11F0 7E          660+      MOV  A,M
11F1 FE20        661+      CPI  20H
11F3 FA0912      662+      JM   DCLR2
11F6 FE3F        663+      CPI  3EH+1
11F8 F20912      664+      JP   DCLR2
665      WAITO
11FB DB61        666+??0027: IN  INT1 ;Get Intl status
11FD E602        667+      ANI  BOM ;Check for byte out
11FF CAFB11      668+      JZ   ??0027 ;If not, try again
1202 7E          669      MOV  A,M
1203 D360        670      OUT  DOUT ;Send listener to GPIB
1205 23          671      INX  H
1206 C3F011      672      JMP  DCLR1
673 DCLR2:  WAITO
1209 DB61        674+??0028: IN  INT1 ;Get Intl status
120B E602        675+      ANI  BOM ;Check for byte out
120D CA0912      676+      JZ   ??0028 ;If not, try again
1210 3E04        677      MVI  A,SDC ;Send device clear
1212 D360        678      OUT  DOUT ;To all addressed listeners
679      WAITO
1214 DB61        680+??0029: IN  INT1 ;Get Intl status
1216 E602        681+      ANI  BOM ;Check for byte out
1218 CA1412      682+      JZ   ??0029 ;If not, try again
121B C9          683      RET
684 ;
685 ;*****
686 ;
687 ;      SERIAL POLL ROUTINE
688 ;
689 ;INPUTS:      HL talker list pointer
690 ;      DE status buffer pointer
691 ;OUTPUTS:     Fills buffer pointed to by DE
692 ;CALLS:       None
693 ;DESTROYS:    A, BC, DE, HL, F
694 ;
121C 3E3F        695 SPOL:  MVI  A,UNL ;Universal unlisten
121E D360        696      OUT  DOUT
697      WAITO
1220 DB61        698+??0030: IN  INT1 ;Get Intl status
1222 E602        699+      ANI  BOM ;Check for byte out
1224 CA2012      700+      JZ   ??0030 ;If not, try again
1227 3E21        701      MVI  A,MLA ;My listen address
1229 D360        702      OUT  DOUT
703      WAITO
122B DB61        704+??0031: IN  INT1 ;Get Intl status
122D E602        705+      ANI  BOM ;Check for byte out
122F CA2B12      706+      JZ   ??0031 ;If not, try again
1232 3E18        707      MVI  A,SPE ;Serial poll enable
1234 D360        708      OUT  DOUT ;To be formal about it
709      WAITO
1236 DB61        710+??0032: IN  INT1 ;Get Intl status

```

APPLICATIONS

```

1238 E602      711+      ANI      BOM      ;Check for byte out
123A CA3512    712+      JZ       ??0032 ;If not, try again
              713 SPOL1:  RANGE    40H,5EH,SPOL2 ;Check for valid talker
              714+      ;Checks for value in range
              715+      ;branches to label if not
              716+      ;in range. Falls through if
              717+      ;lower <= ( (H)(L) ) <= upper.
              718+      ;Get next byte.

123D 7E        719+      MOV      A,M
123E FE40      720+      CPI      40H
1240 FA9412    721+      JM       SPOL2
1243 FE5F      722+      CPI      5EH+1
1245 F29412    723+      JP       SPOL2
1248 7E        724      MOV      A,M      ;Get talker
1249 D360      725      OUT     DOUT      ;Send to GPIB
124B 23        726      INX     H         ;Incr talker list pointer
124C 3E40      727      MVI     A,LOH    ;Listen only
124E D364      728      OUT     ADRMD
              729      WAITO    ;Wait for talk address to complete
1250 DB61      730+??0033: IN     INT1  ;Get Intl status
1252 E602      731+      ANI     BOM      ;Check for byte out
1254 CA5012    732+      JZ      ??0033  ;If not, try again
              733      CLRA    ;Pattern for immediate XEQ PON
1257 AF        734+      XRA     A        ;A XOR A =0
1258 D365      735      OUT     AUXMD
125A 3EF6      736      MVI     A,GTSB   ;Goto standby
125C D369      737      OUT     CMD92
              738      WAITX    ;Wait for TCI false
125E DB6F      739+??0034: IN     PRTE  ;
1260 E602      740+      ANI     TCIF
1262 C25E12    741+      JNZ     ??0034
              742      WAITT    ;Wait for TCI
1265 DB6F      743+??0035: IN     PRTE  ;Get task complete int,etc.
1267 E602      744+      ANI     TCIF
1269 CA5512    745+      JZ      ??0035  ;Mask it
              746      WAITI    ;Wait for task to be complete
126C DB61      747+??0036: IN     INT1  ;Wait for status byte input
126E 47        748+      MOV     B,A     ;Get INT1 status
126F E601      749+      ANI     B,A     ;Save status in B
1271 CA6C12    750+      JZ      ??0036  ;Check for byte in
1274 3EFD      751      MVI     A,TCSY  ;If not, just try again
1276 D359      752      OUT     A,TCSY ;Take control sync
              753      OUT     CMD92
              754+??0037: IN     PRTE  ;Wait for TCI false
1278 DB6F      755+      ANI     TCIF
127A E602      756+      JNZ     ??0037
127C C27812    757      WAITT    ;Wait for TCI
127F DB6F      758+??0038: IN     PRTE  ;Get task complete int,etc.
1281 E602      759+      ANI     TCIF
1283 CA7F12    760+      JZ      ??0038  ;Mask it
1286 DB60      761      IN      DIN     ;Wait for task to be complete
1288 12        762      STAX   D        ;Get serial poll status byte
1289 13        763      INX     D        ;Store it in buffer
128A 3E80      764      MVI     A,TON   ;Incr pointer
128C D364      765      OUT     A,TON   ;Talk only for controller
              766      OUT     ADRMD
              767+      CLRA    ;
128E AF        767+      XRA     A        ;A XOR A =0
128F D365      768      OUT     AUXMD   ;Immeditate XEQ PON
              769      CLRA    ;CLR LA
1291 C33D12    770      JMP     SPOL1    ;Go on to next device on list
              771 ;
1294 3E19      772 SPOL2:  MVI     A,SPD ;Serial poll disable
1296 D360      773      OUT     DOUT    ;We know BO was set (WAITO above)
              774      WAITO    ;
1298 DB61      775+??0039: IN     INT1  ;Get Intl status
129A E602      776+      ANI     BOM      ;Check for byte out
129C CA9812    777+      JZ      ??0039  ;If not, try again
              778      CLRA    ;
129F AF        779+      XRA     A        ;A XOR A =0
12A0 D365      780      OUT     AUXMD   ;Immeditate XEQ PON to clear LA
12A2 C9        781      RET
              782 ;
              783 ;*****
              784 ;

```

# APPLICATIONS

```

785 ;          PARALLEL POLL ENABLE ROUTINE
786 ;
787 ;INPUTS:      HL listener list pointer
788 ;          DE configuration byte pointer
789 ;OUTPUTS:     None
790 ;CALLS:       None
791 ;DESTROYS:    A, DE, HL, F
792 ;
793 ;
12A3 3E3F      794 PPEN:  MVI  A,UNL  ;Universal unlisten
12A5 D360      795          OUT  DOUT
              796 PPEN1: RANGE 20H,3EH,PPEN2 ;Check for valid listener
              797+          ;Checks for value in range
              798+          ;branches to label if not
              799+          ;in range. Falls through if
              800+          ;lower <= ( (H)(L) ) <= upper.
              801+          ;Get next byte.
12A7 7E        802+          MOV  A,M
12A8 FE20      803+          CPI  20H
12AA FAD812    804+          JM   PPEN2
12AD FE3F      805+          CPI  3EH+1
12AF F2D812    806+          JP   PPEN2
              807          WAITO          ;Valid wait 91 data out reg
12B2 DB61      808+??0040: IN  INT1  ;Get Intl status
12B4 E602      809+          ANI  BOM    ;Check for byte out
12B6 CAB212    810+          JZ   ??0040 ;If not, try again
12B9 7E        811          MOV  A,M    ;Get listener
12BA D350      812          OUT  DOUT
              813          WAITO
12BC DB61      814+??0041: IN  INT1  ;Get Intl status
12BE E602      815+          ANI  BOM    ;Check for byte out
12C0 CAB012    816+          JZ   ??0041 ;If not, try again
12C3 3E05      817          MVI  A,PPC  ;Parallel poll configure
12C5 0350      818          OUT  DOUT
              819          WAITO
12C7 DB61      820+??0042: IN  INT1  ;Get Intl status
12C9 E602      821+          ANI  BOM    ;Check for byte out
12CB CAC712    822+          JZ   ??0042 ;If not, try again
12CE 1A        823          LDAX D      ;Get matching configuration byte
12CF F660      824          ORI  PPE    ;Merge with parallel poll enable
12D1 D360      825          OUT  DOUT
12D3 23        826          INX  H      ;Incr pointers
12D4 13        827          INX  D
12D5 C3A712    828          JMP  PPEN1  ;Loop until invalid listener char
              829 PPEN2: WAITO
12D8 DB61      830+??0043: IN  INT1  ;Get Intl status
12DA E602      831+          ANI  BOM    ;Check for byte out
12DC CAD812    832+          JZ   ??0043 ;If not, try again
12DF C9        833          RET
              834 ;
              835 ;PARALLEL POLL DISABLE ROUTINE
              836 ;
              837 ;INPUTS:      HL listener list pointer
              838 ;OUTPUTS:     None
              839 ;CALLS:       None
              840 ;DESTROYS:    A, HL, F
              841 ;
12E0 3E3F      842 PPDS:  MVI  A,UNL  ;Universal unlisten
12E2 D360      843          OUT  DOUT
              844 PPDS1: RANGE 20H,3EH,PPDS2 ;Check for valid listener
              845+          ;Checks for value in range
              846+          ;branches to label if not
              847+          ;in range. Falls through if
              848+          ;lower <= ( (H)(L) ) <= upper.
              849+          ;Get next byte.
12E4 7E        850+          MOV  A,M
12E5 FE20      851+          CPI  20H
12E7 FAFD12    852+          JM   PPDS2
12EA FE3F      853+          CPI  3EH+1
12EC F2FD12    854+          JP   PPDS2
              855          WAITO
12EF DB61      856+??0044: IN  INT1  ;Get Intl status
12F1 E602      857+          ANI  BOM    ;Check for byte out
12F3 CAEF12    858+          JZ   ??0044 ;If not, try again

```



```

12F6 7E          859      MOV      A,M      ;Get listener
12F7 D360       860      OUT      DOUT
12F9 23         861      INX      H          ;Incr pointer
12FA C3E412     862      JMP      PPDS1     ;Loop until invalid listener
                863 PPDS2: WAITO
12FD DB61       864+??0045: IN      INT1     ;Get Intl status
12FF E602       865+     ANI      BOM      ;Check for byte out
1301 CAFD12     866+     JZ       ??0045   ;If not, try again
1304 3E05       867      MVI      A,PPC   ;Parallel poll configure
1306 D350       868      OUT      DOUT
                869      WAITO
1308 DB61       870+??0046: IN      INT1     ;Get Intl status
130A E602       871+     ANI      BOM      ;Check for byte out
130C CA0813     872+     JZ       ??0046   ;If not, try again
130F 3E70       873      MVI      A,PPD   ;Parallel poll disable
1311 D360       874      OUT      DOUT
                875      WAITO
1313 DB61       876+??0047: IN      INT1     ;Get Intl status
1315 E602       877+     ANI      BOM      ;Check for byte out
1317 CA1313     878+     JZ       ??0047   ;If not, try again
131A C9         879      RET
                880 ;
                881 ;          PARALLEL POLL UNCONFIGURE ALL ROUTINE
                882 ;
                883 ;
131B 3E15       884 ;INPUTS:      None
131D D360       885 ;OUTPUTS:      None
                886 ;CALLS:        None
                887 ;DESTROYS:     A, F
                888 ;
131F DB61       889 PPUN:  MVI      A,PPU   ;Parallel poll unconfigure
1321 E602       890      OUT      DOUT
                891      WAITO
1323 CA1F13     892+??0048: IN      INT1     ;Get Intl status
1326 C9         893+     ANI      BOM      ;Check for byte out
                894+     JZ       ??0048   ;If not, try again
                895      RET
                896 ;
                897 ;*****
                898 ;
1327 3E40       899 ;CONDUCT A PARALLEL POLL
1329 D364       900 ;
                901 ;
                902 ;INPUTS:      None
                903 ;OUTPUTS:      None
                904 ;CALLS:        None
                905 ;DESTROYS:     A, B, F
                906 ;RETURNS:      A= parallel poll status byte
                907 ;
132B AF         908 PPOL:  MVI      A,LON   ;Listen only
132C D365       909      OUT      ADRMD
                910      CLRA      ;Immediate XEQ PON
                911+     XRA      A          ;A XOR A =0
132E 3EF5       912      OUT      AUXMD     ;Reset TON
1330 D369       913      MVI      A,EXPP   ;Execute parallel poll
                914      OUT      CMD92
                915      WAITI     ;Wait for completion= BI on 91
1332 DB61       916+??0049: IN      INT1     ;Get INT1 status
1334 47         917+     MOV      B,A        ;Save status in B
1335 E601       918+     ANI      BIM      ;Check for byte in
1337 CA3213     919+     JZ       ??0049   ;If not, just try again
133A 3E80       920      MVI      A,TON   ;Talk only
133C D364       921      OUT      ADRMD
                922      CLRA      ;Immediate XEQ PON
133E AF         923+     XRA      A          ;A XOR A =0
133F D365       924      OUT      AUXMD     ;Reset LON
1341 DB60       925      IN       DIN      ;Get PP byte
1343 C9         926      RET
                927 ;
                928 ;*****
1343 C9         929 ;PASS CONTROL ROUTINE
                930 ;
                931 ;INPUTS:      HL pointer to talker
                932 ;OUTPUTS:      None

```

# APPLICATIONS

```

933 ;CALLS:           None
934 ;DESTROYS:       A, HL, F
935 PCTL:   RANGE    40H,5EH,PCTL1 ;Is it a valid talker ?
936+                ;Checks for value in range
937+                ;branches to label if not
938+                ;in range. Falls through if
939+                ;lower <= ( (H)(L) ) <= upper.
940+                ;Get next byte.
1344 7E             941+      MOV      A,M
1345 FE40           942+      CPI      40H
1347 FA8A13        943+      JM      PCTL1
134A FE5F           944+      CPI      5EH+1
134C F28A13        945+      JP      PCTL1
134F FE41           946+      CPI      MTA      ;Is it my talker address
1351 CA8A13        947+      JZ      PCTL1      ;Yes, just return
1354 D360           948+      OUT     DOUT      ;Send on GPIB
                949+      WAITO
1356 DB61           950+??0050: IN     INT1      ;Get Intl status
1358 E602           951+      ANI     BOM      ;Check for byte out
135A CA5613        952+      JZ      ??0050 ;If not, try again
135D 3E09           953+      MVI     A,TCT   ;Take control message
135F D360           954+      OUT     DOUT
                955+      WAITO
1361 DB61           956+??0051: IN     INT1      ;Get Intl status
1363 E602           957+      ANI     BOM      ;Check for byte out
1365 CA6113        958+      JZ      ??0051 ;If not, try again
1368 3E01           959+      MVI     A,MODEL ;Not talk only or listen only
136A D364           960+      OUT     ADRMD   ;Enable 91 address mode 1
                961+      CLRA
136C AF             962+      XRA      A      ;A XOR A =0
136D D365           963+      OUT     AUXMD   ;Immediate XEQ PON
136F 3E01           964+      MVI     A,MDA   ;My device address
1371 D366           965+      OUT     ADR01   ;enabled to talk and listen
1373 3EA1           966+      MVI     A,AXRB+CPTEN ;Command pass thru enable
1375 D365           967+      OUT     AUXMD
                968 ;*****optional PP configuration goes here*****
1377 3EF1           969+      MVI     A,GIDL  ;92 go idle command
1379 D369           970+      OUT     CMD92
                971+      WAITX
137B DB6F           972+??0052: IN     PRTF
137D E602           973+      ANI     TCIF
137F C27B13        974+      JNZ     ??0052
                975+      WAITT      ;Wait for TCI
1382 DB6F           976+??0053: IN     PRTF      ;Get task complete int,etc.
1384 E602           977+      ANI     TCIF      ;Mask it
1386 CA8213        978+      JZ      ??0053 ;Wait for task to be complete
1389 23             979+      INX     H
138A C9             980 PCTL1:  RET
                981 ;
                982 ;
                983 ;*****
                984 ;
                985 ;RECEIVE CONTROL ROUTINE
                986 ;
                987 ;INPUTS:           None
                988 ;OUTPUTS:          None
                989 ;CALLS:             None
                990 ;DESTROYS:         A, F
                991 ;RETURNS:          0= invalid (not take control to us or CPT bit not on)
                992 ;                    < > 0 = valid take control-- 92 will now be in control
                993 ;NOTE:             THIS CODE MUST BE TIGHTLY INTEGRATED INTO ANY USER
                994 ;                    SOFTWARE THAT FUNCTIONS WITH THE 8291 AS A DEVICE.
                995 ;                    NORMALLY SOME ADVANCE WARNING OF IMPENDING PASS
                996 ;                    CONTROL SHOULD BE GIVEN TO US BY THE CONTROLLER
                997 ;                    WITH OTHER USEFUL INFO. THIS PROTOCOL IS SITUATION
                998 ;                    SPECIFIC AND WILL NOT BE COVERED HERE.
                999 ;
                1000 ;
138B DB61           1001 RCTL:  IN     INT1      ;Get INT1 req (i.e. CPT etc.)
138D E680           1002      ANI     CPT      ;Is command pass thru on ?
138F CACF13        1003      JZ      RCTL2   ;No, invalid-- go return
1392 DB65           1004      IN     CPTRG   ;Get command
1394 FE09           1005      CPI      TCT      ;Is it take control ?

```

# APPLICATIONS

```

1396 C2CA13      1006      JNZ      RCTL1      ;No, go return invalid
1399 DB64        1007      IN       ADRST      ;Get address status
139B E602        1008      ANI     TA         ;Is TA on ?
139D CACA13      1009      JZ      RCTL1      ;No -- go return invalid
13A0 3E60        1010      MVI     A,DTDL1    ;Disable talker listener
13A2 D366        1011      OUT    ADR01
13A4 3E80        1012      MVI     A,TON      ;Talk only
13A6 D364        1013      OUT    ADRMD
                1014      CLRA
13A8 AF          1015+     XRA     A          ;A XOR A =0
13A9 D361        1016      OUT    INT1       ;Mask off INT bits
13AB D362        1017      OUT    INT2
13AD D365        1018      OUT    AUXMD
13AF 3EFA        1019      MVI     A,TCNTR    ;Take (receive) control 92 command
13B1 D369        1020      OUT    CMD92
13B3 3E0F        1021      MVI     A,VSCMD    ;Valid command pattern for 91
13B5 D365        1022      OUT    AUXMD
1023 ;***** optional TOUT1 check could be put here *****
1024      WAITX
13B7 DB6F        1025+??0054: IN      PRTF
13B9 E602        1026+     ANI     TCIF
13BB C2B713      1027+     JNZ     ??0054
                1028      WAITT          ;Wait for TCI
13BE DB6F        1029+??0055: IN      PRTF          ;Get task complete int,etc.
13C0 E602        1030+     ANI     TCIF          ;Mask it
13C2 CABE13      1031+     JZ      ??0055       ;Wait for task to be complete
13C5 3E09        1032      MVI     A,TCF       ;Valid return pattern
13C7 C3CF13      1033      JMP     RCTL2       ;Only one return per routine
13CA 3E0F        1034 RCTL1: MVI     A,VSCMD ;Acknowledge CPT
13CC D365        1035      OUT    AUXMD
                1036      CLRA          ;Error return pattern
13CE AF          1037+     XRA     A          ;A XOR A =0
13CF C9          1038 RCTL2: RET
                1039 ;
1040 ;*****
1041 ;
                1042 ;          SRQ ROUTINE
1043 ;
1044 ;INPUTS:          None
1045 ;OUTPUTS:        None
1046 ;CALLS:          None
1047 ;RETURNS:        A= 0 no SRQ
1048 ;                A < > 0 SRQ occurred
1049 ;
1050 ;
13D0 DB69        1051 SRQD:  IN      INTST   ;Get 92's INTRO status
13D2 E620        1052      ANI     SRQBT     ;Mask off SRQ
13D4 CAE213      1053      JZ      SRQD2     ;Not set--- go return
13D7 F60B        1054      ORI     IACK      ;Set--- must clear it with IACK
13D9 D369        1055      OUT    CMD92
13DB DB69        1056 SRQD1: IN      INTST   ;Get IBF
13DD E602        1057      ANI     IBFBT    ;Mask it
13DF CADB13      1058      JZ      SRQD1     ;wait if not set
13E2 C9          1059 SRQD2: RET
                1060 ;
1061 ;*****
1062 ;
1063 ;REMOTE ENABLE ROUTINE
1064 ;
1065 ;INPUTS:          None
1066 ;OUTPUTS:        None
1067 ;CALLS:          NONE
1068 ;DESTROYS:       A, F
1069 ;
13E3 3EF8        1070 REME:  MVI     A,SREM
13E5 D369        1071      OUT    CMD92     ;92 asserts remote enable
                1072      WAITX          ;Wait for TCI = 0
13E7 DB6F        1073+??0056: IN      PRTF
13E9 E602        1074+     ANI     TCIF
13EB C2E713      1075+     JNZ     ??0056
                1076      WAITT          ;Wait for TCI
13EE DB6F        1077+??0057: IN      PRTF          ;Get task complete int,etc.
13F0 E602        1078+     ANI     TCIF          ;Mask it
13F2 CAE13      1079+     JZ      ??0057       ;Wait for task to be complete

```

# APPLICATIONS

```

13F5 C9      1080      RET
1081 ;
1082 ;*****
1083 ;
1084 ;LOCAL ROUTINE
1085 ;
1086 ;
1087 ;INPUTS:      None
1088 ;OUTPUTS:     None
1089 ;CALLS:       None
1090 ;DESTROYS:    A, F
1091 ;
13F6 3EF7    1092 LOCL:  MVI    A,SLOC
13F8 D369    1093      OUT    CMD92 ;92 stops asserting remote enable
1094      WAITX   ;Wait for TCI =0
13FA DB6F    1095+??0058: IN     PRTF
13FC E602    1096+      ANI    TCIF
13FE C2FA13  1097+      JNZ    ??0058
1098      WAITT   ;Wait for TCI
1401 DB6F    1099+??0059: IN     PRTF ;Get task complete int,etc.
1403 E602    1100+      ANI    TCIF ;Mask it
1405 CA0114  1101+      JZ     ??0059 ;Wait for task to be complete
1408 C9      1102      RET
1103 ;
1104 ;*****
1105 ;
1106 ;INTERFACE CLEAR / ABORT ROUTINE
1107 ;
1108 ;
1109 ;INPUTS:       None
1110 ;OUTPUTS:      None
1111 ;CALLS:        None
1112 ;DESTROYS:    A, F
1113 ;
1114 ;
1409 3EF9    1115 IFCL:  MVI    A,ABORT
140B D369    1116      OUT    CMD92 ;Send IFC
1117      WAITX   ;Wait for TCI =0
140D DB6F    1118+??0060: IN     PRTF
140F E602    1119+      ANI    TCIF
1411 C20D14  1120+      JNZ    ??0060
1121      WAITT   ;Wait for TCI
1414 DB6F    1122+??0061: IN     PRTF ;Get task complete int,etc.
1416 E602    1123+      ANI    TCIF ;Mask it
1418 CA1414  1124+      JZ     ??0061 ;Wait for task to be complete
1125 ;Delete both WAITX & WAITT if this routine
1126 ;is to be called while the 9292 is
1127 ;Controller-in-Charge. If not C.I.C. then
1128 ;TCI is set, else nothing is set (IFC is sent)
1129 ;and the WAIT'S will hang forever
141B C9      1130      RET
1132 ;

```

# APPLICATIONS

```

1133 ;APPLICATION EXAMPLE CODE FOR 8085
1134 ;
0032 1135 FGDNL EQU '2' ;Func gen device num "2" ASCII,1stn
0031 1136 FCDNL EQU '1' ;Freq ctr device num "1" ASCII,1stn
0051 1137 FCDNT EQU 'Q' ;Freq ctr talk address
000D 1138 CR EQU 0DH ;ASCII carriage return
000A 1139 LF EQU 0AH ;ASCII line feed
00FF 1140 LEND EQU 0FFH ;List end for Talk/Listen lists
0040 1141 SRQM EQU 40H ;Bit indicating device sent SRQ
1142 ;
141C 46553146 1143 FGDATA: DB 'FU1FR37KHAM2VO',CR ;Data to set up func. gen
1420 5233374B
1424 48414D32
1428 564F
142A 0D
000F 1144 LIM1 EQU 15 ;Buffer length
142B 50463447 1145 FCDATA: DB 'PF4G7T' ;Data to set up freq ctr
142F 3754
0006 1146 LIM2 EQU 6 ;Buffer length
1431 31 1147 LL1: DB FCDNL,LEND ;Listen list for freq ctr
1432 FF
1433 32 1148 LL2: DB FGDNL,LEND ;Listen list for func. gen
1434 FF
1435 51 1149 TL1: DB FCDNT,LEND ;Talk list for freq ctr
1436 FF

1150 ;
1437 060D 1151 ;SETUP FUNCTION GENERATOR
1152 MVI B,CR ;EOS
1439 0E0F 1153 MVI C,LIM1 ;Count
143B 111C14 1154 LXI D,FGDATA ;Data pointer
143E 213314 1155 LXI H,LL2 ;Listen list pointer
1441 CD1C10 1156 CALL SEND
1157 ;
1158 ;SETUP FREQ COUNTER
1159 ;
1444 0554 1160 MVI B,'T' ;EOS
1446 0E06 1161 MVI C,LIM2 ;Count
1448 112B14 1162 LXI D,FCDATA ;Data pointer
144B 213114 1163 LXI H,LL1 ;Listen list pointer
144E CD1C10 1164 CALL SEND
1165 ;
1166 ;WAIT FOR SRQ FROM FREQ CTR
1167 ;
1451 CDD013 1168 LOOP: CALL SRQD ;Has SRQ occurred ?
1454 CA5114 1169 JZ LOOP ;No, wait for it
1170 ;
1171 ;SERIAL POLL TO CLEAR SRQ
1172 ;
1457 11003C 1173 LXI D,SPBYTE ;Buffer pointer
145A 213514 1174 LXI H,TL1 ;Talk list pointer
145D CD1C12 1175 CALL SPOL
1460 1B 1176 DCX D ;Backup buffer pointer to ctr byte
1461 1A 1177 LDAX D ;Get status byte
1462 E640 1178 ANI SRQM ;Did ctr assert SRQ ?
1464 CA7714 1179 JZ ERROR ;Ctr should have said yes
1180 ;
1181 ;RECEIVE READING FROM COUNTER
1182 ;
1467 060A 1183 MVI B,LF ;EOS
1469 0E11 1184 MVI C,LIM3 ;Count
146B 213514 1185 LXI H,TL1 ;Talk list pointer
146E 11013C 1186 LXI D,FCDATI ;Data in buffer pointer
1471 CD9F10 1187 CALL RECV
1474 C27714 1188 JNZ ERROR
1189 ;
1190 ;***** rest of user processing goes here *****
1191 ;
1192 ;
1477 00 1193 ERROR: NOP ;User dependant error handling
1194 ; ETC.
3C00 1195 ORG 3C00H
3C00 1196 SPBYTE: DS 1 ;Location for serial poll byte
0011 1197 LIM3 EQU 17 ;Max freq counter input

```



APPENDIX B

TEST CASES FOR THE SOFTWARE DRIVERS

The following test cases were used to exercise the software routines and to check their action. To provide another device/controller on the GPIB a ZT488 GPIB Analyzer was used. This analyzer

acted as a talker, listener or another controller as needed to execute the tests. The sequence of outputs are shown with each test. All numbers are hexadecimal.

SEND TEST CASES

B = 44	44	44
C = 30	2	0
DE = 3E80	3E80	3E80
HL = 3E70	3E70	3E70
3E70: 20 30 3E 3F		
3E80: 11 44		
GPIB output:	41 ATN	41 ATN
	3F ATN	3F ATN
	20 ATN	20 ATN
	30 ATN	30 ATN
	3E ATN	3E ATN
	11	
	44 EOI	
Ending B = 44	44	44
Ending C = 2E	0	0
Ending DE = 3E82	3E82	3E80
Ending HL = 3E73	3E73	3E73

RECEIVE TEST CASES

B = 44	44	44	44	44	44	44
C = 30	30	30	30	4	4	0=256
DE = 3E80	3E80	3E80	3E80	3E80	3E80	3E80
HL = 3E70	3E70	3E70	3E70	3E70	3E70	3E70
3E70: 40	50	5E	5F	40	40	40
GPIB output:	40 ATN	50 ATN	5E ATN	40 ATN	40 ATN	40 ATN
	3F ATN	3F ATN	3F ATN	3F ATN	3F ATN	3F ATN
	21 ATN	21 ATN	21 ATN	21 ATN	21 ATN	21 ATN
ZT488 Data	1	1	1	1	11	1
In	2	2	2	2	22	2
	3	3	3	3	33	3
	4	4	44,EOI	4	44	44
	44	5,EOI				
Ending A = 0	0	0	5F	40	0	0
Ending B = 0	0	0	44	40	0	0
Ending C = 2B	2B	2C	30	0	0	FC
Ending DE = 3E85	3E85	3E84	3E80	3E84	3E84	3E84
Ending HL = 3E71	3E71	3E71	3E70	3E71	3E71	3E71

SERIAL POLL TEST CASES

C = 30	C = 30
DE = 3E80	DE = 3E80
HL = 3E70	HL = 3E70
3E70: 40	3E70: 5F
50	GPIB output: 3F ATN
5E	21 ATN
5F	18 ATN





GPIB output: 3F ATN 3F ATN  
20 ATN 05 ATN  
30 ATN 70 ATN  
3E ATN  
05 ATN  
70 ATN

Ending HL = 3E73 3E70

**PARALLEL POLL UNCONFIGURE TEST CASE**

GPIB output: 15 ATN

**PARALLEL POLL TEST CASES**

Set DIO #	1	2	3	4	5	6	7	8	None
Ending A	1	2	4	8	10	20	40	80	0

**SRQ TEST**

	Set SRQ momentarily	Reset SRQ
Ending A	= 02	00

**TRIGGER TEST**

HL = 3E70  
DE = 3E80  
BC = 4430  
3E70: 20 30 3E 3F  
GPIB output: 3F ATN  
20 ATN  
30 ATN  
3E ATN  
08 ATN  
Ending HL = 3E73  
DE = 3E80  
BC = 4430

**DEVICE CLEAR TEST**

HL = 3E70  
DE = 3E80  
BC = 4430  
3E70: 20 30 3E 3F  
GPIB output: 3F ATN  
20 ATN  
30 ATN  
3E ATN  
14 ATN  
Ending HL = 3E73  
DE = 3E80  
RC = 4430

# APPLICATIONS

---

## XFER TEST

B = 44  
HL = 3E70  
3E70: 40 20 30 3E 3F  
GPIB output: 40 ATN  
3F ATN  
20 ATN  
30 ATN  
3E ATN  
GPIB input: 0  
1  
2  
3  
44  
Ending A = 0  
B = 44  
HL = 3E74

## APPLICATION EXAMPLE

### GPIB OUTPUT/INPUT

GPIB output: 41 ATN  
3F ATN  
32 ATN  
46  
55  
31  
46  
52  
33  
37  
4B  
48  
41  
4D  
32  
56  
4F  
0D EOI  
41 ATN  
3F ATN  
31 ATN  
50  
46  
34  
47  
37  
54 EOI  
GPIB input: SRQ  
GPIB output: 3F ATN  
21 ATN  
18 ATN  
51 ATN  
GPIB input: 40 SRQ  
GPIB output: 19 ATN  
51 ATN

# APPLICATIONS

GPIB input: 3F ATN  
 21 ATN  
 20  
 2B  
 20  
 20  
 33  
 37  
 30  
 30  
 30  
 2E  
 30  
 45  
 2B  
 30  
 0D  
 0A  
 GPIB output: XX ATN

## APPENDIX C

### REMOTE MESSAGE CODING

Mnemonic	Message Name	T y p e	C l a s s	D i s t r i b u t i o n	Bus Signal Line(s) and Coding That Asserts the True Value of the Message												
					8	7	6	5	4	3	2	1	VDC	NI	Q	CN	
ACG	addressed command group	M	AC	Y	0	0	0	X	X	X	X	XXX	1	X	X	X	X
ATN	attention	U	UC	X	X	X	X	X	X	X	X	XXX	1	X	X	X	X
DAB	data byte (Notes 1, 9)	M	DD	D	D	D	D	D	D	D	D	XXX	0	X	X	X	X
DAC	data accepted	U	HS	X	X	X	X	X	X	X	X	XX	0	X	X	X	X
DAV	data valid	U	HS	X	X	X	X	X	X	X	X	1XX	X	X	X	X	X
DCL	device clear	M	UC	Y	0	0	1	0	1	0	0	XXX	1	X	X	X	X
END	end	U	ST	X	X	X	X	X	X	X	X	XXX	0	1	X	X	X
EOS	end of string (Notes 2, 9)	M	DD	E	E	E	E	E	E	E	E	XXX	0	X	X	X	X
GET	group execute trigger	M	AC	Y	0	0	0	1	0	0	0	XXX	1	X	X	X	X
GTL	go to local	M	AC	Y	0	0	0	0	0	0	0	1	XXX	1	X	X	X
IDY	identify	U	UC	X	X	X	X	X	X	X	X	XXX	X	1	X	X	X
IFC	interface clear	U	UC	X	X	X	X	X	X	X	X	XXX	X	X	X	1	X
LAG	listen address group	M	AD	Y	0	1	X	X	X	X	X	XXX	1	X	X	X	X
LLO	local lock out	M	UC	Y	0	0	1	0	0	0	1	XXX	1	X	X	X	X
MLA	my listen address (Note 3)	M	AD	Y	0	1	L	L	L	L	L	XXX	1	X	X	X	X
MTA	my talk address (Note 4)	M	AD	Y	1	0	T	T	T	T	T	XXX	1	X	X	X	X
MSA	my secondary address (Note 5)	M	SE	Y	1	1	S	S	S	S	S	XXX	1	X	X	X	X

# APPLICATIONS

Mnemonic	Message Name	T y p e	C l a s s	D l i n e	Bus Signal Line(s) and Coding That Asserts the True Value of the Message															
					8	7	6	5	4	3	2	1	VDC	N	I	Q	C	N		
NUL	null byte	M	DD	0 0 0 0 0 0 0 0	XXX	X	X	X	X	X	X	X	X	X	X	X	X	X		
OSA	other secondary address	M	SE	(OSA = SCG ^ MSA)																
OTA	other talk address	M	AD	(OTA = TAG ^ MTA)																
PCG	primary command group	M	—	(PCG = ACG v UCG v LAG v TAG)																
PPC	parallel poll configure	M	AC	Y 0 0 0 0 1 0 1	XXX	1	X	X	X	X	X	X	X	X	X	X	X	X		
PPE	parallel poll enable (Note 6)	M	SE	Y 1 1 0 S P P P 3 2 1	XXX	1	X	X	X	X	X	X	X	X	X	X	X	X		
PPD	parallel poll disable (Note 7)	M	SE	Y 1 1 1 D D D D 4 3 2 1	XXX	1	X	X	X	X	X	X	X	X	X	X	X	X		
PPR1	parallel poll response 1	U	ST	X X X X X X X X	XXX	1	1	X	X	X	X	X	X	X	X	X	X	X		
PPR2	parallel poll response 2		ST	X X X X X X X 1	XXX	1	1	X	X	X	X	X	X	X	X	X	X	X		
PPR3	parallel poll response 3		ST	X X X X X 1 X X	XXX	1	1	X	X	X	X	X	X	X	X	X	X	X		
PPR4	parallel poll response 4		ST	X X X X 1 X X X	XXX	1	1	X	X	X	X	X	X	X	X	X	X	X		
PPR5	parallel poll response 5		ST	X X X 1 X X X X	XXX	1	1	X	X	X	X	X	X	X	X	X	X	X		
PPR6	parallel poll response 6		ST	X X 1 X X X X X	XXX	1	1	X	X	X	X	X	X	X	X	X	X	X		
PPR7	parallel poll response 7		ST	X 1 X X X X X X	XXX	1	1	X	X	X	X	X	X	X	X	X	X	X		
PPR8	parallel poll response 8		ST	1 X X X X X X X	XXX	1	1	X	X	X	X	X	X	X	X	X	X	X		
PPU	parallel poll unconfigure	M	UC	Y 0 0 1 0 1 0 1	XXX	1	X	X	X	X	X	X	X	X	X	X	X			
REN	remote enable	U	UC	X X X X X X X X	XXX	X	X	X	X	X	X	X	X	X	X	X	X			
RFD	ready for data	U	HS	X X X X X X X X	X0X	X	X	X	X	X	X	X	X	X	X	X	X			
RQS	request service (Note 9)	U	ST	X 1 X X X X X X	XXX	0	X	X	X	X	X	X	X	X	X	X	X			
SCG	secondary command group	M	SE	Y 1 1 X X X X X	XXX	1	X	X	X	X	X	X	X	X	X	X	X			
SDC	selected device clear	M	AC	Y 0 0 0 0 1 0 0	XXX	1	X	X	X	X	X	X	X	X	X	X	X			
SPD	serial poll disable	M	UC	Y 0 0 1 1 0 0 1	XXX	1	X	X	X	X	X	X	X	X	X	X	X			
SPE	serial poll enable	M	UC	Y 0 0 1 1 0 0 0	XXX	1	X	X	X	X	X	X	X	X	X	X	X			
SRQ	service request	U	ST	X X X X X X X X	XXX	X	X	1	X	X	X	X	X	X	X	X	X			
STB	status byte (Notes 8, 9)	M	ST	S X S S S S S S 8 6 5 4 3 2 1	XXX	0	X	X	X	X	X	X	X	X	X	X	X			
TCT	take control	M	AC	Y 0 0 0 1 0 0 1	XXX	1	X	X	X	X	X	X	X	X	X	X	X			
TAG	talk address group	M	AD	Y 1 0 X X X X X	XXX	1	X	X	X	X	X	X	X	X	X	X	X			
UCG	universal command group	M	UC	Y 0 0 1 X X X X	XXX	1	X	X	X	X	X	X	X	X	X	X	X			
UNL	unlisten	M	AD	Y 0 1 1 1 1 1 1	XXX	1	X	X	X	X	X	X	X	X	X	X	X			
UNT	untalk (Note 11)	M	AD	Y 1 0 1 1 1 1 1	XXX	1	X	X	X	X	X	X	X	X	X	X	X			

The 1/0 coding on ATN when sent concurrent with multiline messages has been added to this revision for interpretive convenience.

**NOTES:**

- (1) D1-D8 specify the device dependent data bits.
- (2) E1-E8 specify the device dependent code used to indicate the EOS message.
- (3) L1-L5 specify the device dependent bits of the device's listen address.
- (4) T1-T5 specify the device dependent bits of the device's talk address.
- (5) S1-S5 specify the device dependent bits of the device's secondary address.
- (6) S specifies the sense of the PPR.

S	Response
0	0
1	1

P1-P3 specify the PPR message to be sent when a parallel poll is executed.

P3	P2	P1	PPR Message
0	0	0	PPR1
1	1	1	PPR8

- (7) D1-D4 specify don't-care bits that shall not be decoded by the receiving device. It is recommended that all zeroes be sent.
- (8) S1-S6, S8 specify the device dependent status. (DIO7 is used for the RQS message.)
- (9) The source of the message on the ATN line is always the C function, whereas the messages on the DIO and EOI lines are enabled by the T function.
- (10) The source of the messages on the ATN and EOI lines is always the C function, whereas the source of the messages on the DIO lines is always the PP function.
- (11) This code is provided for system use, see 6.3.

# **Controllers**

---

# 8255A Programmable Peripheral Interface Applications

<b>Contents</b>	
<b>INTRODUCTION</b>	2-410
<b>OVERVIEW</b>	2-410
<b>8080 CPU MODULE INTERFACE</b>	2-410
<b>PERIPHERAL INTERFACE SECTION</b>	2-412
<b>INTERNAL LOGIC SECTION</b>	2-413
Mode Definition	
Bit Set/Reset	
<b>INTERRUPT CONTROL LOGIC STATUS WORDS</b>	2-415
<b>SOFTWARE CONSIDERATIONS</b>	2-417
<b>MODE 0—STATUS DRIVEN PERIPHERAL INTERFACE</b>	2-419
8255A to Peripheral Hardware Interface	
8080 CPU Module to 8255A Interface	
Mode 0 Interface Software	
Summary/Conclusions	
<b>MODE 1—INTERRUPT DRIVEN PRINTER INTERFACE</b>	2-424
CPU Module to 8255A Interface	
8255A to Peripheral Interface	
Mode 1 Software Driver	
Summary/Conclusions	
<b>MODE 2—8080 TO 8080 INTERFACE</b>	2-429
Hardware Discussion	
Software Discussion	
Summary/Conclusions	
<b>APPENDIX</b>	2-436
8255A Quick Reference	

## INTRODUCTION

Microprocessor-based system designs are a cost-effective solution to a wide variety of problems. When a system designer is presented with the task of selecting a microprocessor for a design, the capabilities of the microprocessor should not be the only consideration. The microprocessor should be an element of a compatible family of devices. The MCS-80 component family is a group of compatible devices which have been designed to directly address and solve the problems of microprocessor-based system design. One member of the MCS-80 component family is Intel's 8255A programmable peripheral interface chip. This device replaces a significant percentage of the logic required to support a variety of byte oriented Input/Output interfaces. Through the use of the 8255A, the I/O interface design task is significantly simplified, the design flexibility is increased, and the number of components required is reduced.

This application note presents detailed design examples from both the hardware and software points of view. Since the 8255A is an extremely flexible device, it is impossible to list all of the applications and configurations of the device. A number of designs are presented which may be modified to fulfill specific user interface requirements.

Detailed design examples are discussed within the context of the 8080 system shown in Figure 1. The basic 8080 system is composed of the CPU module, memory module, and the I/O module. CPU module and memory module design are discussed

within other Intel publications. This application note deals exclusively with I/O module design.

It is assumed that the reader is familiar with the MCS-80 User's Manual and/or the MCS-85 User's Manual, particularly the 8255A device description.

## OVERVIEW OF THE 8255A

The 8255A block diagram shown in Figure 2 has been divided into three sections: 8080 CPU Module Interface, Peripheral Interface, and the Internal Logic.

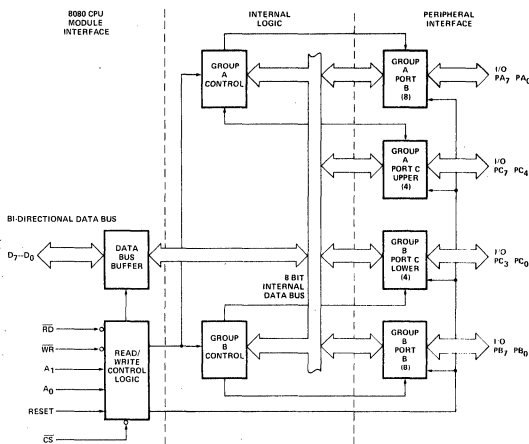
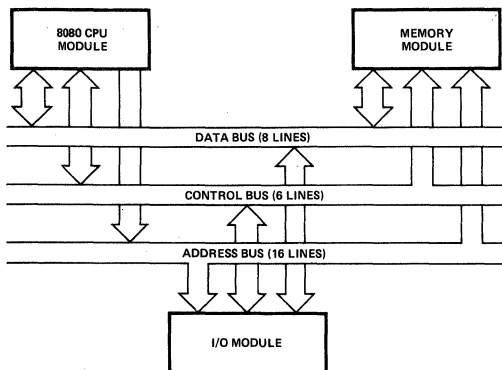


Figure 2. 8255A Block Diagram



## 8080 CPU MODULE INTERFACE

The 8255A is a compatible member of the MCS-80 component family and, therefore, may be directly interfaced to the 8080. Figure 3 displays one method of interconnecting the 8255A and an 8080 CPU module. The 8080 CPU module consists of the 8080A CPU, the 8224 Clock Generator, and the 8228 System Controller. The system shown in Figure 3 utilizes a linear select scheme which dedicates an address line as an exclusive enable (chip select) for each specific I/O device. The chip select signal is used to enable communication between the selected 8255A and the 8080 CPU. I/O Ports A, B, C, or the Control Word Register are selected by the two port select signals ( $A_1$ ,  $A_0$ ). These signals ( $A_1$  and  $A_0$ ) are driven by the least significant bits of the address bus. The I/O port select characters required by this configuration are shown in Figure 4.

# APPLICATIONS

When a system utilizing the linear select scheme is implemented, a maximum of six I/O devices may be selected. If more than six I/O devices must be addressed, the six device select bits must be encoded to generate a maximum of 64 device select lines. Note that when large systems are implemented, bus loading considerations may require that bus drivers be included in the CPU module. The MCS-80 component family contains parts which are designed to perform this function (8216, 8226).

The 8255A I/O read ( $\overline{RD}$ ) and I/O write ( $\overline{WR}$ ) signals may be directly driven by the 8228. This results in an isolated I/O architecture where 8080 Input/Output instructions are used to reference an independent I/O address space. An alternate approach is memory mapped I/O. This architecture treats an area of memory as the I/O address space. The memory mapped I/O architecture utilizes 8080 memory reference instructions to access the I/O address space. Interfacing with the 8080 is outlined in Chapter 3 of the "8080 Microcomputer User's Manual".

The most important feature of the 8255A to 8080 CPU Module Interface is that for small system designs the 8255A may be interfaced directly to

the standard MCS-80 component family with no external logic. Minimum external logic is required in large system designs.

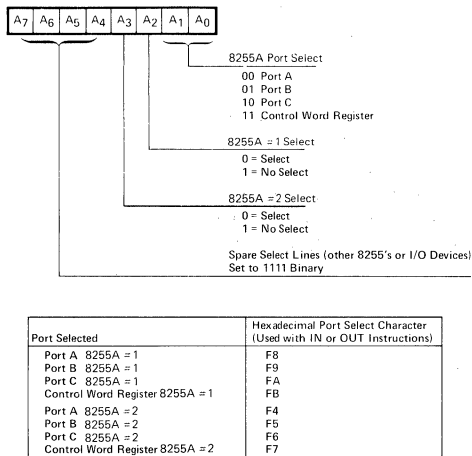


Figure 4. I/O Port Select Characters

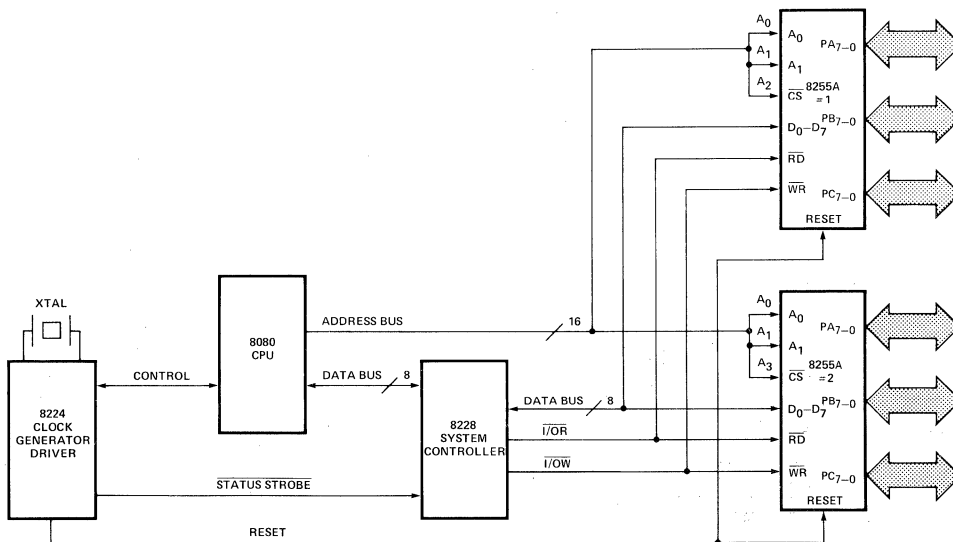


Figure 3. Linear Select 8255A Interconnect



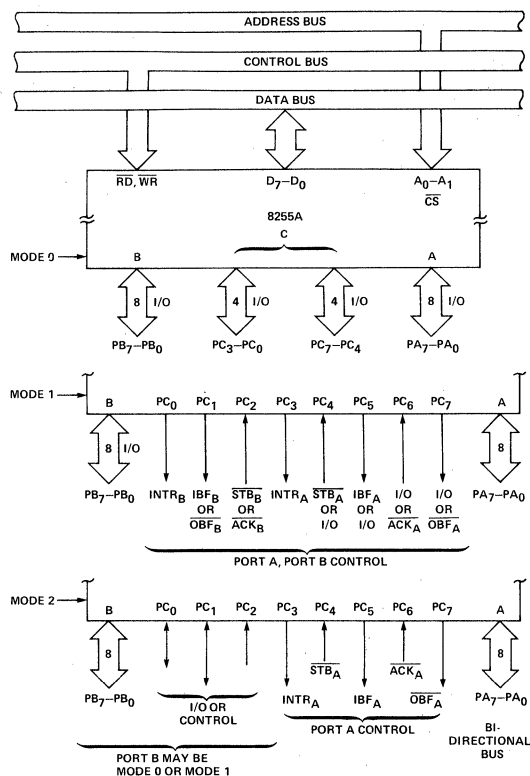
## PERIPHERAL INTERFACE SECTION

The peripheral interface section contains 24 peripheral interface lines, buffers, and control logic. The characteristics and functions of the interface lines are determined by the operating mode selected under program control. The flexibility of the 8255A is due to the fact that the device is programmable. Three modes of operation may be selected under program control: Mode 0 – Basic Input/Output, Mode 1 – Strobed Input/Output with interrupt support, and Mode 2 – Bidirectional bus with interrupt support. Through selecting the correct operating mode, the interface lines may be configured to fulfill specific interface requirements. The characteristics of the interface lines within each mode must be understood so that the designer may utilize the 8255A to achieve the most efficient design. Table I lists the basic features of the peripheral interface lines within each mode group. Figure 5 shows the grouping of the peripheral interface lines within each mode.

**Table I. Features of Peripheral Interface Lines**

<p><b>Mode 0 – Basic Input/Output</b></p> <p>Two 8-bit ports Two 4-bit ports with bit set/reset capability Outputs are latched Inputs are not latched</p>
<p><b>Mode 1 – Strobed Input/Output</b></p> <p>One or two strobed ports Each Mode 1 port contains:     8-bit data port     3 control lines     Interrupt support logic</p> <p>Any port may be input or output If one Mode 1 port is used, the remaining 13 lines may be configured in Mode 0. If two Mode 1 ports are used, the remaining 2 bits may be input or output with bit set/reset capability.</p>
<p><b>Mode 2 – Strobed Bidirectional Bus</b></p> <p>One bidirectional bus which contains:     8-bit bidirectional bus supported by Port A     5 control lines     Interrupt support logic     Inputs and outputs are latched</p> <p>The remaining 11 lines may be configured in either Mode 0 or Mode 1.</p>

One feature of Port C is important to note. Each Port C bit may be individually set and reset. Through the use of this feature, device strobes may be easily generated by software without utilizing external logic. The Mode 1 and Mode 2 configurations use a number of the Port C lines for interrupt control lines. Thus, the 8255A contains a large portion of the logic required to implement an interrupt driven I/O interface. This feature simplifies interrupt driven hardware design and saves a significant amount of the external logic that is normally required when less powerful I/O chips are used. In fact, the design examples contained in this application note describe how interrupt driven interfaces may be designed such that the only interrupt control logic required is that contained in the 8255A.



**Figure 5. Grouping of Peripheral Interface Lines**

**INTERNAL LOGIC SECTION**

The internal logic section manages the transfer of data and control information on the internal data bus (refer to Figure 2). If the port select lines (A<sub>1</sub> and A<sub>0</sub>) specify Ports A, B, or C, the operation is an I/O port data transfer. The internal logic will select the specified I/O port and perform the data transfer between the I/O port and the CPU interface. As was previously mentioned, both the functional configuration of each port and bit set/reset on Port C are controlled by the system's software. When the control word register is selected, the internal logic performs the operation described by the control word. The control word contains an opcode field which defines which of the two functions are to be performed (mode definition or bit set/reset).

**Mode Definition**

When the opcode field (Bit 7) of the control word is equal to a one, the control word is interpreted by the 8255A as a mode definition control word. The mode definition control word (shown in Figure 6) is used to specify the configuration of the

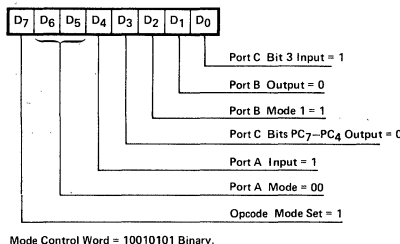
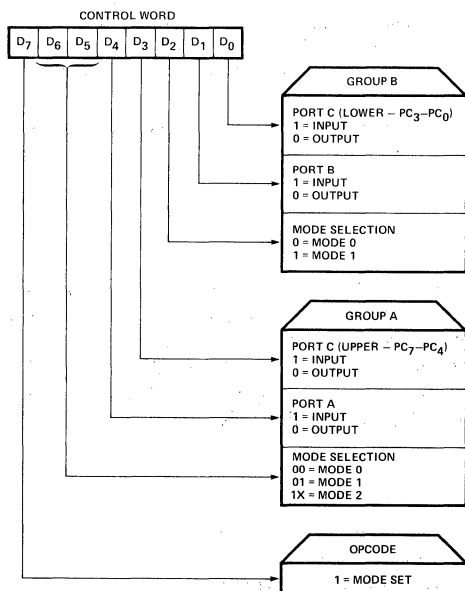
24 8255A peripheral interface lines. The system's software may specify the modes of Port A and Port B independently. Port C may be treated independently or divided into two portions as required by the Port A and Port B mode definitions.

**Example #1:** This example demonstrates how a mode control word is constructed and issued to an 8255A. The mode control word is passed to the device through the use of an output instruction that references an 8080 I/O port address. The value of the I/O port address is determined by the 8080 CPU interface implemented. This example references the I/O port addresses realized by the simple 8080 to 8255A interface shown in Figure 3.

If an 8255A is to be configured through the use of the mode control word interface as:

- Port A      Mode 0 Input
- Port B      Mode 1 Output
- Port C      Bits PC<sub>7</sub>–PC<sub>4</sub> Output
- Port C      Bit 3 Input

The following mode control word is used:



The assembly language program is:

```

CWR            EQU    0FBH            ; 8255A = 1 CONTROL WORD REGISTER
:            .....
:            .....            ISSUE MODE CONTROL WORD
:            .....
              MVI    A,10010101B     ; GET MODE CONTROL WORD
              OUT    CWR            ; OUTPUT TO 8255A = 1 CONTROL WORD REGISTER
              .....

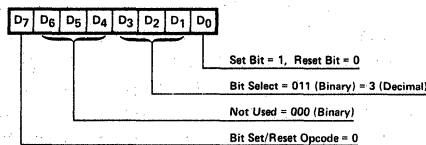
```

**Figure 6. Mode Definition Control Word**

**Bit Set/Reset**

When the opcode field (Bit 7) of the control word is equal to a zero, the control word is interpreted by the 8255A as a Port C bit set/reset command word (see Figure 7). Through the use of the bit set/reset command, any of the 8 bits on Port C may be independently set or reset. Note that control word bits 6-4 are not used. Bits 6-4 should be set to zero.

Control word (see Figure 7).



The control word for set Port C bit 3 is 00000111 binary.  
The control word for reset Port C bit 3 is 00000110 binary.

The assembly language program is:

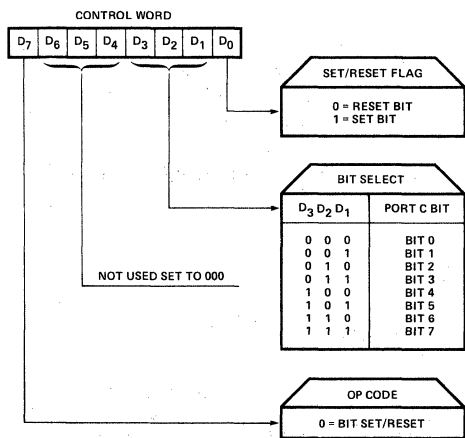


Figure 7. Bit Set/Reset Control Word

**Example #2:** This example demonstrates how a Port C bit set/reset control word is constructed and issued to an 8255A. The bit set/reset control word is passed to the device through the use of an output instruction that references an 8080 I/O port address. The value of the I/O port address is determined by the 8080 CPU interface implemented. This example references the I/O port addresses realized by the simple 8080 to 8255A interface shown in Figure 3.

```
CWR EQU 0FBH ; 8255A = 1 CONTROL WORD REGISTER
.....
SET BIT 3
MVI A, 00000111B ; GET SET BIT 3 CONTROL WORD
OUT CWR ; OUTPUT TO 8255A = 1 CONTROL WORD REGISTER
.....
RESET BIT 3
MVI A, 00000110B ; GET RESET BIT 3 CONTROL WORD
OUT CWR ; OUTPUT TO 8255A = 1 CONTROL WORD REGISTER
```

**NOTE:** An MVI instruction is used to load the reset bit 3 control word into the A register. Since it is known that the set bit control word is already in the A register, a "DCR A" instruction could be used to generate the correct control word and save one byte of code.

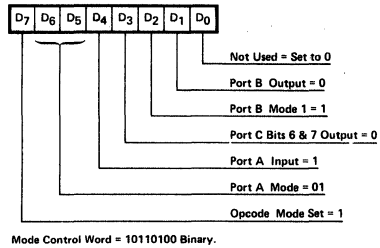
00000111 - 1 = 00000110 (RESET BIT 3 CONTROL WORD)

**Example #3:** This example demonstrates one simple method of performing a bit set/reset operation on Ports A and B. The state of any output port may be determined by reading the port. The assembly language program which may be used to set/reset Port A or B bits is:

```
PORTA EQU 0FBH ; 8255A = 1 PORT A
.....
SET BIT 0
.....
IN PORTA ; GET STATE OF PORT
ORI 01H ; SET BIT 0
OUT PORTA ; OUTPUT TO PORT
.....
RESET BIT 0
.....
IN PORTA ; GET STATE OF PORT
ANI 0FEH ; RESET BIT 0
OUT PORTA ; OUTPUT TO PORT
```

**INTERRUPT CONTROL LOGIC STATUS WORDS**

As previously mentioned, the 8255A Mode 1 and Mode 2 configurations support interrupt control logic. If a read of Port C is issued when the 8255A is configured in Mode 1, the software will receive the Mode 1 status word shown in Figure 8. The bits in the status word correspond to the state of the associated Port C lines (buffer full, interrupt request, etc.). The INTE bit shown in the status word corresponds to the interrupt enable flip-flop contained in the 8255A. This signal is not available externally. The structure of the Mode 1 status word varies as a function of the mode of the 8255A. Example #4 shows the status word which results from reading Port C from an 8255A which is configured with Port A Mode 1 input and Port B Mode 1 output.



After the 8255A mode control word has been issued, a READ of Port C will obtain the following Mode 1 status word:

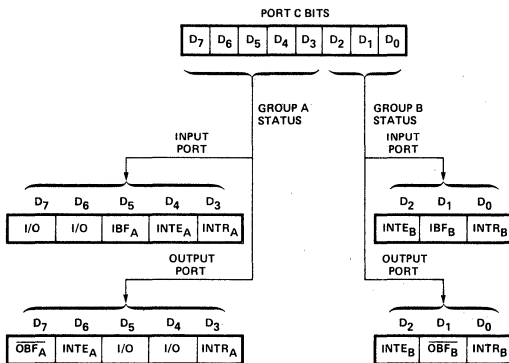


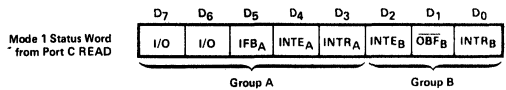
Figure 8. Mode 1 Status Word

**Example #4 – MODE 1 STATUS WORD**

If an 8255A is to be configured through the use of the mode control word interface as:

- Port A Mode 1 Input
- Port B Mode 1 Output
- Port C Bits 6 & 7 Output

The following mode control word is used:



**NOTE:** The Port C I/O bits D7 and D6 should be modified through the use of the Port C bit set/reset command word. If a write to Port C is issued, the INTE<sub>A</sub> and INTE<sub>B</sub> bits may be inadvertently modified by the user. The IBF<sub>A</sub>, INTR<sub>A</sub>, OBF<sub>B</sub>, and INTR<sub>B</sub> bits will not be modified by either a write to Port C or a bit set/reset command. These four bits always reflect the state of the interrupt control logic.

Note that the Mode 2 status word (shown in Figure 9) differs from the Mode 1 status word. The format of the status word data bits D<sub>2</sub>–D<sub>0</sub> are defined by the specification of the Port B configuration. Example #5 shows the structure of the Mode 2 status word when the 8255A is configured with Port A Mode 2 (bidirectional bus) and Port B Mode 1 input.

The Mode 1 and Mode 2 status words reflect the state of the interrupt logic supported by the 8255A.

Example #6 demonstrates how the interrupt enable bits are controlled through the use of the Port C bit set/reset feature. The application examples provide a more detailed explanation of the use of the Port C status word in the Mode 1 and Mode 2 configurations.

After the 8255A mode control word has been issued, a read of Port C will obtain the following Mode 2 status word:

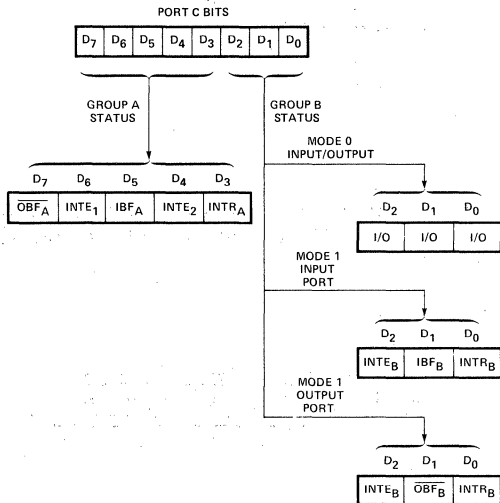


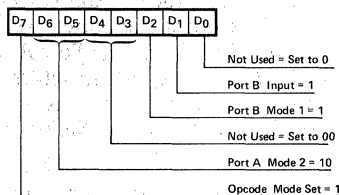
Figure 9. Mode 2 Status Word

**Example #5 – MODE 2 STATUS WORD**

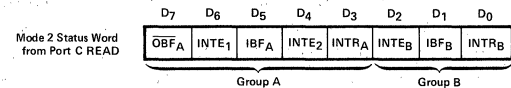
If the 8255A is to be configured as follows:

- Port A Mode 2 Bidirectional Bus
- Port B Mode 1 Input

The following mode control word is used:



Mode Control Word = 11000110 Binary.



**Example #6 – MODE 2 INTERRUPT ENABLE/DISABLE**

The Mode 2 status word shown in Figure 9 contains two interrupt enable bits:

- INTE<sub>1</sub> – Bit 6 – Enable output interrupts
- INTE<sub>2</sub> – Bit 4 – Enable input interrupts

Bit set/reset control words may be constructed which may be used to control the INTE bits.

Set Bit 6 (Enable Output Interrupts) = 00001101 Binary

Reset Bit 6 (Disable Output Interrupts) = 00001100 Binary

Set Bit 4 (Enable Input Interrupts) = 00001001 Binary

Reset Bit 4 (Disable Input Interrupts) = 00001000 Binary

The control words shown were constructed from the standard bit set/reset format shown in Figure 7.

The value of CWR used in the following program example corresponds to the 8080 configuration shown in Figure 3.

```

CWR EQU 0FBH ; 8255A #1 CONTROL WORD REGISTER
.....
ENABLE INTERRUPTS FOR MODE 2 OUTPUT (SET PORT C BIT 6)
.....
MVI A, 00001101B ; GET SET BIT 6 CONTROL WORD
OUT CWR ; OUTPUT TO 8255 #1 CONTROL WORD REGISTER
.....
DISABLE INTERRUPTS FOR MODE 2 OUTPUT (RESET PORT C BIT 6)
.....
MVI A, 00001100B ; GET RESET BIT 6 CONTROL WORD
OUT CWR ; OUTPUT TO 8255A #1 CONTROL WORD REGISTER

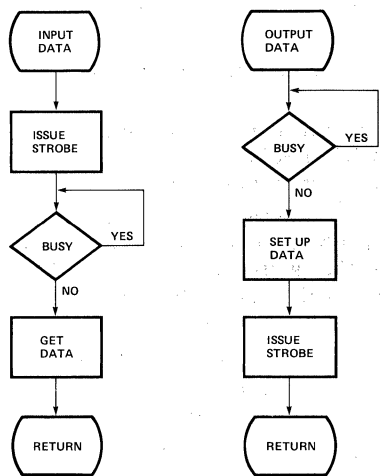
```

**SOFTWARE CONSIDERATIONS**

Regardless of the mode selected, the software must always issue the correct mode control word after a reset of the device. Generally, an initialization routine is constructed which issues the correct mode control word, sets up the initial state of the control lines, and initializes any program internal data.

Many of the software requirements of the 8255A vary as a function of the mode selected. The simplest mode supported by the device is Mode 0 (Basic Input/Output). Generally, Mode 0 is used for simple status driven device interfaces (no interrupts). Figure 10 illustrates sample software that could be used to support such interfaces. Most devices support a BUSY or READY signal which is used to determine when the device is ready to input or output data and a DATA STROBE which is used to request data transfer (DATA STROBE may easily be generated with the Port C bit set/reset feature). In the Mode 0 configuration, Ports A and B are used to input/output byte oriented data. Port C is used to input 8255A status, peripheral status and to drive peripheral control lines.

When the Mode 1 and Mode 2 configurations are used, the software is generally required to support interrupts. Software routines written for an interrupt driven environment tend to be more complex than status driven routines. The added complexity is due to the fact that interrupt driven systems are constructed such that other software tasks are run while the I/O transaction is in progress. A software routine that controls a peripheral device is generally referred to as a device driver. One method of implementing an interrupt driven device driver is to partition the device driver into a "Command Processor" and an "Interrupt Service Routine". The command processor is the module that validates



**Figure 10. Sample Status Driven Software Flowchart**

and initiates user program requests to the device driver. A common method of passing information between the various software programs is to have the requesting routine provide a device control block in memory. A sample device control block is shown in Table II.

**Table II. Sample Device Control Block**

NAME	DESCRIPTION
Status	This 1-byte field is used to transmit the status of the I/O transaction (busy, complete, etc.).
Opcode	This 1-byte field defines the type of I/O (READ, WRITE, etc.).
Buffer Address	This 2-byte field specifies the source/destination of the data block.
Character Count	This 1-byte field is a count of the number of characters involved in the transaction.
Character Transferred Count	This 1-byte count of the number of characters which were actually transferred.
Completion Address	This 2-byte field is the address of the user supplied completion routine which will be called after the I/O has been performed.

The command processor validates the transaction and initiates the operation described by the control block. Control is then returned to the requestor so that other processing may proceed. The interrupt service routine processes the remainder of the transaction.

The interrupt service routine supports the following functions:

1. The state of the machine (registers, status, etc.) must be saved so that it may be restored after the interrupt is processed.
2. The source of the interrupt must be determined. The hardware may support a register which indicates the interrupting device, or the software may poll the devices through interrogating the Port C status word of each 8255A.
3. Data must be passed to or from the device.
4. Control must be passed to the requesting routine at the completion of the I/O.
5. The state of the machine must be restored before returning to the interrupted program.

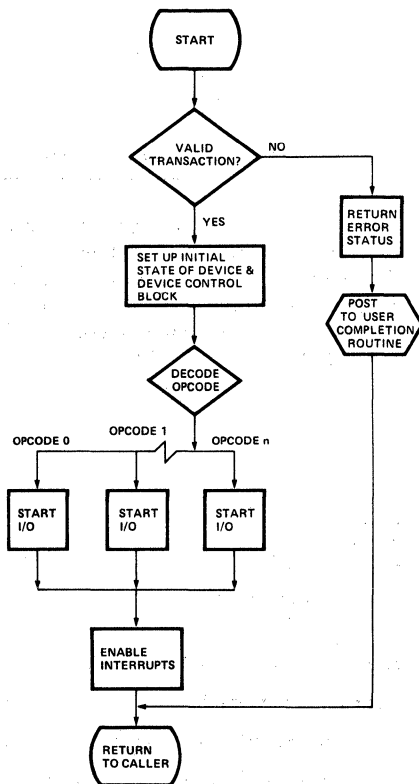


Figure 11. Command Processor

Figures 11 and 12 are simplified flowcharts of one of the many methods of implementing command processor and interrupt service routine modules.

The rest of this application note presents specific application examples. All of the 8080 assembly language programs supplied with the application examples use the standard Intel 8080 assembly language mnemonics. The programs discussed use the program equate statement to specify all hardware related data. Equate statements are used so that all references to an I/O port may be changed through a simple reassignment of the port address in the equate statement.

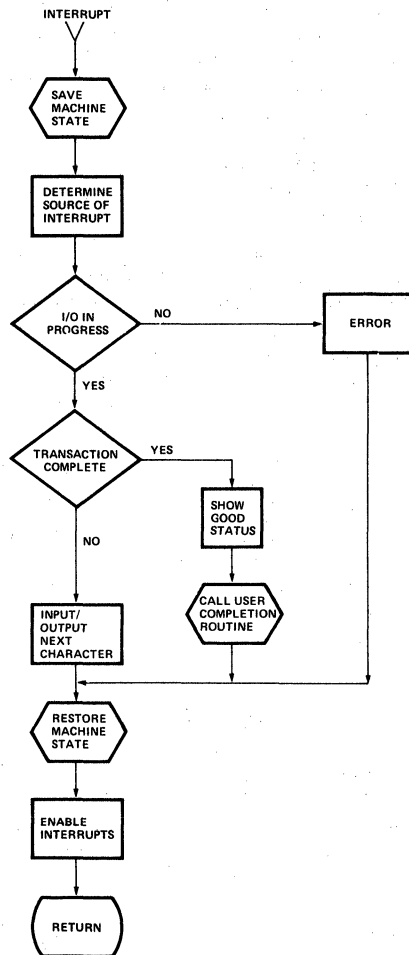


Figure 12. Interrupt Service Routine

### MODE 0 – STATUS DRIVEN PERIPHERAL INTERFACE

This design example shows how a single 8255A in Mode 0 may be used to develop a status driven interface (no interrupts) for the Centronics 306 character printer, the Remex paper tape punch, and the Remex paper tape reader.

#### 8255A To Peripheral Hardware Interface

The first step in the design is to examine the specification for the peripheral devices and identify the control and data signals which must be supported by the interface. Table III lists the signals which were chosen to be supported by the 8255A interface. All three of the devices support the standard

Table III. Peripheral Interface Signals

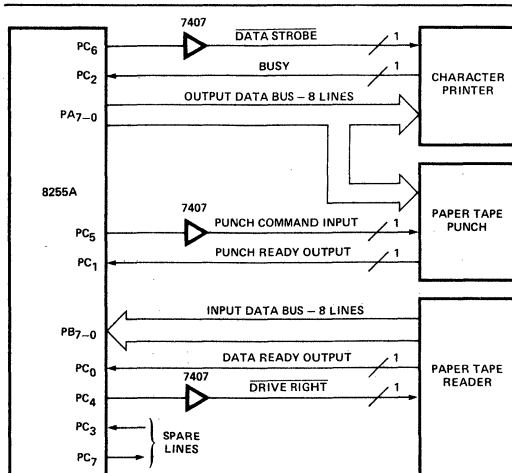
CHARACTER PRINTER	
Name:	DATA 0–DATA 7
Definition:	Input data levels. A high signal represents a binary 1 and a low signal represents a binary 0. These eight lines are the data lines to the printer.
Name:	DATA STROBE
Definition:	A 0.5 μsec pulse (minimum) used to transfer data from the 8255A to the printer.
Name:	BUSY
Definition:	The level indicating that the printer cannot receive data.
PAPER TAPE PUNCH	
Name:	TRACKS 1–8 DATA INPUT
Definition:	Input data levels. A high signal causes a hole to be punched on the associated track. These eight lines are the data lines to the printer.
Name:	PUNCH COMMAND INPUT
Definition:	A true condition moves the tape and initiates punching the tape. This signal is actually a data strobe.
Name:	PUNCH READY OUTPUT
Definition:	True signal indicates that the punch is ready to accept a punch command. This is the punch busy line.
PAPER TAPE READER	
Name:	DATA TRACK OUTPUTS
Definition:	True signal indicates data track hole. These eight lines are the data lines from the punch.
Name:	DRIVE RIGHT
Definition:	True signal drives the tape to the right and reads a character. This signal is actually the data strobe (initiate read signal).
Name:	DATA READY OUTPUT
Definition:	True signal indicates data track outputs are in "On character" condition. This signal is the reader busy line.

BUSY/DATA STROBE interface discussed previously (see Figure 10). Figure 13 is a block diagram of the interface design. The 8255A Port A is configured as a Mode 0 output port which is used to support the printer and the paper tape punch data bus. Port B is configured as a Mode 0 input port and is used to input the paper tape reader data. Three of the Port C lower bits (PC<sub>2</sub>–PC<sub>0</sub>) configured in input mode are used to input the device busy indications. Three of the Port C upper bits (PC<sub>6</sub>–PC<sub>4</sub>) configured in output mode are used to support the device strobe signals required by each device.

The drive requirements of the interface lines are a function of the peripheral interface circuitry, the length of the interface cable, and the environment in which the unit is running. In this particular design example, all output lines from the 8255A to the peripherals were buffered through a 7407 buffer/driver. The input lines from the peripherals were fed directly into the Port C and Port B inputs.

#### 8080 CPU Module To 8255A Interface

The schematic of the completed hardware design is shown in Figure 14. The CPU module design shown is the design which was implemented for Intel's SDK 80 kit board. The 8255A is addressed through the use of an isolated I/O architecture utilizing a linear select scheme. Address bits A<sub>1</sub> and A<sub>0</sub> are used to select the 8255A port. Address bit A<sub>3</sub> is the exclusive enable for 8255A #1. Examination of the schematic shows that all of the 8255A interface lines are directly driven by the CPU module.



NOTE:  
 1. OUTPUT DATA BUS BUFFERED WITH 7407.  
 2. ALL 8255A OUTPUT LINES ARE PULLED UP TO +5V AT THE PERIPHERAL.

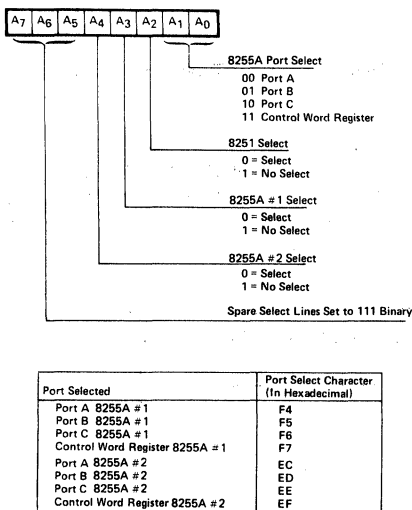
Figure 13. Interface Block Diagram





## Mode 0 Interface Software

An initialization routine and three device drivers (one for each peripheral device) are required to support the peripheral interface. The I/O port addresses implemented by the hardware are shown in Figure 15. The unused chip select bits are set to one so that chip select conflicts will not result if the unused bits are required by an expanded system.

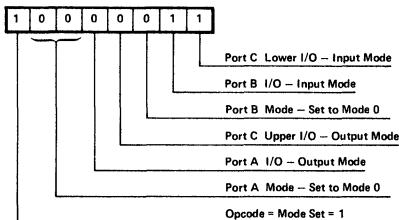


**Figure 15. I/O Port Addresses**

Note that the initialization routine issues the mode control word (shown in Figure 16). It also sets the low true DATA STROBE signals to an inactive (high) state.

### Hardware Requirements:

- Port A - Output Mode 0
- Port C Upper Bits C7-C4 - Output Mode 0
- Port C Lower Bits C3-C0 - Input Mode 0
- Port B - Input Mode 0



Mode Control Word = 10000011 Binary = 83 HEX.

**Figure 16. Mode Control Word**

```

ISIS 8080 MACRO ASSEMBLER, V1.0                PAGE 1
MODE ZERO EXAMPLE

;*****
;***** TITLE 'MODE ZERO EXAMPLE'
;*****
;***** CHARACTER PRINTER, PAPER TAPE PUNCH, PAPER TAPE READER
;***** MODE ZERO EXAMPLE
;*****
;***** PROGRAM EQUATES
;*****
00F4  PORTA EQU 0F4H ; 8255 PORT A
00F6  PORTB EQU 0F6H ; 8255 PORT B
00F7  PORTC EQU 0F7H ; 8255 PORT C
;***** CWR EQU 0F7H ; 8255 CONTROL WORD REGISTER
;*****

;*****
;***** INITIALIZATION CONTROL WORD
;*****
;***** USED TO CONFIGURE THE 8255 AS FOLLOWS:
;*****
;***** PORT A - OUTPUT MODE ZERO
;***** PORT B - INPUT MODE ZERO
;***** PORT C (UPPER) - OUTPUT
;***** PORT C (LOWER) - INPUT
;*****

0083  ICW EQU 10000011B ; INITIALIZATION CONTROL WORD
;*****
;***** SET/RESET CONTROL WORDS FOR GENERATION OF DATA STROBES
;***** ON PORT C.
;*****
000D  LPSON EQU 00001101B ; PRINTER DATA STROBE ON
000C  LPSON EQU 00001100B ; PRINTER DATA STROBE OFF
0008  PMSON EQU 00001011B ; PUNCH DATA STROBE ON
000A  PMSOF EQU 00001010B ; PUNCH DATA STROBE OFF
0009  RDSON EQU 00001001B ; READER DATA STROBE ON
0008  RDSOF EQU 00001000B ; READER DATA STROBE OFF
;*****
;***** BIT MASK FOR DEVICE BUSY CHECK
;*****
0004  LPSBY EQU 04H ; LINE PRINTER BUSY
0002  PMSBY EQU 02H ; PUNCH BUSY
0001  RDBSY EQU 01H ; READER BUSY
    
```

```

ISIS 8080 MACRO ASSEMBLER, V1.0                PAGE 2
MODE ZERO EXAMPLE - INITIALIZATION ROUTINE

;*****
;***** PROGRAM ORIGIN
;*****
3000  ORG 03000H
;*****
;***** INITIALIZATION ROUTINE
;***** A REGISTER MODIFIED
;*****
INIT:
3000 3E83 MVI A,ICW ; GET INITIALIZATION CONTROL WORD
3002 D3F7 OUT CWR ; OUTPUT TO CONTROL WORD REGISTER
;*****
;***** SET ALL LOW TRUE DATA STROBES ON
;*****
3004 3E0D MVI A,LPSON ; GET CONTROL WORD TO TURN ON PRINTER DATA STROBE
3006 D3F7 OUT CWR ; OUTPUT TO CONTROL WORD REGISTER
3008 3E09 MVI A,RDSON ; GET CONTROL WORD TO TURN ON READER DATA STROBE
300A D3F7 OUT CWR ; OUTPUT TO CONTROL WORD REGISTER
300C C9 RET ; RETURN TO CALLER
    
```

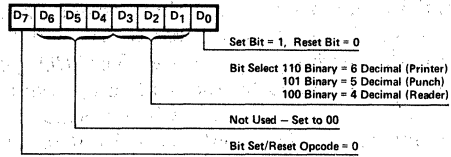
# APPLICATIONS

The three peripheral drivers which follow all have the basic structure discussed previously. Consider the printer routine. Here the user routine places an ASCII data character in the C-register and passes control to the LPST location through a subroutine call. The printer driver interrogates the status of the printer by reading Port C. If the printer is busy, the routine will loop until the printer is idle. When the printer is ready to accept a data character, the character is placed on the Port A lines and a DATA STROBE is generated. After generating the DATA STROBE, the driver executes a subroutine return to the caller.

The DATA STROBE signals to the devices are generated through the use of the Port C bit set/reset feature. The bit set/reset control words used are shown in Figure 17.

## Summary/Conclusions

This design example discussed the basic hardware and software required to handle a simple device interface. The 8255A will easily accommodate a more complex interface design which utilizes additional interface lines supported by the peripheral.

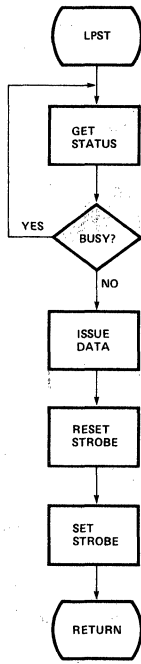


The control word for set Printer DATA STROBE (PC 6) = 00001101 binary.  
 The control word for reset Printer DATA STROBE (PC 6) = 00001100 binary.  
 The control word for set Punch DATA STROBE (PC 5) = 00001011 binary.  
 The control word for reset Punch DATA STROBE (PC 5) = 00001010 binary.  
 The control word for set Reader DATA STROBE (PC 4) = 00001001 binary.  
 The control word for reset Reader DATA STROBE (PC 4) = 00001000 binary.

Figure 17. Bit Set/Reset Control Words

For instance, one of the spare Port C output lines may be used to control the punch direction. Support of this additional feature would require minor modification of the device driver so that the punch direction line could be specified by the user routine.

Through consideration of this example, the use of the 8255A in Mode 0 should become evident.



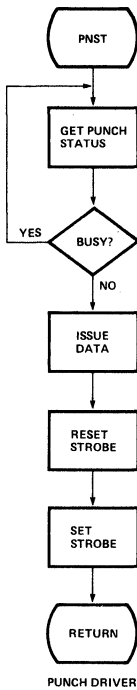
PRINTER DRIVER

ISIS 8080 MACRO ASSEMBLER, V1.0 PAGE 3  
 MODE ZERO EXAMPLE - CHARACTER PRINTER DRIVER

```

:*****
:
: CHARACTER PRINTER DRIVER
:
: INPUTS: CHARACTER TO PRINT IN C-REG
: OUTPUTS: CHARACTER TO PRINTER
:
: A REGISTER MODIFIED
:*****
LPST:
300D DBF6 IN PORTC ; GET STATUS OF PRINTER
300F E604 ANI LPSY ; SEE IF BUSY
:
: JNZ LPST ; IF BUSY - JUMP TO LPST (WAIT LOOP)
:*****
PRINTER IS IDLE - OUTPUT A CHARACTER
:*****
3014 79 MOV A,C ; GET DATA BYTE SUPPLIED BY CALLER
3015 D3F4 OUT PORTA ; OUTPUT DATA TO DATA LINES
3017 360C MVI A,LPSDF ; GET DATA STROBE CONTROL WORD
3019 D3F7 OUT CWR ; RESET DATA STROBE (LOW 'TRUE SIGNAL')
301B 3C INR A ; GENERATE SET DATA STROBE CONTROL WORD
301C D3F7 OUT CWR ; SET DATA STROBE
301E C9 RET ; RETURN TO CALLER
    
```

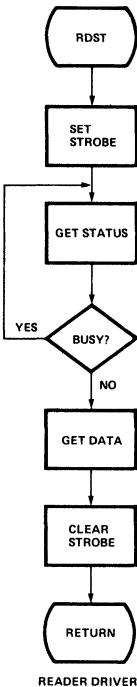
# APPLICATIONS



ISIS 8080 MACRO ASSEMBLER, V1.0 PAGE 4  
 MODE ZERO EXAMPLE - PAPER TAPE PUNCH DRIVER

```

:*****
: PAPER TAPE PUNCH DRIVER
:
: INPUTS : DATA TO PUNCH IN C-REGISTER
: OUTPUTS: DATA TO PUNCH
:
: A REGISTER MODIFIED
:*****
PNST:
301F DBF6      IN   PORTC  ; GET STATUS OF PUNCH
3021 B602      ANI   PMSBY  ; SEE IF BUSY
3023 C21F30    JNZ   PNST   ; IF BUSY - JUMP TO PNST (WAIT LOOP)
:*****
: PUNCH IS IDLE - OUTPUT A CHARACTER
:*****
3026 79      MOV   A,C    ; GET DATA BYTE SUPPLIED BY CALLER
:          OUT  PORTA  ; OUTPUT DATA TO DATA LINES
3029 3B08    MVI   A,PMSON ; SET DATA STROBE CONTROL WORD
302B D3F7    OUT  CWR   ; SET DATA STROBE
302D 3D      DCR   A    ; GENERATE RESET DATA STROBE CONTROL WORD
302E D3F7    OUT  CWR   ; RESET DATA STROBE
3030 C9      RET                   ; RETURN TO CALLER
  
```



ISIS 8080 MACRO ASSEMBLER, V1.0 PAGE 5  
 MODE ZERO EXAMPLE - PAPER TAPE READER DRIVER

```

:*****
: PAPER TAPE READER DRIVER
:
: INPUTS : DATA FROM READER
: OUTPUTS: CHARACTER TO USER IN C-REGISTER
:
: A AND C REGISTER MODIFIED
:*****
RDST:
3031 3E08    MVI   A,RDSOF ; GET STROBE CONTROL WORD (LOW TRUE SIGNAL)
3033 D3F7    OUT  CWR   ; SET DATA STROBE
:*****
RDLP:
3035 DBF6      IN   PORTC  ; GET STATUS OF DEVICE
3037 E601      ANI   RDSBY  ; SEE IF BUSY
3039 C23530    JNZ   RDLP   ; IF BUSY - LOOP UNTIL IDLE
:*****
: READER NOT BUSY - GET CHAR AND CLEAR STROBE
:*****
303C DBF5      IN   PORTB  ; GET CHARACTER
303E 0E07    MVI   C,A    ; SAVE CHARACTER
3040 3E09    MVI   A,RDSON ; GET STROBE SET CONTROL WORD (LOW TRUE SIGNAL)
3042 D3F7    OUT  CWR   ; TURN OFF STROBE
3044 C9      RET                   ; RETURN TO CALLER
:*****
: END OF MODE ZERO EXAMPLE
:*****
0000      END
  
```

## MODE 1 INTERRUPT DRIVEN PRINTER INTERFACE

The status driven interface described in the previous example required the software driver to poll the device status for completion. An alternate approach is to construct the device interface such that an interrupt is used to signal the completion of the operation. When an interrupt driven interface is utilized, the time that was dedicated to polling can be used to perform other functions and the effective processor through-put is increased. This example demonstrates how an 8255A configured in Mode 1 may be used to develop an interrupt driven interface for the Centronics 306 character printer.

### CPU Module To 8255A Interface

The 8080 bus interface implemented for this example is the same as the Mode 0 example with the addition of interrupt support. Interrupt support is implemented through the use of a special feature of the 8228 System Controller. If only one interrupt vector is required (such as in small systems), the 8228 can automatically assert an RST 7 instruction onto the data bus at the proper time. This option is selected by connecting the  $\overline{INTA}$  output of the 8228 to the +12-volt supply through a 1K ohm series resistor.

The Mode 1 interrupt support logic of the 8255A provides an interrupt request line for each port. The 8255A interrupt request line ( $\overline{INTR}_A$ ) must be connected to the INT line of the 8080. A 10K ohm pullup resistor is used to insure that the  $V_{IH}$  requirements of the 8080 are met.

### 8255A To Peripheral Interface

The interrupt driven configuration control signal interface to the printer is different than the status driven interface. Instead of a BUSY/DATA STROBE interface, a DATA STROBE/ACK interface is supported. The ACK signal notifies the 8255A that a character transferred to the printer by a DATA STROBE has been accepted. After an ACK is issued the printer is considered idle. The block diagram shown in Figure 18 displays the interface signals used.

The Mode 1 interrupt driven peripheral support signals used are:

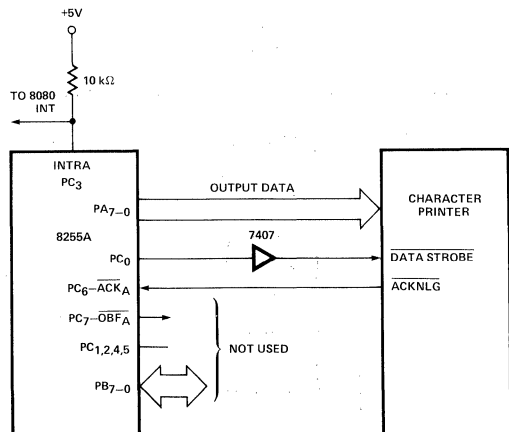
- PA<sub>7</sub>-PA<sub>0</sub> - Output Data  
Used to support the printer data port.
- $\overline{OBF}$  - Output Buffer Full  
This line goes low when data is placed in the output buffer. The OBF signal may be used as a data

strobe signal when interfacing to peripherals which do not require a pulsed input. The Centronics 306 requires a pulsed DATA STROBE signal. This signal is supported by Port C bit 0.

$\overline{ACK}$

- ACKnowledge

This line is used to signal the 8255A that the device has accepted the data. This line is supported by the printer ACKNLG signal.



**NOTES:**

1. DATA BUS BUFFERED WITH 7407.
2. ALL 8255A OUTPUT LINES ARE PULLED UP TO +5V AT THE PERIPHERAL.

Figure 18. Interface Block Diagram

# APPLICATIONS

## Mode 1 Software Driver

The software driver implemented for this example utilizes the typical interrupt driven software structure outlined previously. The initialization routine issues the mode control word (shown in Figure 19) to the 8255A after reset of the device. The initialization routine also places a jump to the interrupt service routine in the interrupt location for RST 7. The command processor is started by the user routine through a subroutine call to PSTRT, with the address of the control block in the D and E registers (the control block format is shown in Table IV). The command processor insures that an I/O operation is not already in progress, starts the I/O, enables interrupts, and returns to the caller so that other processing may proceed.

After a character is placed in the output buffer, the DATA STROBE signal is generated through the use of the Port C bit set/reset feature. When the ACK is generated by the printer, the buffer full indication is cleared and the 8255A generates an interrupt. If interrupts are enabled, the interrupt request is serviced by the 8080 CPU through disabling processor interrupts and then executing the instruction at location 38 hexadecimal in program memory. The interrupt service routine saves the processor state and polls the 8255A to determine the source of the interrupt. Once the interrupting device is located, the control block is used to locate the next data character for transfer to the 8255A output buffer. After the entire buffer has been printed, the interrupt service routine passes control to the user-supplied completion routine. Before returning from the interrupt, the state of the processor is restored.

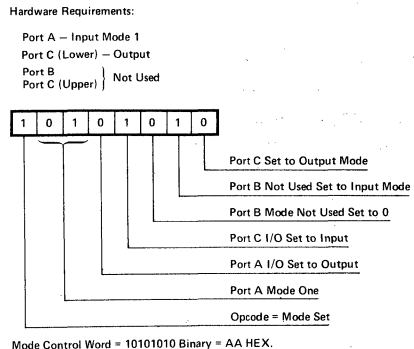


Figure 19. Mode Control Word

Table IV. Printer Software Control Block

NAME	POSITION	DEFINITION
Status	Byte 0	A 1-byte field which defines the completion status of an I/O. 00 = Good completion 01 = Error – command already in progress
Buffer Address	Byte 1, 2	Pointer to the start of the data to print.
Character Count	Byte 3	Count of the number of characters to print.
Character Transferred Count	Byte 4	The number of characters transferred.
Completion Address	Byte 5, 6	Address of a user supplied routine which will be called after the I/O has been performed.

**NOTES:**

1. An opcode field is not required because WRITE is the only operation performed.
2. The control block must reside above location FF Hex.

# APPLICATIONS

There are a number of error conditions which may occur, such as an interrupt from a device which does not have a control block in progress, or an interrupt when polling establishes that no device requires service. Neither of these errors should occur, but if they do, the driver should perform in a consistent fashion. The recovery routines implemented to handle error conditions are determined by the particular applications environment.

## Summary/Conclusions

When utilized in a small system design, the 8255A interrupt support logic provides all of the capabilities required to implement an interrupt driven hardware interface without the use of external logic. In larger system designs, the designer may chose to use additional hardware to determine the source of interrupt requests without software polling. The software design required by an interrupt driven system is inherently more complex than the status driven interface. If an interrupt driven system is required the added complexity is a small price to pay for a significant increase in system through-put.

ISIS 8080 MACRO ASSEMBLER, V1.0  
MODE ONE EXAMPLE

PAGE 1

```

;*****
;
; TITLE 'MODE ONE EXAMPLE'
;
; CHARACTER PRINTER - INTERRUPT DRIVEN
; MODE ONE EXAMPLE
;*****
;
; PROGRAM EQUATES
;*****
00F4 PORTA EQU 0F4H ; 8255 PORT A
00F5 PORTB EQU 0F5H ; 8255 PORT B
00F6 PORTC EQU 0F6H ; 8255 PORT C
00F7 CWR EQU 0F7H ; 8255 CONTROL WORD REGISTER
0038 RST7 EQU 038H ; RST7 ADDRESS
;*****
;
; INITIALIZATION CONTROL WORD
;
; USED TO CONFIGURE THE 8255 AS FOLLOWS:
;
; PORT A - OUTPUT MODE 1
; PORT B - INPUT MODE 0 (NOT USED)
; PORT C LOWER - OUTPUT
;*****
000A ICW EQU 10101010B ; INITIALIZATION CONTROL WORD
;*****
; SET/RESET CONTROL WORDS
;*****
0001 STBCN EQU 0000001B ; SET STROBE
0000 STBOF EQU 0000000B ; RESET STROBE
;*****
; 8255 ENABLE/DISABLE INTERRUPT CONTROL WORDS
;*****
000D IEN EQU 0001101B ; ENABLE INTERRUPTS
000C IDN EQU 0001100B ; DISABLE INTERRUPTS
;*****
; DEVICE STATUS EQUATES
;*****
0006 LPBSY EQU 008H ; BUFFER FULL (LINE PRINTER BUSY)
0008 INTRA EQU 008H ; INTERRUPT REQUEST

```

ISIS 8080 MACRO ASSEMBLER, V1.0  
MODE ONE EXAMPLE

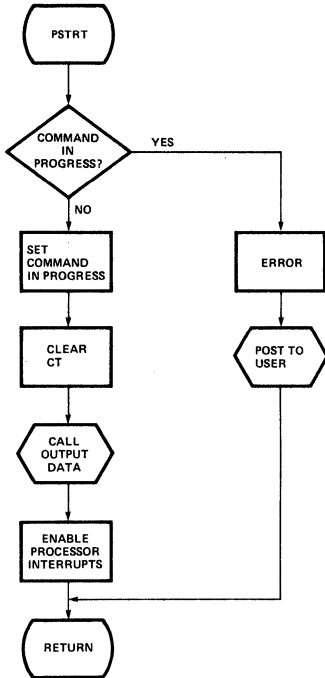
PAGE 2

```

;*****
; CONTROL BLOCK EQUATES
;*****
0000 CBST EQU 00H ; STATUS BYTE
0001 CBUF EQU 01H ; BUFFER ADDRESS
0003 CBCR EQU 03H ; CHARACTER COUNT
0004 CBCT EQU 04H ; CHARACTER TRANSFERRED COUNT
0005 CBCMP EQU 05H ; COMPLETION SERVICE ADDRESS
;*****
; COMPLETION STATUS EQUATES
;*****
0000 STGD EQU 00H ; GOOD COMPLETION
0001 STEL EQU 01H ; ERROR - COMMAND ALREADY IN PROGRESS
;*****
; PROGRAM ORIGIN
;*****
3000 ORG 03000H
;*****
; INITIALIZATION ROUTINE
;
; A,H,L REGISTERS MODIFIED
;*****
INIT:
3000 3EAA MVI A,ICW ; GET MODE CONTROL WORD
3002 D3F7 OUT CWR ; OUTPUT TO CONTROL WORD REGISTER
3004 3E91 MVI A,STBCN ; GET SET DATA STROBE CONTROL WORD
3006 D3F7 OUT CWR ; SET DATA STROBE (LOW TRUE SIGNAL)
;*****
; SET UP RST7 LOCATION WITH JUMP TO FINI
;*****
3008 3EC3 MVI A,0C3H ; GET "JMP"
300A 223000 STA RST7 ; PLACE IN RST7 LOCATION
300D 213030 LXI H,FINI ; GET ADDRESS OF INTERRUPT SERVICE ROUTINE
3010 223900 SHLD RST7+1 ; STORE ADDRESS
3013 C9 RET ; RETURN TO CALLER

```

# APPLICATIONS

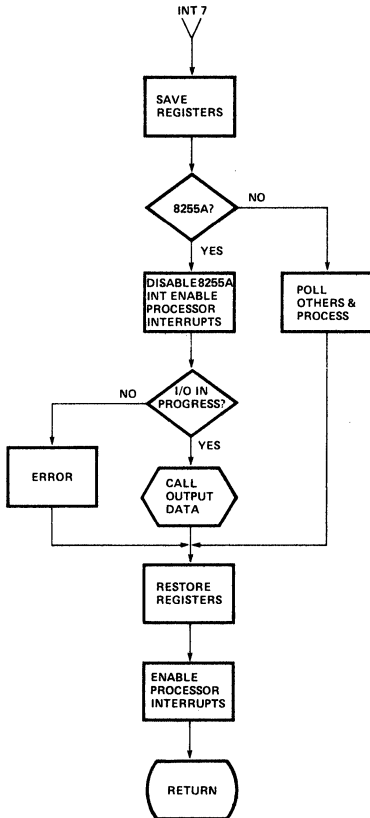


ISIS 8080 MACRO ASSEMBLER, V1.0  
COMMAND PROCESSOR

PAGE 3

```

;*****
; COMMAND PROCESSOR
; INPUTS: CONTROL BLOCK ADDRESS IN D AND E REGISTERS
; OUTPUTS: START I/O OR ERROR STATUS IN CONTROL BLOCK
; A,H,L REGISTERS MODIFIED
;*****
PSTRT:
3014 3AA230 LDA PIPRG+1 ; GET PRINT IN PROGRESS BLOCK ADDRESS
3017 A7 ANA A ; SEE IF ZERO (PRINT IN PROGRESS)
; IF BLOCK ADDRESS NOT EQUAL TO ZERO THEN
; PRINT IN PROGRESS
; IF YES - BRANCH TO ERROR
3018 C22B30 JNZ PSTE
301B EB XCHG PIPRG ; SAVE CONTROL BLOCK ADDRESS
301C 22A130 SHLD PIPRG ; GET INDEX TO CT
301D EB XCHG D ; COMPUTE ADDRESS OF CT
3020 210400 LXI H,CBCT ; COMPUTE ADDRESS OF CT
3023 19 DAD D ; CLEAR CT
3024 3600 MVI M,00H ; START I/O
3026 CD5830 CALL PDATA ; ENABLE PROCESSOR INTERRUPTS
3029 FB EI ; RETURN TO CALLER
302A C9 RET
;*****
; ERROR - TRANSACTION ALREADY IN PROGRESS
;*****
PSTE:
302B 3E01 MVI A,STE1 ; GET ERROR STATUS CODE
302D C39430 JMP POST ; PASS CONTROL TO COMPLETION ROUTINE
  
```



ISIS 8080 MACRO ASSEMBLER, V1.0  
PRINTER INTERRUPT SERVICE ROUTINE

PAGE 4

```

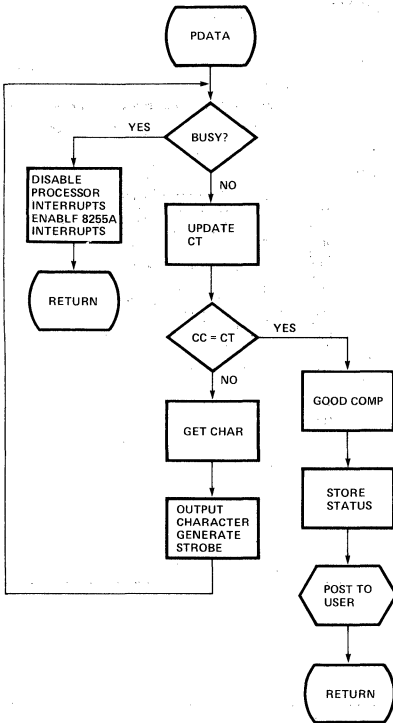
;*****
; PRINTER INTERRUPT SERVICE ROUTINE
; ALL REGISTERS SAVED AND RESTORED
;*****
PINT:
3030 F5 PUSH PSW ; SAVE PSW
3031 C5 PUSH B ; SAVE REGISTER PAIR B AND C
3032 D5 PUSH D ; SAVE REGISTER PAIR D AND E
3033 D5 PUSH H ; SAVE REGISTER PAIR H AND L
;*****
; POLL INTERRUPT SOURCE - SEE IF 8255
;*****
3034 DBF6 IN PORTE ; GET STATUS OF DEVICE
3036 E608 ANI INTRA ; SEE IF INT
3038 CA5230 JZ PPOLL ; NO - BRANCH TO POLL OTHER DEVICES IF ANY
303B 3E0C MVI A,10H ; GET 8255 INT DISABLE CONTROL WORD
303D D3F7 OUT CWR ; DISABLE DEVICE INTERRUPTS
303F FB EI ; ENABLE PROCESSOR INTERRUPTS
3040 2AA130 LHLD PIPRG ; GET CONTROL BLOCK ADDRESS
3043 AF XRA A ; CLEAR A REG
3044 BC CMP H ; SEE IF PRINT IN PROGRESS
3045 CA5530 JZ PIER1 ; NO - BRANCH TO ERROR ROUTINE
3048 EB XCHG PDATA ; PRINT DATA
3049 CD5830 CALL PDATA
;*****
; RESTORE REGISTERS AND RETURN FROM INTERRUPT
;*****
PRTN:
304C E1 POP H ; RESTORE REGISTER PAIR H AND L
304D D1 POP D ; RESTORE REGISTER PAIR D AND E
304E C1 POP B ; RESTORE REGISTER PAIR B AND C
304F F1 POP PSW ; RESTORE PSW
3050 FB EI ; ENABLE PROCESSOR INTERRUPTS
3051 C9 RET ; RETURN TO INTERRUPTED PROCESS
;*****
; POLL OTHER DEVICES IF ANY
; IF NO OTHER DEVICES TO POLL - USER SUPPLIED ERROR
; RECOVERY ROUTINE.
;*****
PPOLL:
3052 C34C30 JMP PRTN ; RETURN
;*****
; ERROR - INTERRUPT FROM IDLE DEVICE
; USER SUPPLIED ERROR RECOVERY ROUTINE
;*****
PIER1:
3055 C34C30 JMP PRTN ; RETURN
  
```



# APPLICATIONS

ISIS 8080 MACRO ASSEMBLER, V1.0  
 PRINTER OUTPUT DATA ROUTINE

PAGE 5



```

*****
;
; PRINTER OUTPUT DATA ROUTINE
;
; CONTROL BLOCK ADDRESS IN D AND E REG
;
*****
PDATA:
3058 DBF6      IN      PORTC      ; GET STATUS OF DEVICE
305A E680      ANI      LPSBY     ; SEE IF BUSY (BUFFER FULL)
305C CA8430    JZ       PD10      ; IF BUSY - BRANCH
305F 210400    LLI      H,CBCT    ; GET INDEX TO CT
3062 19        D        D         ; COMPUTE ADDRESS OF CT
3063 7E        MOV      A,M       ; GET CT
3064 34        INR      M         ; INC CT
3065 2B        DCR      H         ; DEC TO CC
3066 BE        CME      M         ; SEE IF EQUAL
3067 CA8A30    JZ       PCOMP     ; IF EQUAL - DONE GO TELL USER
306A 210100    LLI      H,CBUF    ; GET INDEX TO BUFFER ADDRESS
306D 19        DAD      D         ; COMPUTE ADDRESS OF BUFFER ADDRESS
306E D5        PUSH     D         ; SAVE D AND E REGISTERS
306F 5E        MOV      E,M       ; GET LSB OF BUFFER ADDRESS
3070 23        INK      H         ; INC TO NEXT BYTE
3071 56        MOV      D,M       ; GET BUFFER MSB
3072 2600     MVI      H,OOH     ; CLEAR H REG
3074 6F        MOV      L,A         ; GET CT
3075 19        DAD      D         ; COMPUTE CHARACTER ADDRESS
3076 7E        MOV      A,M       ; GET CHARACTER
3077 D3F4     OUT      PORTA     ; OUTPUT CHARACTER TO PRINTER
3079 3E00     MVI      A,STOB    ; RESET DATA STROBE (LOW TRUE SIGNAL)
307B D3F7     OUT      CWR       ; SET DATA STROBE
307D 3C        INR      A         ; GENERATE SET CONTROL WORD
307E D3F7     OUT      CWR       ; SET DATA STROBE
3080 D1        POP      D         ; RESTORE CONTROL BLOCK ADDRESS
3081 C35830   JMP      PDATA     ; LOOP UNTIL BUSY
;
*****
; PRINTER BUSY - RETURN
;
*****
PD10:
3084 F3        DI          ; DISABLE INTERRUPTS
3085 3E0D     MVI      A, IEN     ; ENABLE DEVICE INTERRUPTS
3087 D3F7     OUT      CWR     ; SET INTERRUPT ENABLE
3089 C9        RET
;
*****
; POST GOOD COMPLETION TO USER
;
*****
PCOMP:
308A 3B00     MVI      A,STGD    ; GET GOOD STATUS CODE
308F AF      CALL     POST     ; POST TO USER
3090 32A230   STA      PIPRG+1   ; CLEAR COMMAND IN PROGRESS ADDRESS
3093 C9      RET
;
*****
; POST TO USER COMPLETION ROUTINE
;
; INPUTS : STATUS CODE IN A REG
; CONTROL BLOCK ADDRESS IN D AND E REG
; OUTPUTS: PASSES CONTROL TO USER COMPLETION ADDRESS
; SPECIFIED IN CONTROL BLOCK
; WITH CONTROL BLOCK ADDRESS IN D AND E
;
; A,H,L,B,C REG MODIFIED
;
*****
POST:
3094 EB      XCHG      M,A       ; UPDATE STATUS
3095 77      MOV      XCHG
3096 EB      XCHG
3097 210500   LLI      H,CBCHP    ; GET INDEX TO COMPLETION ADDRESS
309A 19      DAD      D         ; COMPUTE ADDRESS
309B 4E      MOV      C,M       ; GET LSB OF COMPLETION ADDRESS
309C 23      INK      H         ; INC TO NEXT BYTE
309D 46      MOV      E,M       ; GET MSB OF COMPLETION ADDRESS
309E C3      PUSH     B         ; PUSH ADDRESS INTO STACK
309F C9      RET
;
*****
; DATA AND TABLES
;
*****
PIPRG: DW 0 ; IN PROGRESS CONTROL BLOCK ADDRESS
; IF DATA = 0 NO CONTROL BLOCK IN PROGRESS
; IF DATA NOT EQUAL TO ZERO CONTROL BLOCK IN PROGRESS
;
*****
END OF MODE ONE EXAMPLE
*****
END
    
```

## MODE 2 – 8080 TO 8080 INTERFACE

Due to the drastic reduction of hardware costs, system designs which utilize multiple CPU Modules are becoming more common. An 8080 may be configured as a master CPU and used to control multiple 8080 slave modules which act as intelligent I/O controllers. When multiple CPUs are utilized, a method of processor intercommunication must be supported. Figure 20 is a block diagram of one method of implementing a master/slave interface through the use of the 8255A Mode 2 bidirectional bus.

### Hardware Discussion

Two complete 8080 systems are required for this example. Intel's SBC 80/10 OEM board is used as the master CPU module and Intel's SDK 80 board is used as the slave CPU. The SBC 80/10 supports an 8255A which is configured in Mode 2. The 8255A is selected through the use of a decoded select scheme. Through the use of the 8228 RST 7 interrupt feature, a simple interrupt structure is supported. The SDK 80 is configured without interrupts for this example. The external logic required for this example is associated with the slave CPU. Simple logic is implemented which allows the slave CPU to generate the  $\overline{\text{ACK}}$  and  $\overline{\text{STB}}$  signals required to READ from and WRITE to the 8255A bidirectional bus with a single I/O instruction.

The system shown in Figure 20 utilizes SSI logic to read the 8255A IBF and OBF signals. If two spare 8255A input lines are available they could be used to input the IBF and OBF signals and eliminate the SSI logic.

### Software Discussion

Two sets of software are required to support the processor to processor interface. The master resident software which follows conforms to the simple interrupt driven software structure outlined previously. The initialization routine issues the Mode 2 control word to the 8255A after device reset. The command processor accepts READ/WRITE control blocks which provide a simple user interface for transferring data to/from the slave CPU. The master software is capable of processing both a read and a write control block simultaneously. The slave resident software shown at the end of this example utilizes the status driven approach.

### Summary/Conclusions

It is important to note that this design may be expanded to include more slave CPUs by simply adding another 8255A to the master module for each slave. The software drivers discussed address only the passing of data between the two processors. Specific applications generally dictate a software protocol be implemented for information transfer.

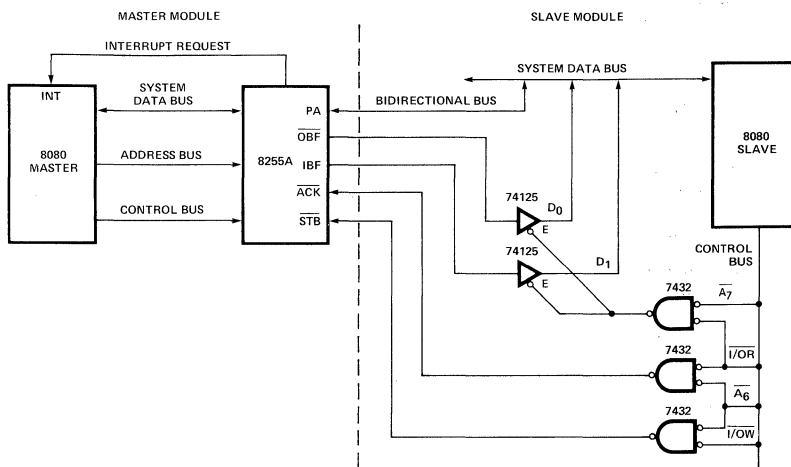
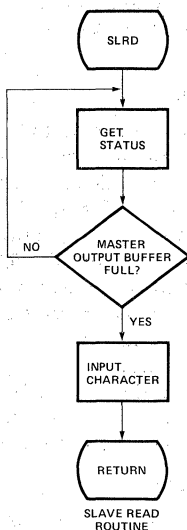


Figure 20. Interface Block Diagram

# APPLICATIONS

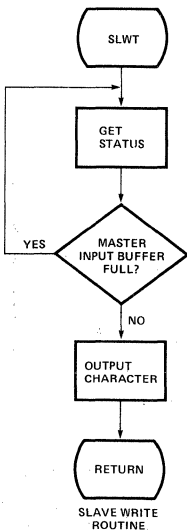


ISIS 8080 MACRO ASSEMBLER, V1.0  
MODE TWO EXAMPLE - SLAVE SOFTWARE

PAGE 1

```

*****
;
; TITLE 'MODE TWO EXAMPLE - SLAVE SOFTWARE
;
; 8080 MASTER TO 8080 SLAVE INTERFACE
; - SLAVE SOFTWARE -
; MODE TWO EXAMPLE
;
*****
;
; PROGRAM EQUATES
;
*****
00BF PDATA EQU 0BFH ; INTERPROCESSOR DATA PORT
007F PSTS EQU 07FH ; STATUS
;
; BUFFER STATUS MASKS
;
*****
0001 OBF EQU 01H ; OUTPUT BUFFER FULL
0002 IBF EQU 02H ; INPUT BUFFER FULL
;
; PROGRAM ORIGIN
;
*****
3000 ORG 03000H
;
;
; SLAVE READ ROUTINE
;
; INPUTS: NONE
; OUTPUTS: CHARACTER READ IN C-REGISTER
;
; A,C REG MODIFIED
;
*****
SLRD:
3000 DB7F IN PSTS ; GET STATUS
3002 E601 ANI OBF ; SEE IF BUFFER FULL
3004 C20030 JNZ SLRD ; NO - LOOP UNTIL FULL
3007 DBE8 IN PDATA ; GET CHARACTER
3009 4F MOV C,A ; PLACE IN C-REG
300A C9 RET ; RETURN TO CALLER
  
```



ISIS 8080 MACRO ASSEMBLER, V1.0  
MODE TWO EXAMPLE - SLAVE SOFTWARE

PAGE 2

```

*****
;
; SLAVE WRITE ROUTINE
;
; INPUTS: CHARACTER TO WRITE IN C-REGISTER
; OUTPUTS: NONE
;
; A REG MODIFIED
;
*****
SLWT:
3008 DB7F IN PSTS ; GET STATUS
300D E602 ANI IBF ; SEE IF BUFFER FULL
300F C20E30 JNZ SLWT ; YES - LOOP UNTIL EMPTY
3012 79 MOV A,C ; GET DATA CHARACTER
3013 DBE8 OUT PDATA ; OUTPUT DATA
3015 C9 RET ; RETURN TO CALLER
;
*****
END OF SLAVE SOFTWARE DRIVER
*****
END
0000
  
```

# APPLICATIONS

ISIS 8080 MACRO ASSEMBLER, V1.0  
MODE TWO EXAMPLE - MASTER SOFTWARE

PAGE 1

```

;*****
;
; TITLE 'MODE TWO EXAMPLE - MASTER SOFTWARE'
;*****
;
; ROBO MASTER TO ROBO SLAVE INTERFACE
; - MASTER SOFTWARE -
; MODE TWO EXAMPLE
;*****
;
; PROGRAM EQUATES
;*****
00E4 PORTA EQU 0E4H ; 8255 PORT A
PORTB EQU 0E5H ; 8255 PORT B
00E6 PORTC EQU 0E6H ; 8255 PORT C
00E7 CWR EQU 0E7H ; 8255 CONTROL WORD REGISTER
0038 RST7 EQU 038H ; RESTART 7 ADDRESS
;*****
;
; INITIALIZATION CONTROL WORD
;
; USED TO CONFIGURE THE 8255 AS FOLLOWS:
;
; PORT A - MODE 2 BIDIRECTIONAL BUS
; PORT B - INPUT MODE 0 (NOT USED)
; REMAINING PORT C LINES - INPUT MODE (NOT USED)
;*****
00CB ICW EQU 11001011B ; INITIALIZATION CONTROL WORD
;*****
; 8255 ENABLE/DISABLE INTERRUPT CONTROL WORDS
;*****
000D IENI EQU 00001101B ; ENABLE INPUT INTERRUPTS
0009 IENO EQU 00001001B ; ENABLE OUTPUT INTERRUPTS
000C IDNI EQU 00001100B ; DISABLE INPUT INTERRUPTS
0008 IDNO EQU 00001000B ; DISABLE OUTPUT INTERRUPTS
;*****
; STATUS EQUATES
;*****
0005 INTRA EQU 05H ; INTERRUPT REQUEST
0060 OBFA EQU 60H ; OUTPUT BUFFER FULL
0020 IBFA EQU 20H ; INPUT BUFFER FULL

```

ISIS 8080 MACRO ASSEMBLER, V1.0  
MODE TWO EXAMPLE - MASTER SOFTWARE

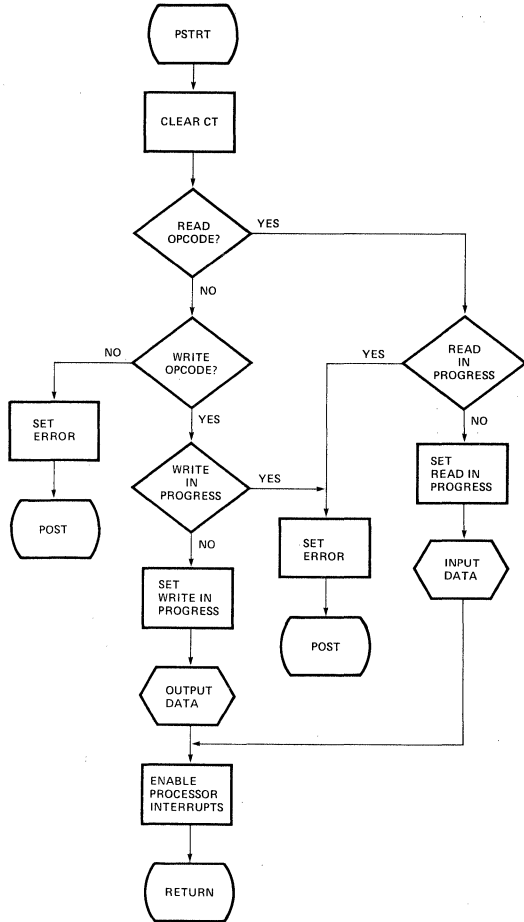
PAGE 2

```

;*****
; CONTROL BLOCK EQUATES
;*****
0000 CBST EQU 00H ; STATUS BYTE
CBOP EQU 01H ; OPCODE = 0 READ
;
; OPCODE = 1 WRITE
0002 CBUF EQU 02H ; BUFFER ADDRESS
0004 CBCC EQU 04H ; CHARACTER COUNT
0005 CBCT EQU 05H ; CHARACTER TRANSFERRED COUNT
0006 CBCMP EQU 06H ; COMPLETION SERVICE ADDRESS
;*****
; OPCODE EQUATES
;*****
0000 OPRD EQU 00H ; READ OPCODE
0001 OPRT EQU 01H ; WRITE OPCODE
;*****
; COMPLETION STATUS EQUATES
;*****
0000 STGD EQU 00H ; GOOD COMPLETION
0001 STE1 EQU 01H ; ERROR - COMMAND ALREADY IN PROGRESS
0002 STE2 EQU 02H ; ERROR - INVALID OPCODE
;*****
; SET UP INTERRUPT VECTOR
;*****
0038 ORG RST7
0038 C34630 JMP FINT ; JUMP TO INTERRUPT SERVICE ROUTINE
;*****
; PROGRAM ORIGIN
;*****
3000 ORG 03000H
;*****
;
; INITIALIZATION ROUTINE
;
; A REGISTER MODIFIED
;*****
;*****
INIT:
3000 36CB MVI A,ICW ; GET MODE CONTROL WORD
3002 03E7 OUT CWR ; OUTPUT TO CONTROL WORD REGISTER
3004 C9 RET ; RETURN TO CALLER

```

# APPLICATIONS



ISIS 8080 MACRO ASSEMBLER, V1.0  
COMMAND PROCESSOR

PAGE 3

```

;*****
;
; COMMAND PROCESSOR
;
; INPUTS: CONTROL BLOCK ADDRESS IN D AND E REGISTERS
;
; OUTPUTS: START I/O OR ERROR STATUS IN CONTROL BLOCK
;         A,H,L
;         MODIFIED
;
;*****
PSTRT:
3008 210500 LXI H,CBCT ; GET INDEX TO CT
3008 19 D ; COMPUTE ADDRESS OF CT
3009 3600 MVI M,OPRD ; CLEAR CT
300B 210100 LXI H,CBOP ; GET INDEX TO OPCODE
300E 19 D ; COMPUTE ADDRESS
3010 FE00 MOV A,M ; GET OPCODE
3012 CA2430 CPI OOH ; SEE IF READ
3015 FE01 JZ PSRD ; YES - GO PROCESS READ
3017 CA3530 JZ PSWT ; YES - GO PROCESS WRITE
;*****
; ERROR - INVALID OPCODE
;*****
301A 3E02 MVI A,STEP ; GET ERROR STATUS CODE
; POST ; CALL COMPLETION ROUTINE
;*****
; ERROR - TRANSACTION ALREADY IN PROGRESS
;*****
PSTE:
301F 3E01 MVI A,STEP1 ; GET ERROR STATUS CODE
3021 C3DC30 JMP POST ; CALL COMPLETION ROUTINE
;*****
; PROCESS READ COMMAND
;*****
PSRD:
3024 3AE430 LDA PRGRD+1 ; GET READ IN PROGRESS ADDRESS
3027 A7 ANA A ; SEE IF READ IN PROGRESS (TEST FOR ZERO)
3028 C21F30 JNZ PSTE ; IF YES - BRANCH
302B EB XCHG
302C 22E930 SHLD PRGRD ; SAVE CONTROL BLOCK ADDRESS
302F EB XCHG
3030 CD7C30 CALL PIN ; START I/O
3033 FB EI ; ENABLE INTERRUPTS
3034 C9 RET ; RETURN TO CALLER
  
```

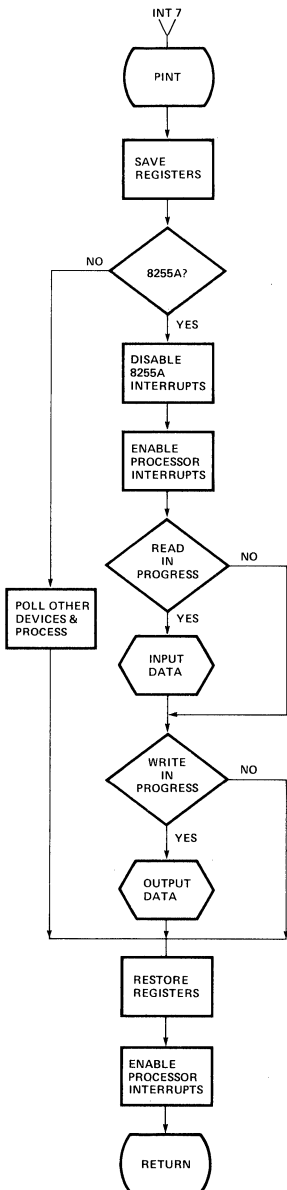
ISIS 8080 MACRO ASSEMBLER, V1.0  
COMMAND PROCESSOR

PAGE 4

```

;*****
;
; PROCESS WRITE COMMAND
;
;*****
PSWT:
3035 3AE330 LDA PRGWT+1 ; GET WRITE IN PROGRESS ADDRESS
3038 A7 ANA A ; SEE IF WRITE IN PROGRESS (TEST FOR ZERO)
3039 C21F30 JNZ PSTE ; IF YES - BRANCH
303C EB XCHG
303D 22E830 SHLD PRGWT ; SAVE CONTROL BLOCK ADDRESS
3040 EB XCHG
3041 CD9C30 CALL POUT ; START I/O
3044 FB EI ; ENABLE INTERRUPTS
3045 C9 RET ; RETURN TO CALLER
  
```

# APPLICATIONS



ISIS 8080 MACRO ASSEMBLER, V1.0  
INTERRUPT SERVICE ROUTINE

PAGE 5

```

*****
;
; INTERRUPT SERVICE ROUTINE
; ALL REGISTERS SAVED AND RESTORED
;
*****
PINT:
3046 F5      PUSH  PSW      ; SAVE PSW
3047 C5      PUSH  B        ; SAVE REGISTER PAIR B AND C
3048 D5      PUSH  D        ; SAVE REGISTER PAIR D AND E
              PUSH  H        ; SAVE REGISTER PAIR H AND L
;
*****
; POLL INTERRUPT SOURCE - SEE IF 8255
;
304A DBE6   IN      PORTC   ; GET STATUS OF DEVICE
304C E608   ANI      INTRA   ; SEE IF INT
304E CA7630 JZ      PPOLL   ; NO - BRANCH TO POLL OTHER DEVICES IF ANY
3051 3E0C   MVI      A,LDWI  ; GET INPUT INT DISABLE CONTROL WORD
3053 D3E7   OUT      CWR     ; DISABLE DEVICE INTERRUPTS
3055 3E08   MVI      A,LDNO  ; GET OUTPUT INT DISABLE CONTROL WORD
3057 D3E7   OUT      CWR     ; DISABLE DEVICE INTERRUPTS
3059 FE     EI           ; ENABLE PROCESSOR INTERRUPTS
305A 2AE930 LHLD   PRGDR   ; GET READ CONTROL BLOCK
305D AF     XRA      A      ; CLEAR A REG
305E BC     CMP      H      ; SEE IF READ IN PROGRESS
305F CA6530 JZ      PINT1   ; NO - BRANCH
3062 CD7C30 CALL   PIN      ; DO INPUT
;
PINT1:
3065 2AB330 LHLD   PRGWT   ; GET WRITE CONTROL BLOCK
3068 AF     XRA      A      ; CLEAR A REG
3069 BC     CMP      H      ; SEE IF WRITE IN PROGRESS
306A CA7030 JZ      PRTN    ; NO - BRANCH
306D CD9C30 CALL   POUT    ; DO OUTPUT
;
*****
; RESTORE REGISTERS AND RETURN FROM INTERRUPT
;
*****
PRTN:
3070 E1     POP      H        ; RESTORE REGISTER PAIR H AND L
3071 D1     POP      B        ; RESTORE REGISTER PAIR D AND E
3072 C1     POP      B        ; RESTORE REGISTER PAIR B AND C
3073 F1     POP      PSW     ; RESTORE PSW
3074 FB     EI           ; ENABLE PROCESSOR INTERRUPTS
3075 C9     RET          ; RETURN TO INTERRUPTED PROCESS
  
```

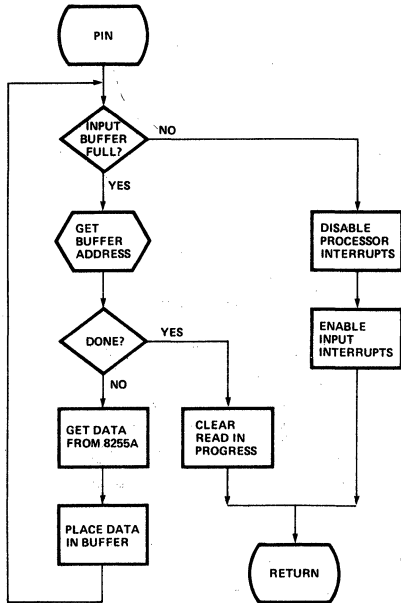
ISIS 8080 MACRO ASSEMBLER, V1.0  
INTERRUPT SERVICE ROUTINE

PAGE 6

```

*****
;
; POLL OTHER DEVICES IF ANY
; IF NO OTHER DEVICES TO POLL - USER SUPPLIED ERROR
; RECOVERY ROUTINE.
;
*****
PPOLL:
3076 C37030 JMP      PRTN    ; RETURN
;
*****
; ERROR - INTERRUPT FROM IDLE DEVICE
; USER SUPPLIED ERROR RECOVERY ROUTINE
;
PIERR1:
3079 C37030 JMP      PRTN    ; RETURN
  
```

# APPLICATIONS

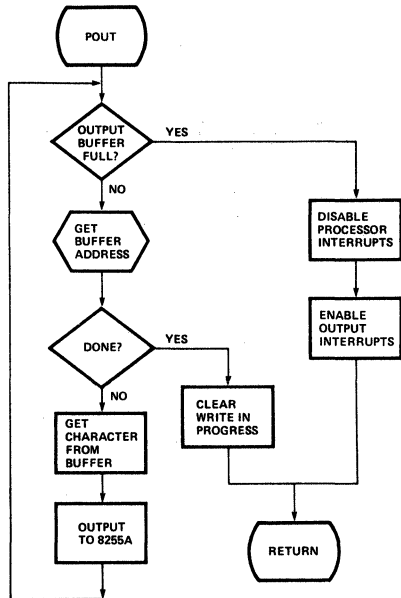


ISIS 8080 MACRO ASSEMBLER, V1.0  
INPUT DATA ROUTINE

PAGE 7

```

:*****
:***** INPUT DATA ROUTINE
:*****
PIN:
307C D8E6      IN   PORTC   ; GET STATUS OF DEVICE
307E E620      ANI   IBFA   ; SEE IF INPUT BUFFER FULL
3080 CA9630    JZ    PRTI   ; NO - BRANCH
3083 C0BC30    CALL  CBFA   ; GET ADDRESS IN BUFFER
3086 DA8F30    JC    PIDON  ; IF DONE - BRANCH
3089 DBE4      IN   PORTA   ; GET DATA
308B 77        MOV   M,A    ; PLACE IN BUFFER
308C C37C30    JMP   FIN    ; LOOP
:*****
:***** END OF INPUT TRANSACTION
:*****
PIDON:
308F AF        XRA   A      ; CLEAR A
3090 32E430    STA  PRD+1 ; CLEAR READ IN PROGRESS
3093 C3B630    JMP  PRTI   ; RETURN
:*****
:***** RETURN FROM INPUT
:*****
PRTI:
3096 F3        DI          ; DISABLE PROCESSOR INTERRUPTS
3097 3E0D      MVI   A,IENI ; GET ENABLE INPUT INTERRUPTS CONTROL WORD
3099 D3E7      OUT   CWR  ; OUTPUT TO CONTROL WORD REGISTER
309B C9        RET          ; RETURN TO CALLER
    
```



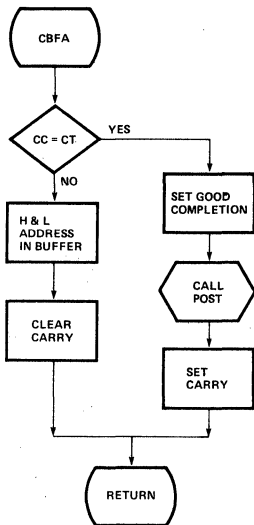
ISIS 8080 MACRO ASSEMBLER, V1.0  
OUTPUT DATA ROUTINE

PAGE 8

```

:*****
:***** OUTPUT DATA ROUTINE
:*****
POUT:
309C D8E6      IN   PORTC   ; GET PORT STATUS
309E E620      ANI   IBFA   ; SEE IF OUTPUT BUFFER FULL
30A3 C0BC30    JNZ  PRTO   ; YES - BRANCH
30A6 DA8F30    CALL  CBFA   ; SET UP ADDRESS OF DATA
30A9 7E        JC    PIDON  ; IF DONE - BRANCH
30AA D3E4      MOV   A,M    ; GET DATA FROM BUFFER
30AC C39C30    OUT   PORTA ; OUTPUT DATA
:*****
:***** END OF OUTPUT TRANSACTION
:*****
PIDON:
30AF AF        XRA   A      ; CLEAR A REG
30B0 32E330    STA  PRTO+1 ; CLEAR WRITE IN PROGRESS
30B3 C3B630    JMP  PRTO   ; RETURN
:*****
:***** RETURN FROM OUTPUT
:*****
PRTO:
30B6 F3        DI          ; DISABLE PROCESSOR INTERRUPTS
30B7 3E09      MVI   A,IENO ; GET ENABLE OUTPUT INTERRUPTS CONTROL WORD
30B9 D3E7      OUT   CWR  ; OUTPUT TO CONTROL WORD REGISTER
30BB C9        RET          ; RETURN TO CALLER
    
```

# APPLICATIONS



Setup Buffer Address Subroutine

ISIS 8080 MACRO ASSEMBLER, V1.0  
COMPUTE BUFFER ADDRESS ROUTINE

PAGE 9

```

;*****
;      COMPUTE BUFFER ADDRESS ROUTINE
;*****
CBFA:
308C 210500 LXI H,CBCT ; GET INDEX TO CT
308F 19 DAD D ; COMPUTE ADDRESS OF CT
3090 7E MOV A,M ; GET CT
30C1 34 INR M ; INC CT
30C2 2B DCX H ; DEC TO CC
30C3 BE CMP H ; SEE IF EQUAL
30C4 CAD530 JZ ; IF EQUAL - DONE GO TELL USER
30C7 210200 LXI H,CBUF ; GET INDEX TO BUFFER ADDRESS
30CA 19 DAD D ; COMPUTE ADDRESS OF BUFFER ADDRESS
30CB 05 PUSH D ; SAVE D AND E REGISTERS
30CC 5E MOV E,M ; GET LSB OF BUFFER ADDRESS
30CD 23 INX H ; INC TO NEXT BYTE
30CE 56 MOV D,M ; GET BUFFER MSB
30CF AC XRA H ; CLEAR H REG
30D0 6F MOV L,A ; GET CT
30D1 19 DAD D ; COMPUTE CHARACTER ADDRESS
30D2 D1 POP D ; RESTORE CONTROL BLOCK ADDRESS
30D3 AF XRA A ; CLEAR CARRY
30D4 C9 RET ; RETURN TO CALLER
  
```

ISIS 8080 MACRO ASSEMBLER, V1.0  
POST TO USER COMPLETION ROUTINE

PAGE 10

```

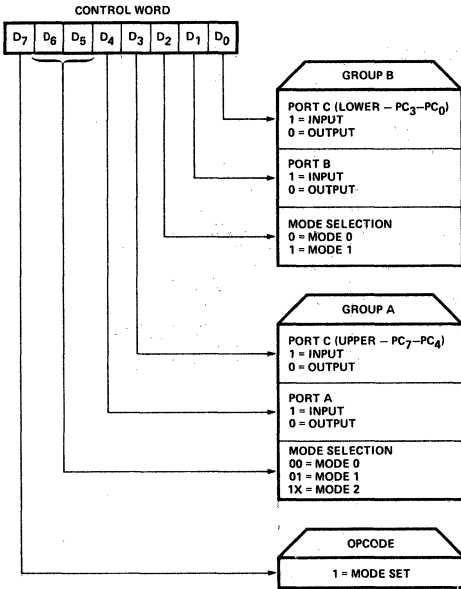
;*****
;      POST GOOD COMPLETION TO USER
;*****
PCOMP:
30D5 3E00 MVI A,STGD ; GET GOOD STATUS CODE
30D7 CDDC30 CALL POST ; CALL USER ROUTINE
30DA 37 STC ; SET CARRY
30DB C9 RET ; RETURN TO CALLER

;*****
;      POST TO USER COMPLETION ROUTINE
;
;      INPUTS : STATUS CODE IN A REG
;              CONTROL BLOCK ADDRESS IN D AND E REG
;      OUTPUTS: PASSES CONTROL TO USER COMPLETION ADDRESS
;              SPECIFIED IN CONTROL BLOCK
;*****
POST:
30DC EB XCHG ;
30DD 77 MOV M,A ; UPDATE STATUS
30DE EB XCHG ;
30DF 210600 LXI H,CBCHP ; GET INDEX TO COMPLETION ADDRESS
30E2 19 DAD D ; COMPUTE ADDRESS
30E3 4E MOV C,M ; GET LSB OF COMPLETION ADDRESS
30E4 23 INX H ; INC TO NEXT BYTE
30E5 46 MOV B,M ; GET MSB BYTE OF COMPLETION ADDRESS
30E6 C5 PUSH B ; PUSH ADDRESS INTO STACK
30E7 C9 RET ; PASS CONTROL TO USER ROUTINE
30E8 C9 RET ; RETURN TO CALLER

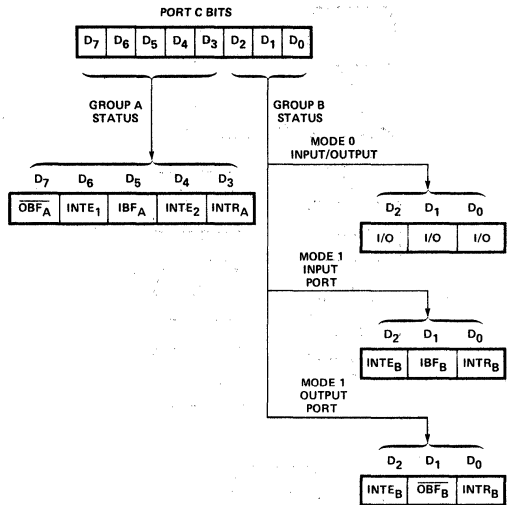
;*****
;      DATA AND TABLES
;
;      IF DATA NON ZERO CONTROL BLOCK IN PROGRESS
;*****
30E9 0000 PRGRD: DW 0 ; IN PROGRESS READ CONTROL BLOCK
30EB 0000 PRGWT: DW 0 ; IN PROGRESS WRITE CONTROL BLOCK

;*****
;      END OF MASTER SOFTWARE DRIVER
;*****
0000 END
  
```

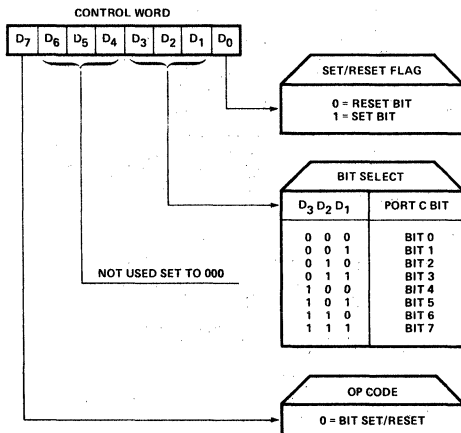




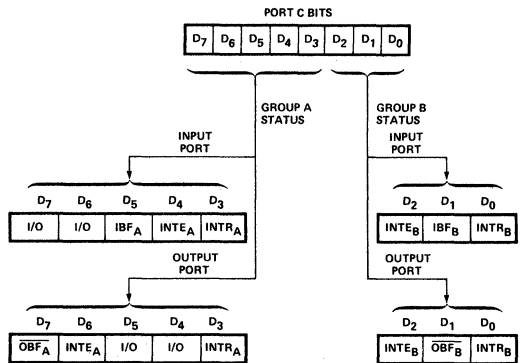
MODE CONTROL WORD



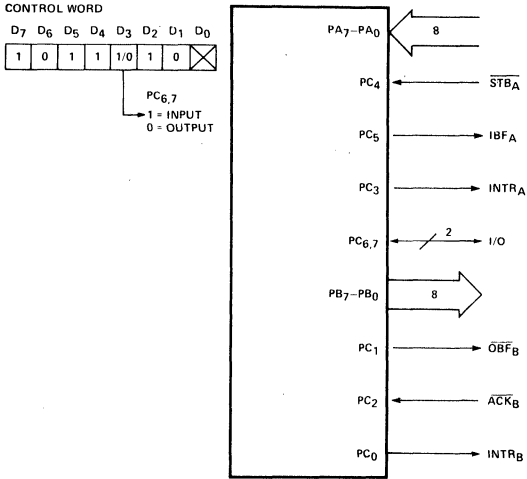
MODE 1 STATUS WORD



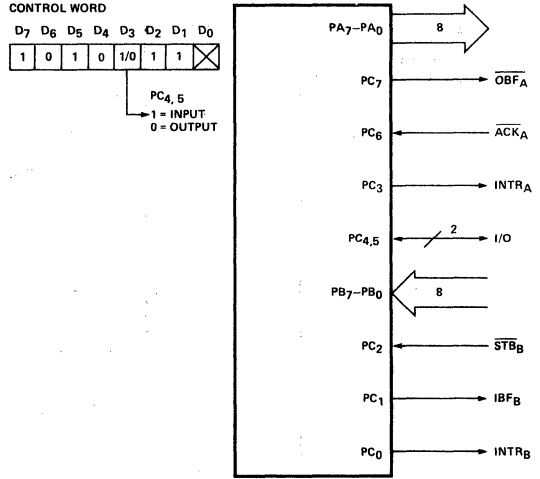
BIT SET/RESET CONTROL WORD



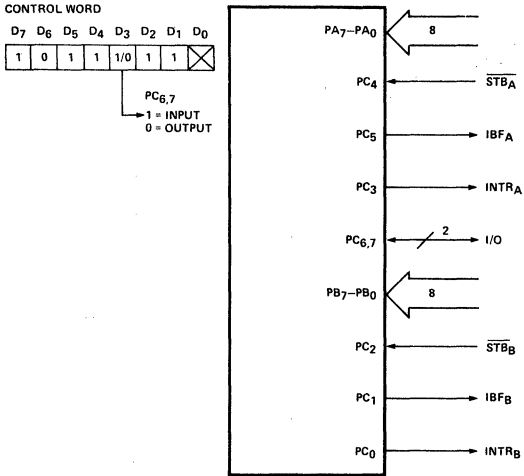
MODE 2 STATUS WORD



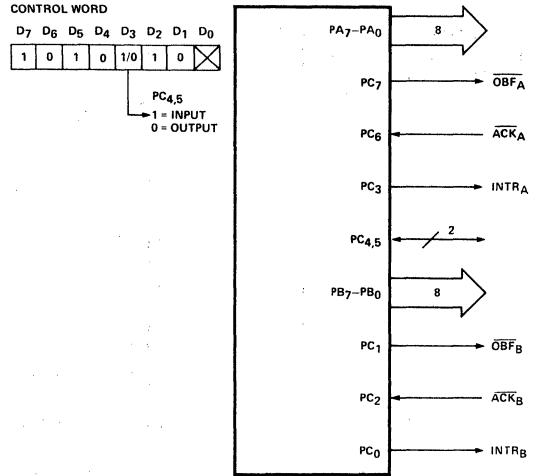
PORT A - STROBED INPUT  
PORT B - STROBED OUTPUT



PORT A - STROBED OUTPUT  
PORT B - STROBED INPUT

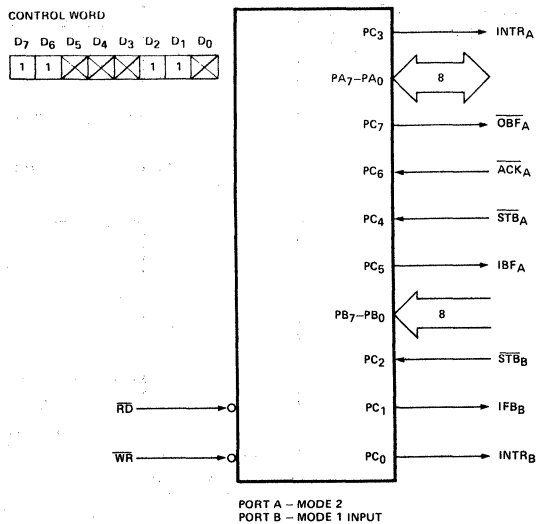
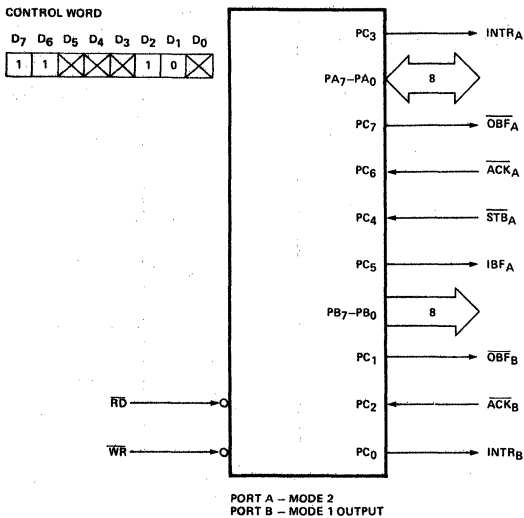
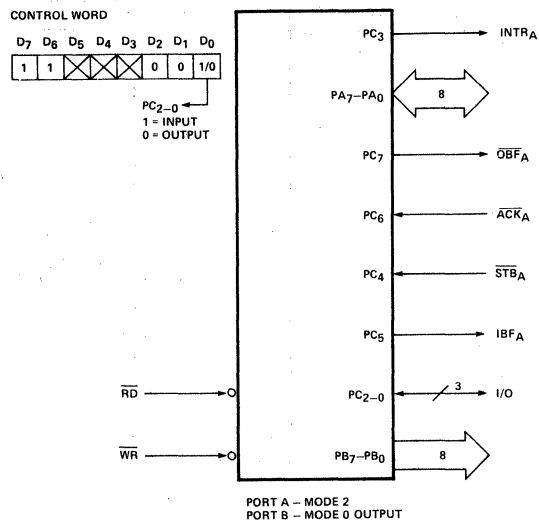
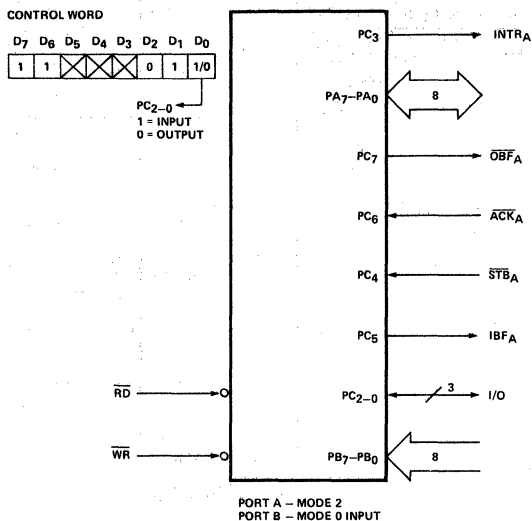


PORT A - STROBED INPUT  
PORT B - STROBED INPUT



PORT A - STROBED OUTPUT  
PORT B - STROBED OUTPUT

# MODE 2 CONFIGURATIONS



---

# **A Low Cost CRT Terminal Using the 8275**

## **Contents**

<b>1. INTRODUCTION</b>	<b>2-440</b>
<b>2. CRT BASICS</b>	<b>2-440</b>
<b>3. 8275 DESCRIPTION</b>	<b>2-443</b>
3.1 CRT Display Refreshing	
3.2 CRT Timing	
3.3 Special Functions	
<b>4. DESIGN BACKGROUND</b>	<b>2-447</b>
4.1 Design Philosophy	
4.2 Using the 8275 without DMA	
<b>5. CIRCUIT DESCRIPTION</b>	<b>2-449</b>
5.1 Scope of the Project	
5.2 System Target Specifications	
5.3 Hardware Descriptions	
5.4 System Operation	
5.5 System Timing	
<b>6. SYSTEM SOFTWARE</b>	<b>2-454</b>
6.1 Software Overview	
6.2 System Memory Organization	
6.3 Memory Pointers and Scrolling	
6.4 Software Timing	
<b>APPENDIX 7.1</b>	<b>2-458</b>
CRT Terminal Schematics	
<b>APPENDIX 7.2</b>	<b>2-460</b>
Keyboard Interface	
<b>APPENDIX 7.3</b>	<b>2-461</b>
Escape/Control/Display Character Summary	
<b>APPENDIX 7.4</b>	<b>2-462</b>
PROM Decoding	
<b>APPENDIX 7.5</b>	<b>2-463</b>
Character Generator	
<b>APPENDIX 7.6</b>	<b>2-464</b>
HEX Dump of Character Generator	
<b>APPENDIX 7.7</b>	<b>2-465</b>
Composite Video	
<b>APPENDIX 7.8</b>	<b>2-465</b>
Software Listings	

## 1. INTRODUCTION

The purpose of this application note is to provide the reader with the design concepts and factual tools needed to integrate Intel peripherals and microprocessors into a low cost raster scan CRT terminal. A previously published application note, AP-32, presented one possible solution to the CRT design question. This application note expands upon the theme established in AP-32 and demonstrates how to design a functional CRT terminal while keeping the parts count to a minimum.

For convenience, this application note is divided into seven general sections:

1. Introduction
2. CRT Basics
3. 8275 Description
4. Design Background
5. Circuit Description
6. Software Description
7. Appendix

There is no question that microprocessors and LSI peripherals have had a significant role in the evolution of CRT terminals. Microprocessors have allowed design engineers to incorporate an abundance of sophisticated features into terminals that were previously mere slaves to a larger processor. To complement microprocessors, LSI peripherals have reduced component count in many support areas. A typical LSI peripheral easily replaces between 30 and 70 SSI and MSI packages, and offers features and flexibility that are usually not available in most hardware designs. In addition to replacing a whole circuit board of random logic, LSI circuits also reduce the cost and increase the reliability of design. Fewer interconnects increases mechanical reliability and fewer parts decreases the power consumption and hence, the overall reliability of the design. The reduction of components also yields a circuit that is easier to debug during the actual manufacturing phase of a product.

Until the era of advanced LSI circuitry, a typical CRT terminal consisted of 80 to 200 or more SSI and MSI packages. The first microprocessors and peripherals dropped this component count to between 30 and 50 packages. This application note describes a CRT terminal that uses 20 packages.

## 2. CRT BASICS

The raster scan display gets its name from the fact that the image displayed on the CRT is built up by generating a series of lines (raster) across the face of the CRT. Usually, the beam starts in the upper left hand corner of the display and simultaneously moves left to right and top to bottom to put a series

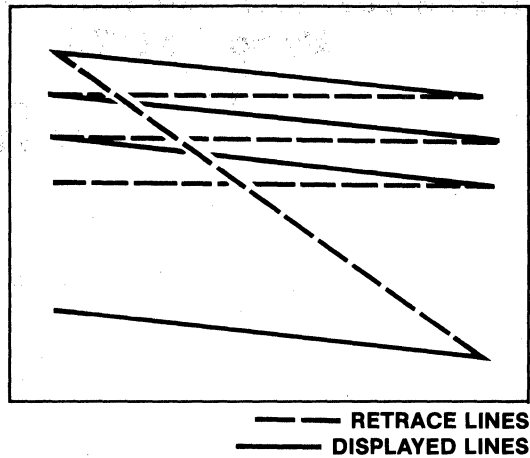


Figure 2-1. Raster Scan

of zig-zag lines on the screen (Fig. 2.1). Two simultaneously operating independent circuits control the vertical and horizontal movement of the beam.

As the electron beam moves across the face of the CRT, a third circuit controls the current flowing in the beam. By varying the current in the electron beam the image on the CRT can be made to be as bright or as dark as the user desires. This allows any desired pattern to be displayed.

When the beam reaches the end of a line, it is brought back to the beginning of the next line at a rate that is much faster than was used to generate the line. This action is referred to as "retrace". During the retrace period the electron beam is usually shut off so that it doesn't appear on the screen.

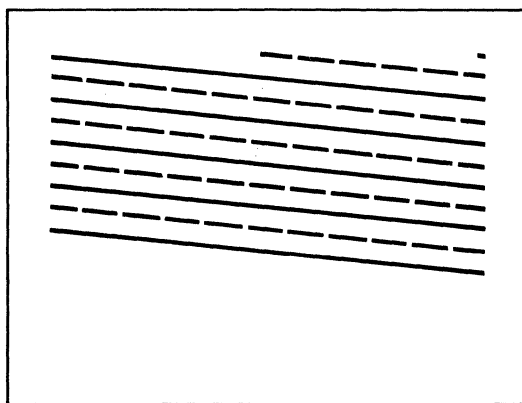
As the electron beam is moving across the screen horizontally, it is also moving downward. Because of this, each successive line starts slightly below the previous line. When the beam finally reaches the bottom right hand corner of the screen, it retraces vertically back to the top left hand corner. The time it takes for the beam to move from the top of the screen to the bottom and back again to the top is usually referred to as a "frame". In the United States, commercial television broadcast use 15,750 Hz as the horizontal sweep frequency (63.5 microseconds per horizontal line) and 60 Hz as the vertical sweep frequency or "frame" (16.67 milliseconds per vertical frame).

Although, the 60 Hz vertical frame and the 15,750 Hz horizontal line are the standards used by commercial broadcasts, they are by no means the only frequency at which CRT's can operate. In fact, many CRT displays use a horizontal scan that is around 18 KHz to 22 KHz and some even exceed 30 KHz. As the

horizontal frequency increases, the number of horizontal lines per frame increases. Hence, the resolution on the vertical axis increases. This increased resolution is needed on high density graphic displays and on special text editing terminals that display many lines of text on the CRT.

Although many CRT's operate at non-standard horizontal frequencies, very few operate at vertical frequencies other than 60 Hz. If a vertical frequency other than 60 Hz is chosen, any external or internal magnetic or electrical variations at 60 Hz will modulate the electron beam and the image on the screen will be unstable. Since, in the United States, the power line frequency happens to be 60 Hz, there is a good chance for 60 Hz interference to exist. Transformers can cause 60 Hz magnetic fields and power supply ripple can cause 60 Hz electrical variations. To overcome this, special shielding and power supply regulation must be employed. In this design, we will assume a standard frame rate of 60 Hz and a standard line rate of 15,750 Hz.

By dividing the 63.5 microsecond horizontal line rate into the 16.67 millisecond vertical rate, it is found that there are 262.5 horizontal lines per vertical frame. At first, the half line may seem a bit odd, but actually it allows the resolution on the CRT to be effectively doubled. This is done by inserting a second set of horizontal lines between the first set (interlacing). In an interlaced system the line sets are not generated simultaneously. In a 60 Hz system, first all of the even-numbered lines are scanned: 0, 2, 4, ... 524. Then all the odd-numbered lines: 1, 3, 5, ... 525. Each set of lines usually contains different data (Fig. 2.2).



————— EVEN FIELD                      RETRACE LINES  
 - - - - - ODD FIELD                      NOT SHOWN

Figure 2-2. Interlaced Scan

Although interlacing provides greater resolution, it also has some distinct disadvantages. First of all, the circuitry needed to generate the extra half horizontal line per frame is quite complex when compared to a noninterlaced design, which requires an integer number of horizontal lines per frame. Next, the overall vertical refresh rate is half that of a noninterlaced display. As a result, flicker may result when the CRT uses high speed phosphors. To keep things as simple as possible, this design uses the noninterlaced approach.

The first thing any CRT controller must do is generate pulses that define the horizontal line timing and the vertical frame timing. This is usually done by dividing a crystal reference source by some appropriate numbers. On most raster scan CRT's the horizontal frequency is very forgiving and can vary by around 500 Hz or so and produce no ill effects. This means that the CRT itself can track a horizontal frequency between 15250 Hz and 16250 Hz, or in other words, there can be 256 to 270 horizontal lines per vertical frame. But, as mentioned earlier, the vertical frequency should be 60 Hz to insure stability.

The characters that are viewed on the screen are formed by a series of dots that are shifted out of the controller while the electron beam moves across the face of the CRT. The circuits that create this timing are referred to as the dot clock and character clock. The character clock is equal to the dot clock divided by the number of dots used to form a character along the horizontal axis and the dot clock is calculated by the following equation:

$$\text{DOT CLOCK (Hz)} = (N + R) * D * L * F$$

where N is the number of displayed characters per row,

R is the number of retrace character time increments,

D is the number of dots per character,

L is the number of horizontal lines per frame and

F is the frame rate in Hz.

In this design N = 80, R = 20, D = 7, L = 270, and F = 60 Hz. If the numbers are plugged in, the dot clock is found to be 11.34 MHz.

The retrace number, R, may vary from system to system because it is used to establish the margins on the left and right hand sides of the CRT. In this particular design R = 20 was empirically found to be optimum. The number of dots per character may vary depending on the character generator used and the number of dot clocks the designer wants to place between characters. This design uses a 5 X 7 dot matrix and allows 2 dot clock periods between characters (see Fig. 2.3); since 5 + 2 equals 7, we find that D = 7.

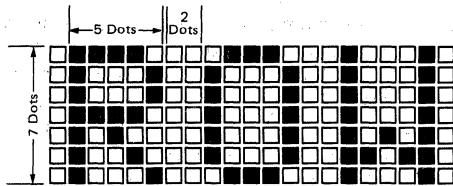


Figure 2-3. 5 X 7 Dot Matrix

The number of lines per frame can be determined by the following equation:

$$L = (H * Z) + V$$

where, H is the number of horizontal lines per character,

Z is the number of character lines per frame and

V is the number of horizontal lines during vertical retrace. In this design, a 5 X 7 dot matrix is to be placed on a 7 X 10 field, so H = 10. Also, 25 lines are to be displayed, so Z = 25. As mentioned before, V = 20. When the numbers are plugged into the equation, L is found to be equal to 270 lines per frame.

The designer should be cautioned that these numbers

are interrelated and that to guarantee proper operation on a standard raster scan CRT, L should be between 256 and 270. If L does not lie within these bounds the horizontal circuits of the CRT may not be able to lock onto the driving signal and the image will roll horizontally. The chosen L of 270 yields a horizontal frequency of 16,200 KHz on a 60 Hz frame and this number is within the 500 Hz tolerance mentioned earlier.

The V number is chosen to match the CRT in much the same manner as the R number mentioned earlier. When the electron beam reaches the bottom right corner of the screen it must retrace vertically to the top left corner. This retrace action requires time, usually between 900-1200 microseconds. To allow for this, enough horizontal sync times must be inserted during vertical retrace. Twenty horizontal sync times at 61.5 microseconds yield a total of 1234.5 microseconds, which is enough time to allow the beam to return to the top of the screen.

The choices of H and Z largely relate to system design preference. As H increases, the character size along the vertical axis increases. Z is simply the number of lines of characters that are displayed and this, of course, is entirely a system design option.

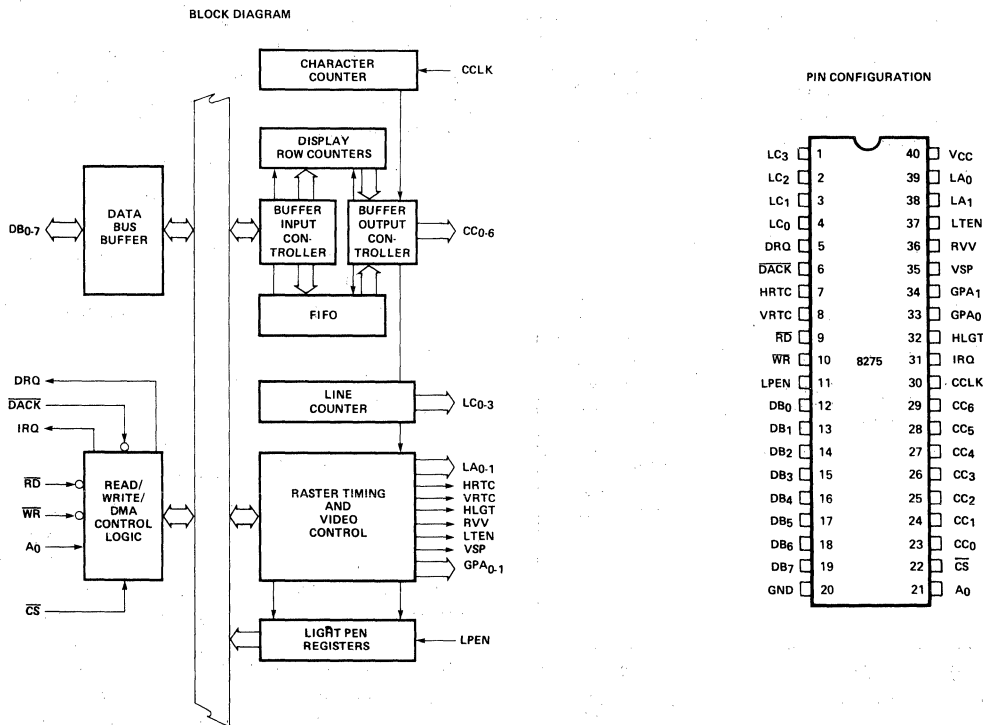


Figure 3-1. 8275 Block Diagram/Pin Configuration

### 3. 8275 DESCRIPTION

A block diagram and pin configuration of the 8275 are shown in Fig. 3.1. The following is a description of the general capabilities of the 8275.

#### 3.1 CRT DISPLAY REFRESHING

The 8275, having been programmed by the designer to a specific screen format, generates a series of DMA request signals, resulting in the transfer of a row of characters from display memory to the 8275's row buffers. The 8275 presents the character codes to an external character generator ROM by using outputs CCO-CC6. External dot timing logic is then used to transfer the parallel output data from the character generator ROM serially to the video input of the CRT. The character rows are displayed on the CRT one line at a time. Line count outputs LC0-LC3 are applied to the character generator ROM to perform the line selection function. The display process is illustrated in Figure 3.2. The entire process is repeated for each display row. At the beginning of the last displayed row, the 8275 issues an interrupt by setting the IRQ output line. The 8275 interrupt output will normally be connected to the interrupt input of the system central processor.

The interrupt causes the CPU to execute an interrupt service subroutine. The service subroutine typically re-initializes DMA controller parameters for the next display refresh cycle, polls the system keyboard controller, and/or executes other appropriate functions. A block diagram of a CRT system implemented with the 8275 CRT Controller is provided in Figure 3.3. Proper CRT refreshing requires that certain 8275 parameters be programmed prior to the beginning of display operation. The 8275 has two types of programming registers, the Command Registers (CREG) and the Parameter Registers (PREG). It also has a Status Register (SREG). The Command Registers may only be written to and the Status Registers may only be read. The 8275 expects to receive a command followed by a sequence of from 0 to 4 parameters, depending on the command. The 8275 instruction set consist of the eight commands shown in Figure 3.4.

To establish the format of the display, the 8275 provides a number of user programmable display format parameters. Display formats having from 1 to 80 characters per row, 1 to 64 rows per screen, and 1 to 16 horizontal lines per row are available.

In addition to transferring characters from memory

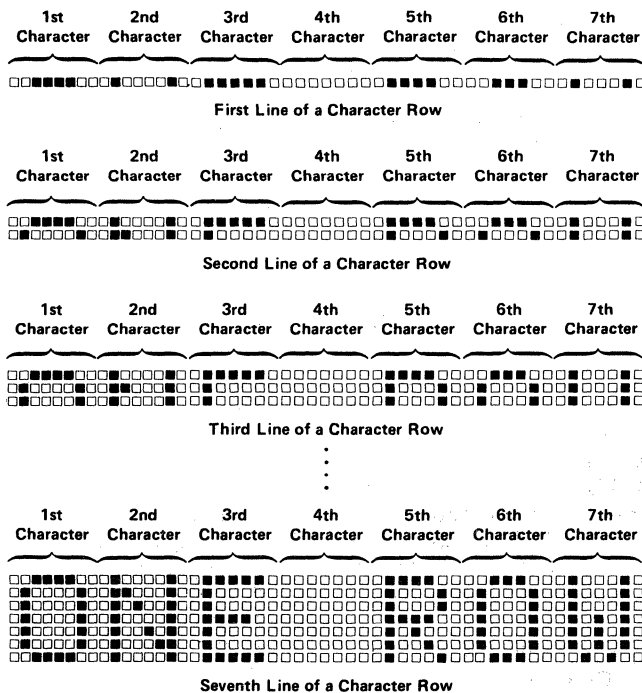


Figure 3-2. 8275 Row Display



# APPLICATIONS

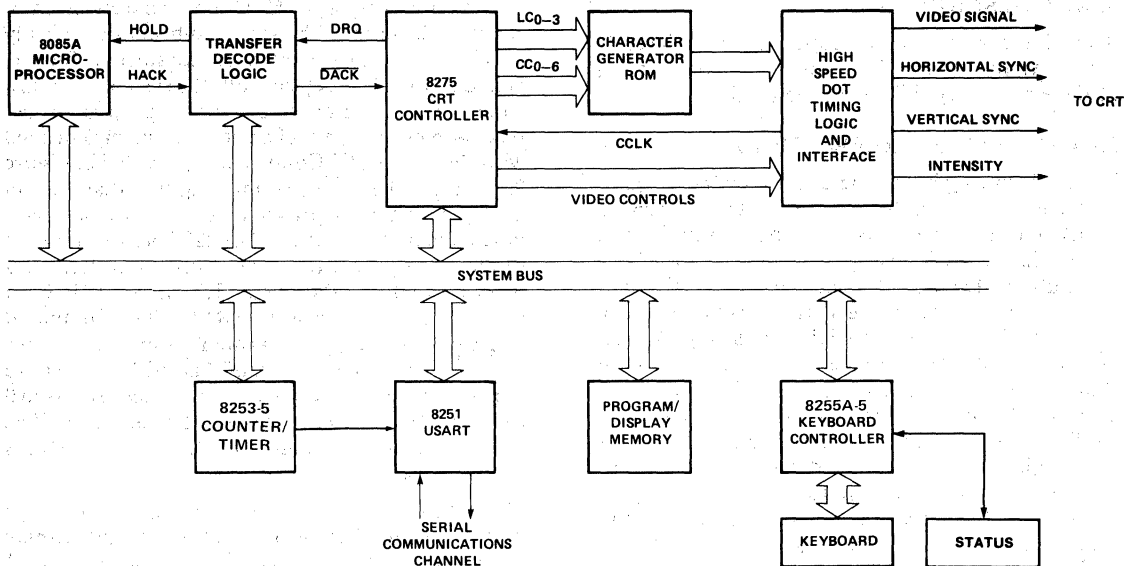


Figure 3-3. CRT System Block Diagram

to the CRT screen, the 8275 features cursor position control. The cursor position may be programmed, via X and Y cursor position registers, to any character position on the display. The user may select from four cursor formats. Blinking or non-blinking underline and reverse video block cursors are available.

## 3.2 CRT TIMING

The 8275 provides two timing outputs, HRTC and VRTC, which are utilized in synchronizing CRT horizontal and vertical oscillators to the 8275 refresh cycle. In addition, whenever HRTC or VRTC is active, a third timing output, VSP (Video Suppress) is true, providing a blinking signal to the dot timing logic. The dot timing logic will normally inhibit the video output to the CRT during the time when video suppress signal is true. An additional timing output, LTEN (Light Enable) is used to provide the ability to force the video output high regardless of the state of VSP. This feature is used by the 8275 to place a cursor on the screen and to control attribute functions. Attributes will be considered in the next section.












The HLGHT (Highlight) output allows an attribute function to increase the CRT beam intensity to a level greater than normal. The fifth timing signal, RVV (Reverse Video) will, when enabled, cause the system video output to be inverted.

COMMAND	NO. OF PARAMETER BYTES	NOTES
RESET	4	Display format parameters required
START DISPLAY	0	DMA operation parameters included in command
STOP DISPLAY	0	---
READ LIGHT PEN	2	---
LOAD CURSOR	2	Cursor X,Y position parameters required
ENABLE INTERRUPT	0	---
DISABLE INTERRUPT	0	---
PRESET COUNTERS	0	Clears all internal counters

Figure 3-4. 8275's Instruction Set

# APPLICATIONS

Character attributes were designed to produce the following graphics:

CHARACTER ATTRIBUTE CODE "CCCC"		OUTPUTS				SYMBOL	DESCRIPTION
		LA <sub>1</sub>	LA <sub>0</sub>	VSP	LTEN		
0000	Above Underline	0	0	1	0		Top Left Corner
	Underline	1	0	0	0		
	Below Underline	0	1	0	0		
0001	Above Underline	0	0	1	0		Top Right Corner
	Underline	1	1	0	0		
	Below Underline	0	1	0	0		
0010	Above Underline	0	1	0	0		Bottom Left Corner
	Underline	1	0	0	0		
	Below Underline	0	0	1	0		
0011	Above Underline	0	1	0	0		Bottom Right Corner
	Underline	1	1	0	0		
	Below Underline	0	0	1	0		
0100	Above Underline	0	0	1	0		Top Intersect
	Underline	0	0	0	1		
	Below Underline	0	1	0	0		
0101	Above Underline	0	1	0	0		Right Intersect
	Underline	1	1	0	0		
	Below Underline	0	1	0	0		
0110	Above Underline	0	1	0	0		Left Intersect
	Underline	1	0	0	0		
	Below Underline	0	1	0	0		
0111	Above Underline	0	1	0	0		Bottom Intersect
	Underline	0	0	0	1		
	Below Underline	0	0	1	0		
1000	Above Underline	0	0	1	0		Horizontal Line
	Underline	0	0	0	1		
	Below Underline	0	0	1	0		
1001	Above Underline	0	1	0	0		Vertical Line
	Underline	0	1	0	0		
	Below Underline	0	1	0	0		
1010	Above Underline	0	1	0	0		Crossed Lines
	Underline	0	0	0	1		
	Below Underline	0	1	0	0		
1011	Above Underline	0	0	0	0		Not Recommended *
	Underline	0	0	0	0		
	Below Underline	0	0	0	0		
1100	Above Underline	0	0	1	0		Special Codes
	Underline	0	0	1	0		
	Below Underline	0	0	1	0		
1101	Above Underline						Illegal
	Underline		Undefined				
	Below Underline						
1110	Above Underline						Illegal
	Underline		Undefined				
	Below Underline						
1111	Above Underline						Illegal
	Underline		Undefined				
	Below Underline						

\*Character Attribute Code 1011 is not recommended for normal operation. Since none of the attribute outputs are active, the character Generator will not be disabled, and an indeterminate character will be generated.

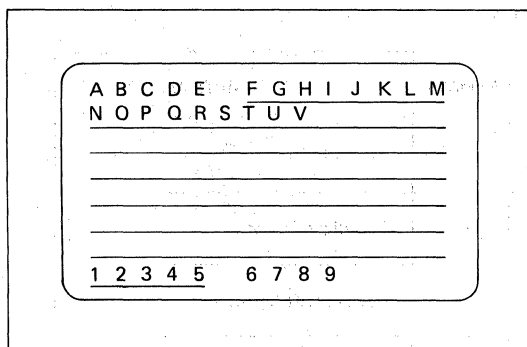
Character Attribute Codes 1101, 1110, and 1111 are illegal.

Blinking is active when B = 1.

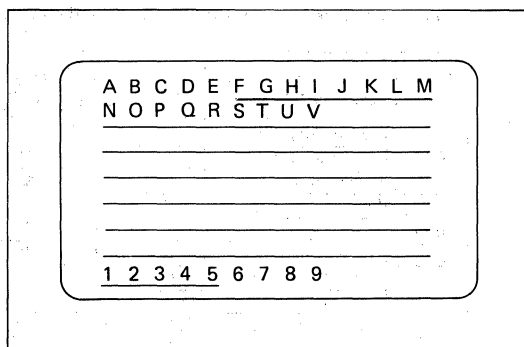
Highlight is active when H = 1.

**Figure 3-5. Character Attributes**

## APPLICATIONS



EXAMPLE OF THE VISIBLE FIELD ATTRIBUTE MODE  
(UNDERLINE ATTRIBUTE)



EXAMPLE OF THE INVISIBLE FIELD ATTRIBUTE MODE  
(UNDERLINE ATTRIBUTE)

Figure 3-6. Field Attribute Examples

### 3.3 SPECIAL FUNCTIONS

**VISUAL ATTRIBUTES**—Visual attributes are special codes which, when retrieved from display memory by the 8275, affect the visual characteristics of a character position or field of characters. Two types of visual attributes exist, character attributes and field attributes.

**Character Attribute Codes:** Character attribute codes can be used to generate graphics symbols without the use of a character generator. This is accomplished by selectively activating the Line Attribute outputs (LAO-LA1), the Video Suppression output (VSP), and the Light Enable output (LTEN). The dot timing logic uses these signals to generate the proper symbols. Character attributes can be programmed to blink or be highlighted individually. Blinking is accomplished with the Video Suppression output (VSP). Blink frequency is equal to the screen refresh frequency divided by 32. Highlighting is accomplished by activating the Highlight output (HGLT). Character attributes were designed to produce the graphic symbols shown in Figure 3.5.

**Field Attribute Codes:** The field attributes are control codes which affect the visual characteristics for a field of characters, starting at the character following the field attribute code up to, and including, the character which precedes the next field attribute code, or up to the end of the frame.

There are six field attributes:

1. *Blink* — Characters following the code are caused to blink by activating the Video Suppression output (VSP). The blink frequency is equal to the screen refresh frequency divided by 32.

2. *Highlight* — Characters following the code are caused to be highlighted by activating the Highlight output (HGLT).
3. *Reverse Video* — Characters following the code are caused to appear in reverse video format by activating the Reverse Video output (RVV).
4. *Underline* — Characters following the code are caused to be underlined by activating the Light Enable output (LTEN).
5. *General Purpose* — There are two additional 8275 outputs which act as general purpose, independently programmable field attributes. These attributes may be used to select colors or perform other desired control functions.

The 8275 can be programmed to provide visible or invisible field attribute characters as shown in Figure 3.6. If the 8275 is programmed in the visible field attribute mode, all field attributes will occupy a position on the screen. They will appear as blanks caused by activation of the Video Suppression output (VSP). The chosen visual attributes are activated after this blanked character. If the 8275 is programmed in the invisible field attribute mode, the 8275 row buffer FIFOs are activated. The FIFOs effectively lengthen the row buffers by 16 characters, making room for up to 16 field attribute characters per display row. The FIFOs are 126 characters by 7 bits in size. When a field attribute is placed in the row buffer during DMA, the buffer input controller recognizes it and places the next character in the proper FIFO. When a field attribute is placed in the buffer output controller during display, it causes the controller to immediately put a character from the FIFO on the Character Code outputs (CCO-6). The chosen attributes are also activated.

**LIGHT PEN DETECTION** — A light pen consists fundamentally of a switch and light sensor. When the light pen is pressed against the CRT screen, the switch enables the light sensor. When the raster sweep coincides with the light sensor position on the display, the light pen output is input and the row and character position coordinates are stored in two 8275 internal registers. These registers can be read by the microprocessor.

**SPECIAL CODES** — Four special codes may be used to help reduce memory, software, or DMA overhead. These codes are placed in character positions in display memory.

1. *End Of Row Code* - Activates VSP. VSP remains active until the end of the line is reached. While VSP is active, the screen is blanked.
2. *End Of Row-Stop DMA Code* - Causes the DMA Control Logic to stop DMA for the rest of the row when it is written into the row buffer. It affects the display in the same way as the End of Row Code.
3. *End Of Screen Code* - Activates VSP. VSP remains active until the end of the frame is reached.
4. *End Of Screen-Stop DMA Code* - Causes the DMA Control Logic to stop DMA for the rest of the frame when it is written into the row buffer. It affects the display in the same way as the End of Screen Code.

**PROGRAMMABLE DMA BURST CONTROL** — The 8275 can be programmed to request single-byte DMA transfers of DMA burst transfers of 2, 4, or 8 characters per burst. The interval between bursts is also programmable. This allows the user to tailor the DMA overhead to fit the system needs.

## 4. DESIGN BACKGROUND

### 4.1 DESIGN PHILOSOPHY

Since the cost of any CRT system is somewhat proportional to parts count, arriving at a minimum part count solution without sacrificing performance has been the motivating force throughout this design effort. To successfully design a CRT terminal and keep the parts count to a minimum, a few things became immediately apparent.

1. An 8085 should be used.
2. Address and data buffering should be eliminated.
3. Multi-port memory should be eliminated.
4. DMA should be eliminated.

Decision 1 is obvious, the 8085's on-board clock generator, bus controller and vectored interrupts greatly reduce the overall part count considerably.

Decision 2 is fairly obvious; if a circuit can be designed so that loading on the data and address lines is kept to a minimum, both the data and address buffers can be eliminated. This easily saves three to eight packages and reduces the power consumption of the design. Both decisions 3 and 4 require a basic understanding of current CRT design concepts.

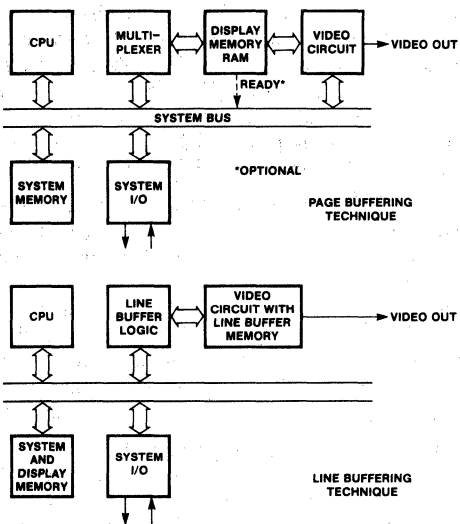
In any CRT design, extreme time conflicts are created because all essential elements require access to the bus. The CPU needs to access the memory to control the system and to handle the incoming characters, but, at the same time, the CRT controller needs to access the memory to keep the raster scan display refreshed. To resolve this conflict two common techniques are employed, page buffering and line buffering.

In the page buffering approach the entire screen memory is isolated from the rest of the system. This isolation is usually accomplished with three-state buffers or two line to one line multiplexers. Of course, whenever a character needs to be manipulated the CPU must gain access to the buffered memory and, again, possible contention between the CPU and the CRT controller results. This contention is usually resolved in one of two ways, (1) the CPU is always given priority, or; (2) the CPU is allowed to access the buffered memory only during horizontal and vertical retrace times.

Approach 1 is the easiest to implement from a hardware point of view, but if the CPU always has priority the display may temporarily blink or "flicker" while the CPU accesses the display memory. This, of course, occurs because when the CPU accesses the display memory the CRT controller is not able to retrieve a character, so the display must be blanked during this time. Aesthetically, this "flickering" is not desirable, so approach 2 is often used.

The second approach eliminates the display flickering encountered in the previously mentioned technique, but additional hardware is required. Usually the vertical and horizontal blank signals are gated with the buffered memory select lines and this line is used to control the CPU's ready line. So, if the CPU wants to use the buffered memory, its ready line is asserted until horizontal or vertical retrace times. This, of course, will impact the CPU's overall throughput.

Both page buffered approaches require a significant amount of additional hardware and for the most part are not well suited for a minimum parts count type of terminal. This guides us to the line buffered approach. This approach eliminates the separate buffered memory for the display, but, at the same time, introduces a few new problems that must be solved.



**Figure 4-1. Line Buffering Technique**

In the line buffered approach both the CPU and the CRT controller share the same memory. Every time the CRT controller needs a new character or line of data, normal processing activity is halted and the CRT controller accesses memory and displays the data. Just how the CRT controller needs to acquire the display data greatly affects the performance of the overall system. Whether the CRT controller needs to gain access to the main memory to acquire a single character or a complete line of data depends on the presence or absence of a separate line or row buffer.

If no row buffer is present the CRT controller must go to the main memory to fetch every character. This of course, is not a very efficient approach because the processor will be forced to relinquish the bus 70% to 80% of the time. So much processor inactivity greatly affects the overall system performance. In fact terminals that use this approach are typically limited to around 1200 to 2400 baud on their serial communication channels. This low baud rate is in general not acceptable, hence this approach was not chosen.

If a separate row buffer is employed the CRT controller only has to access the memory once for each displayed character per line. This forces the processor to relinquish the bus only about 20% to 35% of the time and a full 4800 to 9600 baud can be achieved. Figure 4.1 illustrates these different techniques.

The 8275 CRT controller is ideal for implementing the row buffer approach because the row buffer is contained on the device itself. In fact, the 8275 contains two 80-byte row buffers. The presence of two row buffers allow one buffer to be filled while the other buffer is displaying the data. This dual row buffer approach enhances CPU performance even further.

## 4.2 USING THE 8275 WITHOUT DMA

Until now the process of filling the row buffer has only been alluded to. In reality, a DMA technique is usually used. This approach was demonstrated in AP-32 where an 8257 DMA controller was mated to an 8275 CRT controller. In order to minimize component count, this design eliminates the DMA controller and its associated circuitry while replacing them with a special interrupt-driven transfer.

The only real concern with using the 8275 in an interrupt-driven transfer mode is speed. Eighty characters must be loaded into the 8275 every 617 microseconds and the processor must also have time to perform all the other tasks that are required. To minimize the overhead associated with loading the characters into the 8275 a special technique was employed. This technique involves setting a special

CLOCK CYCLES	SEQ	SOURCE STATEMENT
10	1	PUSH PSH ;SAVE R AND FLAGS
10	2	PUSH H ;SAVE H AND L
10	3	PUSH D ;SAVE D AND E
10	4	LXI H,0000H ;ZERO H AND L
10	5	DAD SP ;PUT STACK POINTER IN H AND L
4	6	YEND ;PUT STACK IN D AND E
-16	7	LHLD CURADR ;GET POINTER
6	8	SPHL ;PUT CURRENT LINE INTO SP
4	9	MVI A,000H ;SET MASK FOR SH
4	10	SHL ;SET SPECIAL TRANSFER BIT
400	11	POP H ;DO 40 MOPS
4	12	RRC ;SET UP A
4	13	SHL ;GO BACK TO NORMAL MODE
10	14	LXI H,0000H ;ZERO HL
10	15	DAD SP ;ADD STACK
4	16	XCHG ;PUT STACK IN H AND L
6	17	SPHL ;RESUME STACK
10	18	LXI H,LAET ;PUT BOTTOM DISPLAY IN H AND L
4	19	XCHG ;SWAP REGISTERS
4	20	MVI A,D ;PUT HIGH ORDER IN A
4	21	CMP H ;SEE IF SAME AS H
7/10	22	JNZ KPTK ;IF NOT LEAVE
4	23	MVI A,E ;PUT LOW ORDER IN A
4	24	CMP L ;SEE IF SAME AS L
7/10	25	JNZ KPTK ;IF NOT LEAVE
10	26	LXI H,TPDIS ;LOAD H AND L WITH TOP OF SCREEN MEMORY
16	27	KPTK: SHLD CURADR ;PUT BACK CURRENT ADDRESS
7	28	MVI A,10H ;GET MASK BYTE
4	29	SIM H ;SET INTERRUPT MASK
10	30	POP D ;GET D AND E
10	31	POP H ;GET H AND L
10	32	POP PSH ;GET A AND FLAGS
4	33	EI ;ENABLE INTERRUPTS
10	34	RET ;GO BACK

TOTAL CLOCK CYCLES = 650 (WORST CASE)

WITH A 6.144 MHZ CRYSTAL TOTAL TIME TO FILL

ROW BUFFER ON 8275 = 650 \* .325 = 211.25 MICROSECONDS

**Figure 4-2. Routine To Load 8275's Row Buffers**

transfer bit and executing a string of POP instructions. The string of POP instructions is used to rapidly move the data from the memory into the 8275. Figure 4.2 shows the basic software structure.

In this design the 8085's SOD line was used as the special transfer bit. In order to perform the transfer properly this special bit must do two things: (1) turn processor reads into  $\overline{\text{DACK}}$  plus  $\overline{\text{WR}}$  for the 8275 and (2) mask processor fetch cycles from the 8275, so that a fetch cycle does not write into the 8275. Conventional logic could have been used to implement this special function, but in this design a small bipolar programmable read only memory was used. Figure 4.3 shows a basic version of the hardware.

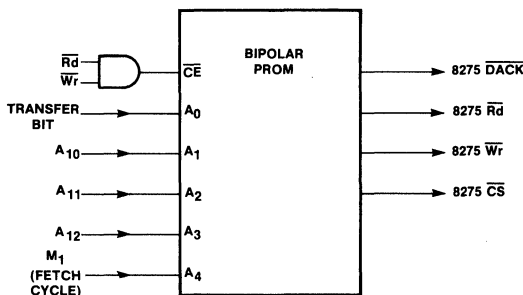


Figure 4-3. Simplified Version of Hardware Decoder

At first, it may seem strange that we are supplying a  $\overline{\text{DACK}}$  when no DMA controller exist in the system. But the reader should be aware that all Intel peripheral devices that have DMA lines actually use  $\overline{\text{DACK}}$  as a chip select for the data. So, when you want to write a command or read status you assert  $\overline{\text{CS}}$  and  $\overline{\text{WR}}$  or  $\overline{\text{RD}}$ , but when you want to read or write data you assert  $\overline{\text{DACK}}$  and  $\overline{\text{RD}}$  or  $\overline{\text{WR}}$ . The peripheral device doesn't "know" if a DMA controller is in the circuit or not. In passing, it should be mentioned that  $\overline{\text{DACK}}$  and  $\overline{\text{CS}}$  should not be asserted on the same device at the same time, since this combination yields an undefined result.

This POP technique actually compares quite favorably in terms of time to the DMA technique. One POP instruction transfers two bytes of data to the 8275 and takes 10 CPU clock cycles to execute, for a net transfer rate of one byte every five clock cycles. The DMA controller takes four clock cycles to transfer one byte but, some time is lost in synchronization. So the difference between the two techniques is one clock cycle per byte maximum. If we compare the overall speed of the 8085 to the

speed of the 8080 used in AP-32, we find that at 3 MHz we can transfer one byte every 1.67 microseconds using the 8085 and POP technique vs. 2 microseconds per byte for the 2 MHz 8080 using DMA.

## 5. CIRCUIT DESCRIPTION

### 5.1 SCOPE OF THE PROJECT

A fully functional, microprocessor-based CRT terminal was designed and constructed using the 8275 CRT controller and the 8085 as the controlling element. The terminal had many of the functions found in existing commercial low-cost terminals and more sophisticated features could easily be added with a modest amount of additional software. In order to minimize component count LSI devices were used whenever possible and software was used to replace hardware.

### 5.2 SYSTEM TARGET SPECIFICATIONS

The design specifications for the CRT terminal were as follows:

#### Display Format

- 80 characters per display row
- 25 display rows

#### Character Format

- 5 X 7 dot matrix character contained within a 7 X 10 matrix
- First and seventh columns blanked
- Ninth line cursor position
- Blinking underline cursor

#### Special Characters Recognized

- Control characters
- Line feed
- Carriage Return
- Backspace
- Form feed

#### Escape Sequences Recognized

- ESC, A, Cursor up
- ESC, B, Cursor down
- ESC, C, Cursor right
- ESC, D, Cursor left
- ESC, E, Clear screen
- ESC, H, Home cursor
- ESC, J, Erase to the end of the screen
- ESC, K, Erase the current line

#### Characters Displayed

- 96 ASCII alphanumeric characters
- Special control characters

# APPLICATIONS

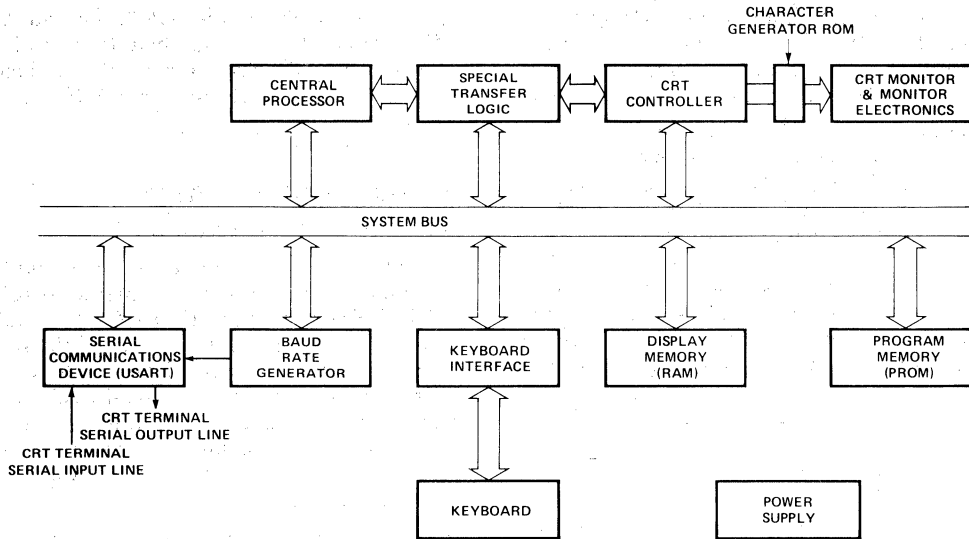


Figure 5-1. CRT Terminal Block Diagram

### Characters Transmitted

- 96 ASCII alphanumeric characters
- ASCII control characters

### Program Memory

- 2K bytes of 2716 EPROM

### Display/ Buffer/ Stack Memory

- 2K bytes 2114 static memory (4 packages)

### Data Rate

- 9600 BAUD using 3MHz 8085

### CRT Monitor

- Ball Bros TV-12, 12MHz B.W.

### Keyboard

- Any standard un-encoded ASCII keyboard

### Screen Refresh Rate

- 60 Hz

Worst case bus loading:

Data Bus:	8275	20pf
	8255A-5	20pf
	8253-5	20pf
	8253-5	20pf
	8251A	20pf
	2x 2114	10pf
	2716	12pf
	8212	12pf
		<hr/>
		114pf max

Only A<sub>8</sub> - A<sub>15</sub> are important since A<sub>0</sub> - A<sub>7</sub> are latched by the 8212

Address Bus:	4x 2114	20pf
	2716	6pf
		<hr/>
		26pf max

This loading assures that all components will be compatible with a 3MHz 8085 and that no wait states will be required

Figure 5-2. Bus Loading

## 5.3 HARDWARE DIScription

A block diagram of the CRT terminal is shown in Figure 5.1. The diagram shows only the essential system features. A detailed schematic of the CRT is contained in the Appendix. The terminal was constructed on a simple 6" by 6" wire wrap board. Because of the minimum bus loading no buffering of any kind was needed (see Figure 5.2).

The "heart" of the CRT terminal is the 8085 microprocessor. The 8085 initializes all devices in the system, loads the CRT controller, scans the keyboard, assembles the characters to be trans-

mitted, decodes the incoming characters and determines where the character is to be placed on the screen. Clearly, the processor is quite busy.

A standard list of LSI peripheral devices surround the 8085. The 8251A is used as the serial communication link, the 8255A-5 is used to scan the keyboard and read the system variables through a set of

# APPLICATIONS

switches, and the 8253 is used as a baud rate generator and as a "horizontal pulse extender" for the 8275.

The 8275 is used as the CRT controller in the system, and a 2716 is used as the character generator. To handle the high speed portion of the terminal the 8275 is surrounded by a small handful of TTL. The program memory is contained in one 2716 EPROM and the data and screen memory use four 2114-type RAMs.

All devices in this system are memory mapped. A bipolar PROM is used to decode all of the addresses for the RAM, ROM, 8275, and 8253. As mentioned earlier, the bipolar prom also turns READs into DACK's and WR's for the 8275. The 8255 and 8253 are decoded by a simple address line chip select method. The total package count for the system is 20, not including the serial line drivers. If this same terminal were designed using the MCS-85 family of integrated circuits, additional part savings could have been realized. The four 2114's could have been replaced by two 8185's and the 8255 and the 2716 program PROM could have been replaced by one 8755. Additionally, since both the 8185 and the 2716 have address latches no 8212 would be needed, so the total parts count could be reduced by three or four packages.

## 5.4 SYSTEM OPERATION

The 8085 CPU initializes each peripheral to the appropriate mode of operation following system reset. After initialization, the 8085 continually polls the 8251A to see if a character has been sent to the terminal. When a character has been received, the 8085 decodes the character and takes appropriate action. While the 8085 is executing the above "foreground" programs, it is being interrupted once every 617 microseconds by the 8275. This "background" program is used to load the row buffers on the 8275. The 8085 is also interrupted once every frame time, or 16.67 ms, to read the keyboard and the status of the 8275.

As discussed earlier, a special POP technique was used to rapidly move the contents of the display RAM into the 8275's row buffers. The characters are then synchronously transferred to the character code outputs CC0-CC5, connected to the character generator address lines A3-A9 (Figure 5.3). Line count outputs LC0-LC2 from the 8275 are applied to the character generator address lines A0-A2. The 8275 displays character rows one line at a time. The line count outputs are used to determine which line of the character selected by A3-A8 will be displayed. Following the transfer of the first line to the dot timing logic, the line count is incremented and the second line of the character row is selected. This

process continues until the last line of the row is transferred to the dot timing logic.

The dot timing logic latches the output of the character generator ROM into a parallel in, serial out synchronous shift register. This shift register is clocked at the dot clock rate (11.34 MHz) and its output constitutes the video input to the CRT.

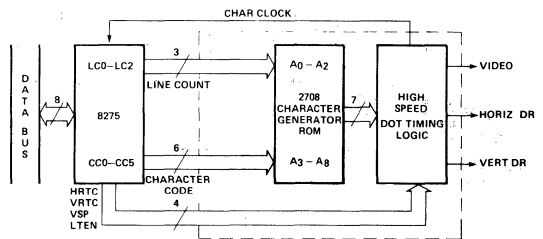


Figure 5-3 Character Generator/Dot Timing Logic Block Diagram

Table 5-1

PARAMETER	RANGE
Vertical Blanking Time (VRTC)	900 $\mu$ sec nominal
Vertical Drive Pulsewidth	$300 \mu\text{sec} \leq \text{PW} \leq 1.4 \text{ ms}$
Horizontal Blanking Time (HRTC)	11 $\mu$ sec nominal
Horizontal Drive Pulsewidth	$25 \mu\text{sec} \leq \text{PW} \leq 30 \mu\text{sec}$
Horizontal Repetition Rate	$15,750 \pm 500 \text{ pps}$

## 5.5 SYSTEM TIMING

Before any specific timing can be calculated it is necessary to determine what constraints the chosen CRT places on the overall timing. The requirements for the Ball Bros. TV-12 monitor are shown in Table 5.1. The data from Table 5.1, the 8275 specifications, and the system target specifications are all that is needed to calculate the system's timing.

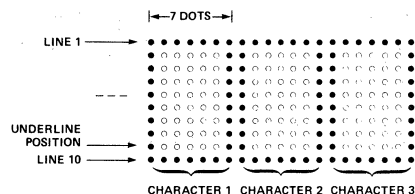


Figure 5-4. Row Format



# APPLICATIONS

First, let's select and "match" a few numbers. From our target specifications, we see that each character is displayed on a 7 X 10 field, and is formed by a 5 X 7 dot matrix (Figure 5.4). The 8275 allows the vertical retrace time to be only an integer multiple of

the horizontal character line. This means that the total number of horizontal lines in a frame equals 10 times the number of character lines plus the vertical retrace time, which is programmed to be either 1, 2, 3, or 4 character lines. Twenty-five display lines

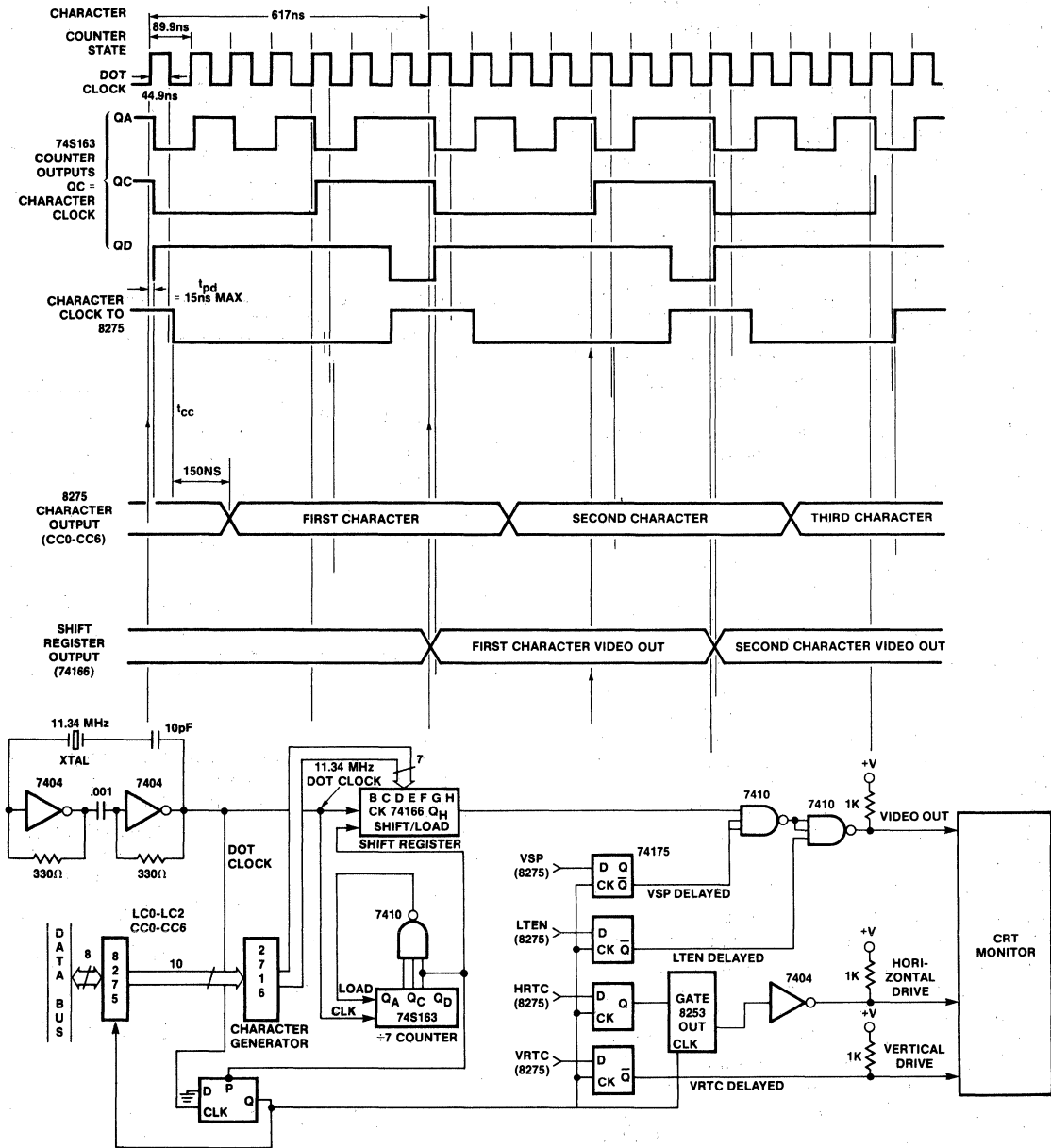


Figure 5-5. Dot Timing Logic

## APPLICATIONS

require 250 horizontal lines. So, if we wish to have a horizontal frequency in the neighborhood of 15,750 Hz we must choose either one or two character lines for vertical retrace. To allow for a little more margin at the top and bottom of the screen, two character lines were chosen for vertical retrace. This choice yields a net  $250 + 20 = 270$  horizontal lines per frame. So, assuming a 60 Hz frame:

$$60 \text{ Hz} * 270 = 16,200 \text{ Hz (horizontal frequency)}$$

This value falls within our target specification of 15,750 Hz with a 500 Hz variation and also assures timing compatibility with the Ball monitor since, 20 horizontal sync times yield a vertical retrace time of:

$$61.7 \text{ microseconds} * 20 \text{ horizontal sync times} = 1.2345 \text{ milliseconds}$$

This number meets the nominal VRTC and vertical drive pulse width time for the Ball monitor. A horizontal frequency of 16,200 Hz implies a  $1/16,200 = 61.73$  microsecond period.

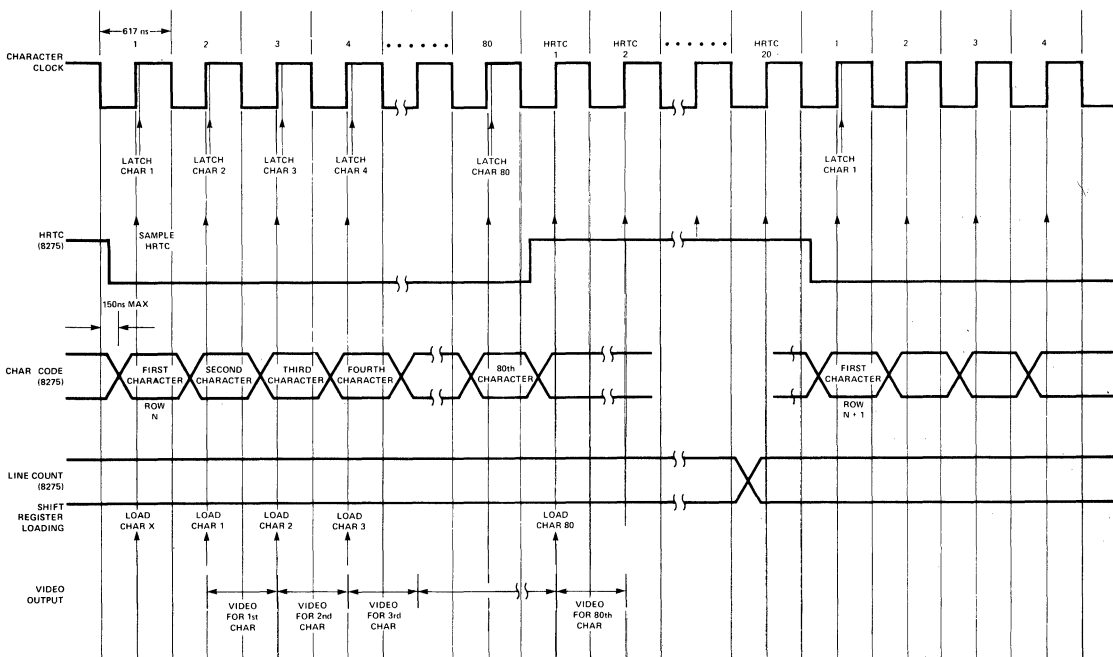
It is now known that the terminal is using 250 horizontal lines to display data and 20 horizontal lines to allow for vertical retrace and that the horizontal frequency is 16,200 Hz. The next thing that needs to be determined is how much time must

be allowed for horizontal retrace. Unfortunately, this number depends almost entirely on the monitor used. Usually, this number lies somewhere between 15 and 30 percent of the total horizontal line time, which in this case is  $1/16,200 \text{ Hz}$  or 61.73 microseconds. Since in most designs a fixed number of characters can be displayed on a horizontal line, it is often useful to express retrace as a given number of character times. In this design, 80 characters can be displayed on a horizontal line, it is often useful to express retrace as a given number of character times. In this design, 80 characters can be displayed on a horizontal line and it was empirically found that allowing 20 horizontal character times for retrace gave the best results. So, in reality, there are 100 character times in every given horizontal line, 80 are used to display characters and 20 are used to allow for retrace. It should be noted that if too many character times are used for retrace, less time will be left to display the characters and the display will not "fill out" the screen. Conversely, if not enough character times are allowed for retrace, the display may "run off" the screen.

One hundred character times per complete horizontal line means that each character requires

$$61.73 \text{ microseconds} / 100 \text{ character times} = 617.3 \text{ nanoseconds.}$$

If we multiply the 20 horizontal retrace times by the



**Figure 5-6. CRT System Timing**

617.3 nanoseconds needed for each character, we find

$$617.3 \text{ nanoseconds} * 20 \text{ retrace times} = 12.345 \text{ microseconds}$$

This value falls short of the 25 to 30 microseconds required by the horizontal drive of the Ball monitor. To correct for this, an 8253 was programmed in the one-shot mode and was used to extend the horizontal drive pulsewidth.

Now that the 617.3 nanosecond character clock period is known, the dot clock is easy to calculate. Since each character is formed by placing 7 dots along the horizontal.

$$\begin{aligned} \text{DOT CLOCK PERIOD} &= 617.3 \text{ ns} \\ &(\text{CHARACTER CLK PERIOD}) / 7 \text{ DOTS} \\ \text{DOT CLOCK PERIOD} &= 88.183 \text{ nanoseconds} \\ \text{DOT CLOCK FREQUENCY} &= 1 / \text{PERIOD} = 11.34 \text{ MHz} \end{aligned}$$

Figures 5.5 and 5.6 illustrate the basic dot timing and the CRT system timing, respectively.

## 6. SYSTEM SOFTWARE

### 6.1 SOFTWARE OVERVIEW

As mentioned earlier the software is structured on a "foreground-background" basis. Two interrupt-driven routines, FRAME and POPDAT (Fig. 6.1) request service every 16.67 milliseconds and 617 microseconds respectively, frame is used to check the baud rate switches, update the system pointers and decode and assemble the keyboard characters. POPDAT is used to move data from the memory into the 8275's row buffer rapidly.

The foreground routine first examines the line-local switch to see whether to accept data from the USART or the keyboard. If the terminal is in the local mode, action will be taken on any data that is entered through the keyboard and the USART will be ignored on both output and input. If the terminal is in the line mode data entered through the keyboard will be transmitted by the USART and action will be taken on any data read out of the USART.

When data has been entered in the terminal the software first determines if the character received was an escape, line feed, form feed, carriage return, back space, or simply a printable character. If an escape was received the terminal assumes the next received character will be a recognizable escape sequence character. If it isn't no operation is performed.

After the character is decoded, the processor jumps to the routine to perform the required task. Figure 6.2 is a flow chart of the basic software operations; the program is listed in Appendix 6.8.

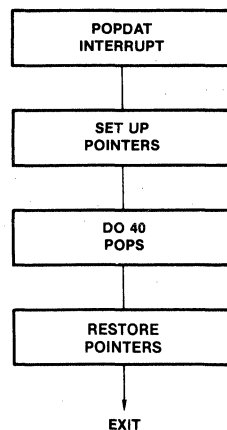
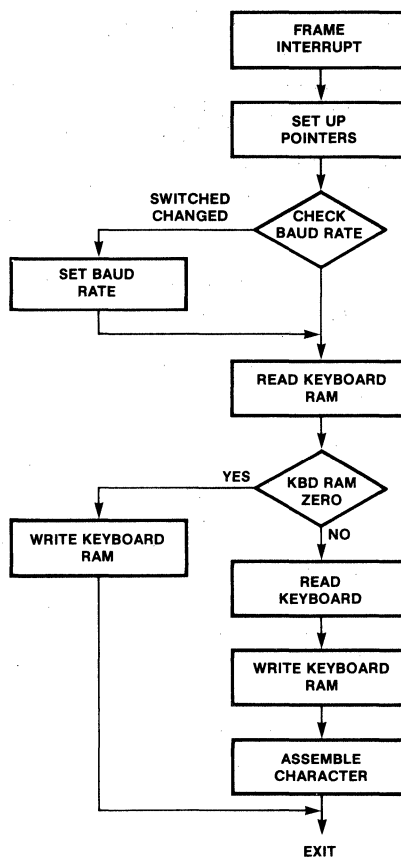
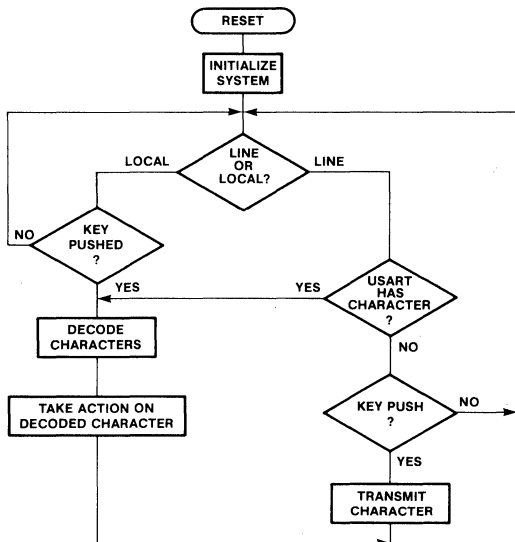


Figure 6-1. Frame and Popdat Interrupt Routines



**Figure 6-2. Basic Terminal Software**

	1st Column	2nd Column	.....	80th Column
ROW 1	0800H	0801H	.....	084FH
ROW 2	0850H	0851H	.....	089FH
ROW 3	08A0H	08A1H	.....	08EFH
ROW 4	08F0H	08F1H	.....	093FH
ROW 5	0940H	0941H	.....	098FH
ROW 6	0990H	0991H	.....	090FH
ROW 7	09E0H	09E1H	.....	0A2FH
ROW 8	0A30H	0A31H	.....	0A7FH
ROW 9	0A80H	0A81H	.....	0ACFH
ROW 10	0AD0H	0AD1H	.....	0B1FH
ROW 11	0B20H	0B21H	.....	0B6FH
ROW 12	0B70H	0B71H	.....	0BBFH
ROW 13	0BC0H	0BC1H	.....	0C0FH
ROW 14	0C10H	0C11H	.....	0C5FH
ROW 15	0C60H	0C61H	.....	0CAFH
ROW 16	0CB0H	0CB1H	.....	0CFFH
ROW 17	0D00H	0D01H	.....	0D4FH
ROW 18	0D50H	0D51H	.....	0D9FH
ROW 19	0DA0H	0DA1H	.....	0DEFH
ROW 20	0DF0H	0DF1H	.....	0E3FH
ROW 21	0E40H	0E41H	.....	0E8FH
ROW 22	0E90H	0E91H	.....	0EDFH
ROW 23	0EE0H	0EE1H	.....	0F2FH
ROW 24	0F30H	0F31H	.....	0F7FH
ROW 25	0F80H	0F81H	.....	0FCFH

**Figure 6-3. Screen Display After Initialization**

## 6.2 SYSTEM MEMORY ORGANIZATION

The display memory organization is shown in Figure 6.3. The display begins at location 0800H in memory and ends at location 0FCFH. The 48 bytes of RAM from location 0FD0H to 0FFFH are used as system stack and temporary system storage. 2K bytes of PROM located at 0000H through 07FFH contain the systems program.

## 6.3 MEMORY POINTERS AND SCROLLING

To calculate the location of a character on the screen, three variables must be defined. Two of these variables are the X and Y position of the cursor (CURSX, CURSY). In addition, the memory address defining the top line of the display must be known, since scrolling on the 8275 is accomplished simply by changing the pointer that loads the 8275's row buffers from memory. So, if it is desired to scroll the display up or down all that must be changed is one 16-bit memory pointer. This pointer is entered into the system by the variable TOPAD (TOP Address) and always defines the top line of the display. Figure 6.4 details screen operation during scrolling.

Subroutines CALCU (Calculate) and ADX (ADd X axis) use these three variables to calculate an absolute memory address. The subroutine CALCU is used whenever a location in the screen memory must be altered.

## 6.4 SOFTWARE TIMING

One important question that must be asked about the terminal software is, "How fast does it run". This is important because if the terminal is running at 9600 baud, it must be able to handle each received character in 1.04 milliseconds. Figure 6.5 is a flowchart of the subroutine execution times. It should be pointed out that all of the times listed are "worst case" execution times. This means that all routines assume they must do the maximum amount of data manipulation. For instance, the PUT routine assumes that the character is being placed in the last column and that a line feed must follow the placing of the character on the screen.

How fast do the routines need to execute in order to assure operation at 9600 baud? Since POPDAT interrupts occur every 617 microseconds, it is possible to receive two complete interrupt requests in every character time (1042 microseconds) at 9600

# APPLICATIONS

ROW 1	0800H	0801H	.....084FH
ROW 2	0850H	0851H	.....089FH
ROW 3	08A0H	08A1H	.....08EFH
ROW 4	08F0H	08F1H	.....093FH
ROW 5	0940H	0941H	.....098FH
ROW 6	0990H	0991H	.....090FH
ROW 7	09E0H	09E1H	.....0A2FH
ROW 8	0A30H	0A31H	.....0A7FH
ROW 9	0A80H	0A81H	.....0ACFH
ROW 10	0AD0H	0AD1H	.....0B1FH
ROW 11	0B20H	0B21H	.....0B6FH
ROW 12	0B70H	0B71H	.....0BBFH
ROW 13	0BC0H	0BC1H	.....0C0FH
ROW 14	0C10H	0C11H	.....0C5FH
ROW 15	0C60H	0C61H	.....0CAFH
ROW 16	0CB0H	0CB1H	.....0CFFH
ROW 17	0D00H	0D01H	.....0D4FH
ROW 18	0D50H	0D51H	.....0D9FH
ROW 19	0DA0H	0DA1H	.....0DEFH
ROW 20	0DF0H	0DF1H	.....0E3FH
ROW 21	0E40H	0E41H	.....0E8FH
ROW 22	0E90H	0E91H	.....0EDFH
ROW 23	0EE0H	0EE1H	.....0F2FH
ROW 24	0F30H	0F31H	.....0F7FH
ROW 25	0F80H	0F81H	.....0FCFH

After Initialization

ROW 2	0850H	0851H	.....089FH
ROW 3	08A0H	08A1H	.....08EFH
ROW 4	08F0H	08F1H	.....093FH
ROW 5	0940H	0941H	.....098FH
ROW 6	0990H	0991H	.....090FH
ROW 7	09E0H	09E1H	.....0A2FH
ROW 8	0A30H	0A31H	.....0A7FH
ROW 9	0A80H	0A81H	.....0ACFH
ROW 10	0AD0H	0AD1H	.....0B1FH
ROW 11	0B20H	0B21H	.....0B6FH
ROW 12	0B70H	0B71H	.....0BBFH
ROW 13	0BC0H	0BC1H	.....0C0FH
ROW 14	0C10H	0C11H	.....0C5FH
ROW 15	0C60H	0C61H	.....0CAFH
ROW 16	0CB0H	0CB1H	.....0CFFH
ROW 17	0D00H	0D01H	.....0D4FH
ROW 18	0D50H	0D51H	.....0D9FH
ROW 19	0DA0H	0DA1H	.....0DEFH
ROW 20	0DF0H	0DF1H	.....0E3FH
ROW 21	0E40H	0E41H	.....0E8FH
ROW 22	0E90H	0E91H	.....0EDFH
ROW 23	0EE0H	0EE1H	.....0F2FH
ROW 24	0F30H	0F31H	.....0F7FH
ROW 25	0F80H	0F81H	.....0FCFH
ROW 1	0800H	0801H	.....084FH

After 1 Scroll

ROW 3	08A0H	08A1H	.....08EFH
ROW 4	08F0H	08F1H	.....093FH
ROW 5	0940H	0941H	.....098FH
ROW 6	0990H	0991H	.....090FH
ROW 7	09E0H	09E1H	.....0A2FH
ROW 8	0A30H	0A31H	.....0A7FH
ROW 9	0A80H	0A81H	.....0ACFH
ROW 10	0AD0H	0AD1H	.....0B1FH
ROW 11	0B20H	0B21H	.....0B6FH
ROW 12	0B70H	0B71H	.....0BBFH
ROW 13	0BC0H	0BC1H	.....0C0FH
ROW 14	0C10H	0C11H	.....0C5FH
ROW 15	0C60H	0C61H	.....0CAFH
ROW 16	0CB0H	0CB1H	.....0CFFH
ROW 17	0D00H	0D01H	.....0D4FH
ROW 18	0D50H	0D51H	.....0D9FH
ROW 19	0DA0H	0DA1H	.....0DEFH
ROW 20	0DF0H	0DF1H	.....0E3FH
ROW 21	0E40H	0E41H	.....0E8FH
ROW 22	0E90H	0E91H	.....0EDFH
ROW 23	0EE0H	0EE1H	.....0F2FH
ROW 24	0F30H	0F31H	.....0F7FH
ROW 25	0F80H	0F81H	.....0FCFH
ROW 1	0800H	0801H	.....084FH
ROW 2	0850H	0851H	.....089FH

After 2 Scrolls

ROW 4	08F0H	08F1H	.....093FH
ROW 5	0940H	0941H	.....098FH
ROW 6	0990H	0991H	.....090FH
ROW 7	09E0H	09E1H	.....0A2FH
ROW 8	0A30H	0A31H	.....0A7FH
ROW 9	0A80H	0A81H	.....0ACFH
ROW 10	0AD0H	0AD1H	.....0B1FH
ROW 11	0B20H	0B21H	.....0B6FH
ROW 12	0B70H	0B71H	.....0BBFH
ROW 13	0BC0H	0BC1H	.....0C0FH
ROW 14	0C10H	0C11H	.....0C5FH
ROW 15	0C60H	0C61H	.....0CAFH
ROW 16	0CB0H	0CB1H	.....0CFFH
ROW 17	0D00H	0D01H	.....0D4FH
ROW 18	0D50H	0D51H	.....0D9FH
ROW 19	0DA0H	0DA1H	.....0DEFH
ROW 20	0DF0H	0DF1H	.....0E3FH
ROW 21	0E40H	0E41H	.....0E8FH
ROW 22	0E90H	0E91H	.....0EDFH
ROW 23	0EE0H	0EE1H	.....0F2FH
ROW 24	0F30H	0F31H	.....0F7FH
ROW 25	0F80H	0F81H	.....0FCFH
ROW 1	0800H	0801H	.....084FH
ROW 2	0850H	0851H	.....089FH
ROW 3	08A0H	08A1H	.....08EFH

After 3 Scrolls

**Figure 6-4. Screen Memory During Scrolling**

# APPLICATIONS

baud. Each POPDAT interrupt executes in 211 microseconds maximum. This means that each routine must execute in:

$$1042 - 2 * 211 = 620 \text{ microseconds}$$

By adding up the times for any loop, it is clear that all routines meet this speed requirement, with the exception of ESC J. This means that if the terminal is operating at 9600 baud, at least one character time must be inserted after an ESC J sequence.

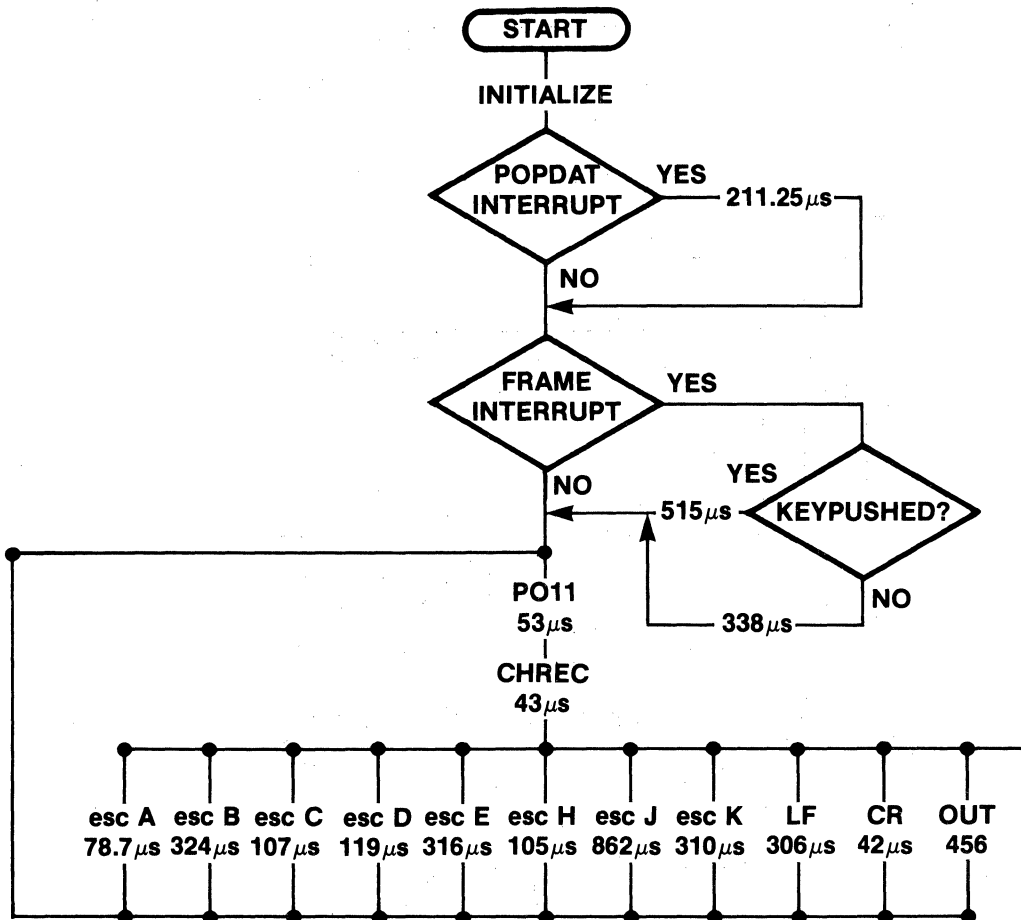
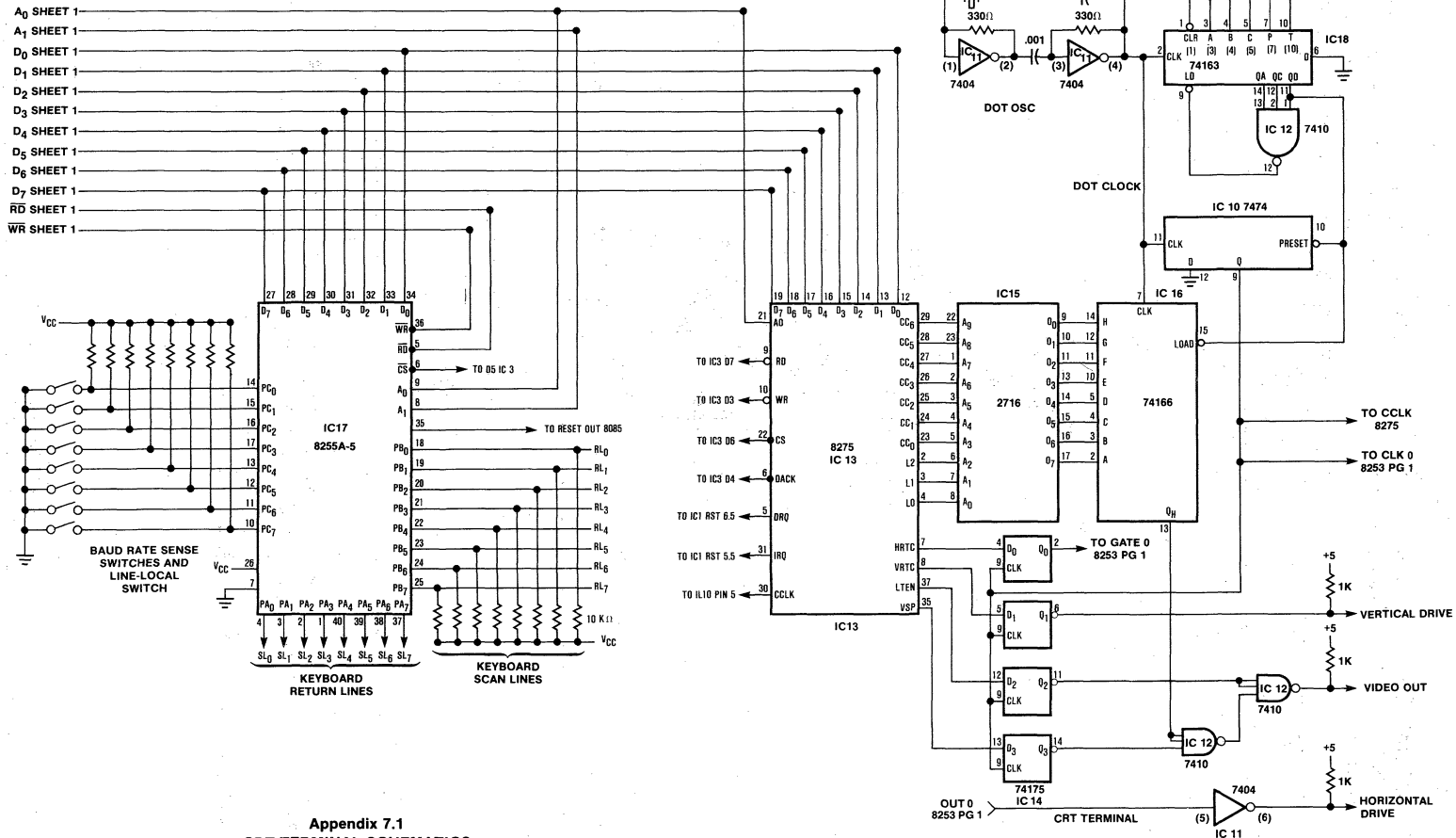


Figure 6-5. Timing Flowchart



Appendix 7.1  
CRT TERMINAL SCHEMATICS





# APPLICATIONS

## Appendix 7.2 KEYBOARD INTERFACE

The keyboard used in this design was a simple unencoded ASCII keyboard. In order to keep the cost to a minimum a simple scan matrix technique was implemented by using two ports of an 8255 parallel I/O device.

When the system is initialized the contents of the eight keyboard RAM locations are set to zero. Once every frame, which is 16.67 milliseconds the contents of the keyboard ram is read and then rewritten with the contents of the current switch matrix. If a non-zero value of one of the keyboard RAM locations is found to be the same as the corresponding current switch matrix, a valid key push is registered and

action is taken. By operating the keyboard scan in this manner an automatic debounce time of 16.67 milliseconds is provided.

Figure 7.2A shows the actual physical layout of the keyboard and Figure 7.2B shows how the individual keys were encoded. On Figure 7.2B the scan lines are the numbers on the bottom of each key position and the return lines are the numbers at the top of each key position. The shift, control, and caps lock key were brought in through separate lines of port C of the 8255. Figure 7.3 shows the basic keyboard matrix.

In order to guarantee that two scan lines could not be shorted together if two or more keys are pushed simultaneously, isolation diodes could be added as shown in Figure 7.4.

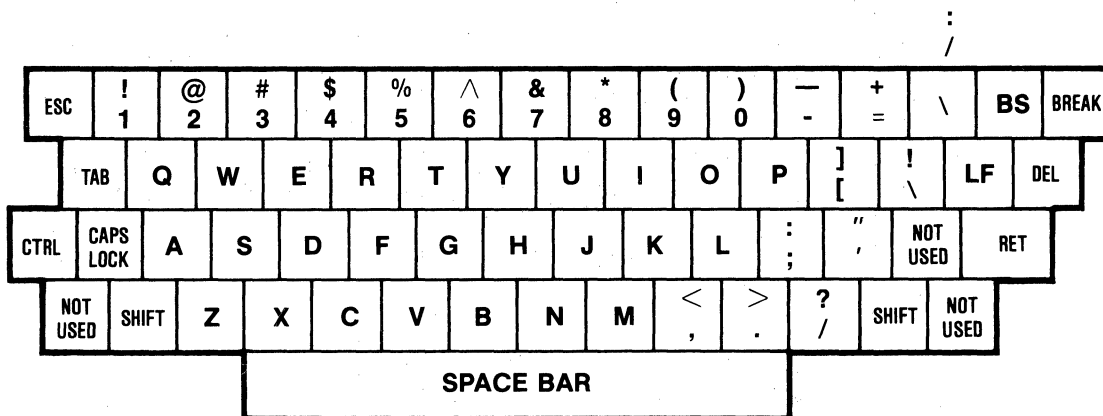
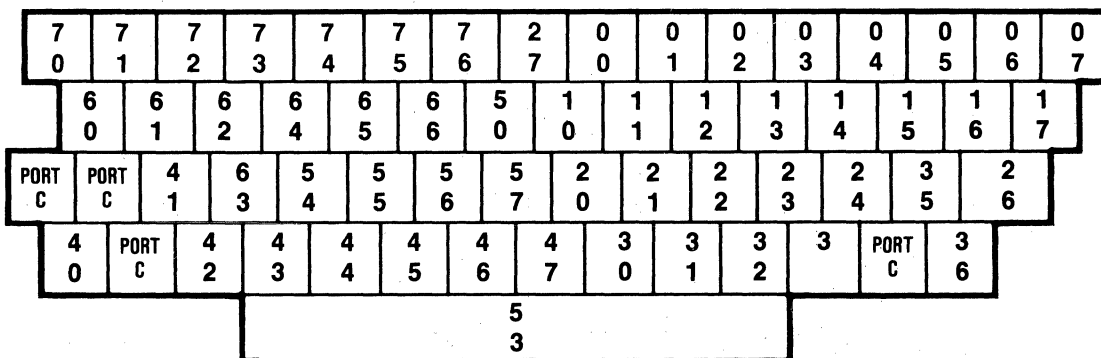


Figure 7-2A. Keyboard Layout



TOP NUMBER = RETURN LINE  
BOTTOM NUMBER = SCAN LINE

Figure 7-2B. Keyboard Encoding

# APPLICATIONS

## Appendix 7.3 ESCAPE/CONTROL/DISPLAY CHARACTER SUMMARY

BIT	CONTROL CHARACTERS			DISPLAYABLE CHARACTER				ESCAPE SEQUENCE						
	0 <sub>0</sub> 0	0 <sub>0</sub> 1	0 <sub>1</sub> 0	0 <sub>1</sub> 1	1 <sub>0</sub> 0	1 <sub>0</sub> 1	1 <sub>1</sub> 0	1 <sub>1</sub> 1	0 <sub>1</sub> 0	0 <sub>1</sub> 1	1 <sub>0</sub> 0	1 <sub>0</sub> 1	1 <sub>1</sub> 0	1 <sub>1</sub> 1
0000	NUL <sup>®</sup>	DLE <sup>P</sup>	SP	φ	@	P		P						
0001	SOH <sup>A</sup>	DC1 <sup>Q</sup>	!	!	A	Q	A	Q			↑ A			
0010	STX <sup>B</sup>	DC2 <sup>R</sup>	"	2	B	R	B	R			↓ B			
0011	ETX <sup>C</sup>	DC3 <sup>S</sup>	#	3	C	S	C	S			→ C			
0100	EOT <sup>D</sup>	DC4 <sup>T</sup>	\$	4	D	T	D	T			← D			
0101	ENQ <sup>E</sup>	NAK <sup>U</sup>	%	5	E	U	E	U			CLR E			
0110	ACK <sup>F</sup>	SYN <sup>V</sup>	&	6	F	V	F	V						
0111	BEL <sup>G</sup>	ETB <sup>W</sup>	'	7	G	W	G	W						
1000	BS <sup>H</sup>	CAN <sup>X</sup>	(	8	H	X	H	X			HOME H			
1001	HT <sup>I</sup>	EM <sup>Y</sup>	)	9	I	Y	I	Y						
1010	LF <sup>J</sup>	SUB <sup>Z</sup>	*	:	J	Z	J	Z			EOS I			
1011	VT <sup>K</sup>	ESC <sup>I</sup>	+	;	K	[	K				EL J			
1100	FF <sup>L</sup>	FS <sup>/</sup>	,	<	L	\	L							
1101	CR <sup>M</sup>	GS	-	=	M	]	M							
1110	SO <sup>N</sup>	RS <sup>^</sup>	.	>	N	^	N							
1111	SI <sup>O</sup>	us <sup>-</sup>	/	?	O	-	O							

NOTE: Shaded blocks = functions terminal will react to. Others can be generated but are ignored up on receipt.

# APPLICATIONS

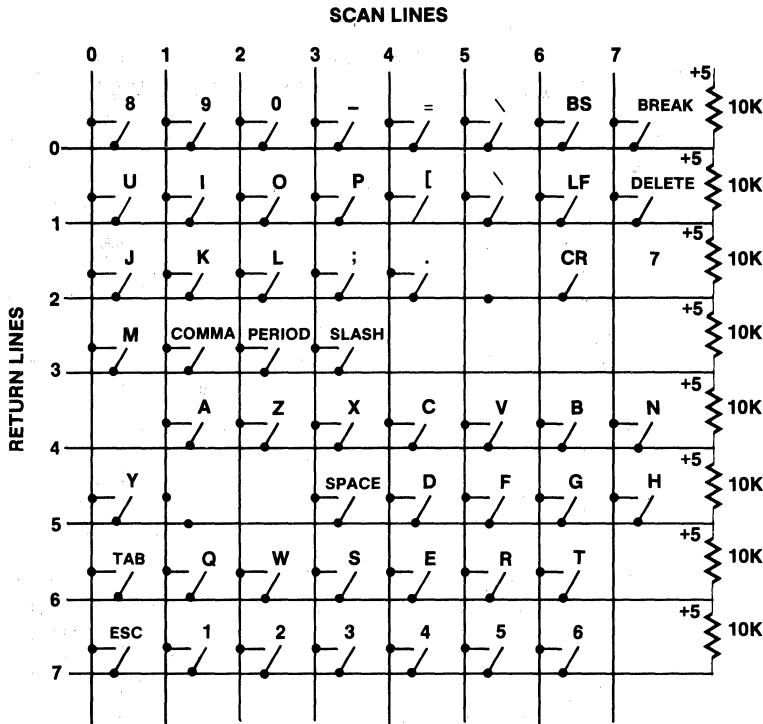


Figure 7-3. Keyboard Matrix

## Appendix 7.4 PROM DECODING

As stated earlier, all of the logic necessary to convert the 8275 into a non-DMA type of device was performed by a single small bipolar prom. Besides turning certain processor READS into DACKS and WRITES for the 8275, this 32 by 8 prom decoded addresses for the system ram, rom, as well as for the 8255 parallel I/O port.

Any bipolar prom that has a by eight configuration could function in this application. This particular device was chosen simply because it is the only "by eight" prom available in a 16 pin package. The connection of the prom is shown in detail in Figure 7.5 and its truth table is shown in Figure 7.6. Note that when a fetch cycle (M1) is not being performed, the state of the SOD line is the only thing that determines if memory reads will be written into the 8275's row buffers. This is done by pulling both DACK and WRITE low on the 8275.

Also note that all of the outputs of the bipolar prom **MUST BE PULLED HIGH** by a resistor. This prevents any unwanted assertions when the prom is disabled.

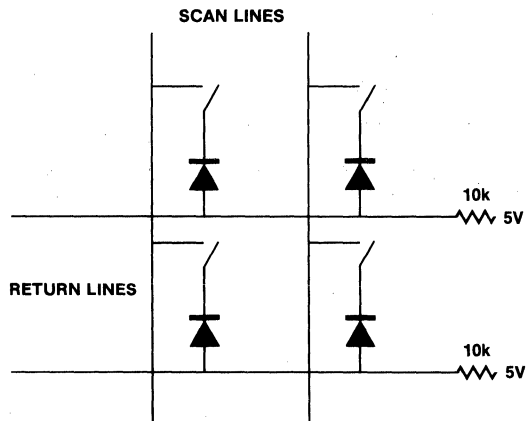


Figure 7-4. Isolating Scan Lines With Diodes

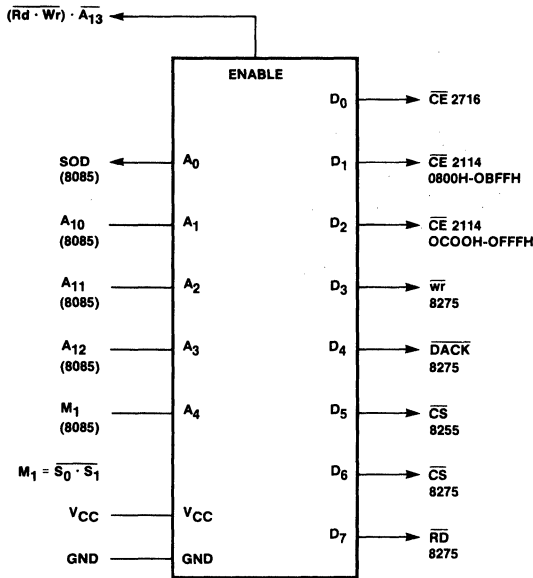


Figure 7-5. Bipolar Prom (825123) Connection

M1	A12	A11	A10	Sod	8275 Rd	8275 CS	8255 CS	8275 DACK	8275 WR	2114 H	2114 L	2716
A4	A3	A2	A1	A0	D7	D6	D5	D4	D3	D2	D1	D0
0	0	0	0	0	1	1	1	1	1	1	1	0
0	0	0	0	1	1	1	1	1	1	1	1	0
0	0	0	1	0	1	1	1	1	1	1	1	0
0	0	0	1	1	1	1	1	1	1	1	1	0
0	0	1	0	0	1	1	1	1	1	1	0	1
0	0	1	0	1	1	1	1	1	1	1	0	1
0	0	1	1	0	1	1	1	1	1	0	1	1
0	0	1	1	1	1	1	1	1	1	0	1	1
0	1	0	0	0	1	0	1	1	0	1	1	1
0	1	0	0	1	1	0	1	1	0	1	1	1
0	1	0	1	0	0	0	1	1	1	1	1	1
0	1	0	1	1	0	0	1	1	1	1	1	1
0	1	1	0	0	1	1	0	1	1	1	1	1
0	1	1	1	0	1	1	0	1	1	1	1	1
0	1	1	1	1	1	1	0	1	1	1	1	1
1	0	0	0	0	1	1	1	1	1	1	1	0
1	0	0	0	1	1	1	1	0	0	1	1	0
1	0	0	1	0	1	1	1	1	1	1	1	0
1	0	0	1	1	1	1	1	0	0	1	1	0
1	0	1	0	0	1	1	1	1	1	1	0	1
1	0	1	0	1	1	1	1	1	0	0	1	0
1	0	1	1	0	1	1	1	1	0	0	1	1
1	0	1	1	1	1	1	1	0	0	1	1	1
1	1	0	0	0	1	0	1	1	0	1	1	1
1	1	0	0	1	1	0	1	1	0	1	1	1
1	1	0	1	0	0	0	1	1	1	1	1	1
1	1	0	1	1	0	0	1	1	1	1	1	1
1	1	1	0	0	1	1	0	1	1	1	1	1
1	1	1	0	1	1	1	0	1	1	1	1	1
1	1	1	1	0	1	1	0	1	1	1	1	1
1	1	1	1	1	1	1	0	1	1	1	1	1

Figure 7-6. Truth Table Bipolar Prom

### Appendix 7.5 CHARACTER GENERATOR

As previously mentioned, the character generator used in this terminal is a 2716 or 2758 EPROM. A 1K by 8 device is sufficient since a 128 character 5 by 7 dot matrix only requires 8K of memory. Any "standard" or custom character generator could have been used.

The three low-order line count outputs (LC0-LC2) from the 8275 are connected to the three low-order address lines of the character generator and the seven character generator outputs (CC0-CC6) are connected to A3-A9 of the character generator. The output from the character generator is loaded into a shift register and the serial output from the shift register is the video output of the terminal.

Now, let's assume that the letter "E" is to be displayed. The ASCII code for "E" is 45H. So, 45H is presented to address lines A2-A9 of the character generator. The scan lines will now count each line from zero to seven to "form" the character as shown in Fig. 7.7. This same procedure is used to form all 128 possible characters.

It should be obvious that "custom" character fonts could be made just by changing the bit patterns in the character generator PROM. For reference, Appendix 7.6 contains a HEX dump of the character generator used in this terminal.

45H = 01000101  
 Address to Prom = 01000101 SL2 SL1 SL0  
 = 228H - 22FH  
 Depending on state of Scan lines.

Character generator output

Rom Address	Rom Hex	Output	Bit Output*
			0 1 2 3 4 5 6 7
228H	3E		XXXXXX
229H	02		XXXXXX
22AH	02		XXXXXX
22BH	0E		XXXXXX
22CH	02		XXXXXX
22DH	02		XXXXXX
22EH	3E		XXXXXX
22FH	00		XXXXXX

Bits 0, 6 and 7 are not used.

\* note bit output is backward from convention.

Figure 7-7. Character Generation



## Appendix 7.7 COMPOSITE VIDEO

In this design, it was assumed that the monitor required a separate horizontal drive, vertical drive, and video input. However, many monitors require a composite video signal. The schematic shown in Figure 7.8 illustrates how to generate a composite video signal from the output of the 8275.

The dual one-shots are used to provide a small delay and the proper horizontal and vertical pulse to the composite video monitor. The delay introduced in the vertical and horizontal timing is used to "center" the display. VR1 and VR2 control the amount of delay. IC3 is used to mix the vertical and horizontal retrace and Q1 along with the R1, R2, and R3 mix the video and the retrace signal and provide the proper DC levels.

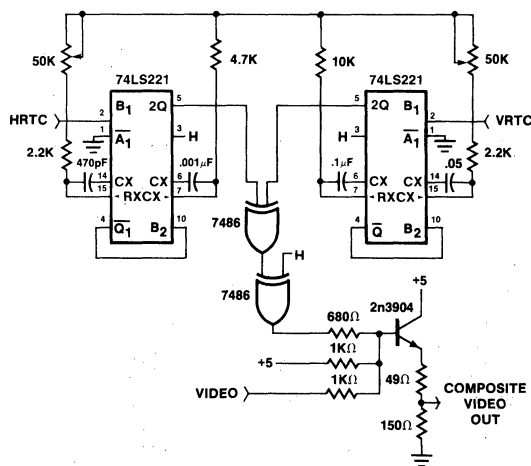


Figure 7-8. Composite Video

## Appendix 7.8 SOFTWARE LISTINGS

ISIS-II 8080/8085 MACRO ASSEMBLER, X108

LOC	OBJ	SEQ	SOURCE STATEMENT
		1	\$MOD85    MACROFILE
		2	;NO DMA 8275 SOFTWARE ALL I/O IS MEMORY MAPPED
		3	;SYSTEM ROM 0000H TO 07FFH
		4	;SYSTEM RAM 0800H TO 0FFFH
		5	;8275 WRITE 1000H TO 13FFH
		6	;8275 READ 1400H TO 17FFH
		7	;8255 READ/WRITE 1800H TO 1FFF
		8	;8253 ENABLED BY A14
		9	;8251 ENABLED BY A15
1800		10	PORTA    EQU    1800H    ;8255 PORT A ADDRESS
1801		11	PORTB    EQU    1801H    ;8255 PORT B ADDRESS
1802		12	PORTC    EQU    1802H    ;8255 PORT C ADDRESS
1803		13	CNWD55   EQU    1803H    ;8255 CONTROL PORT ADDRESS
A001		14	USTF     EQU    0A001H    ;8251 FLAGS
A000		15	USTD     EQU    0A000H    ;8251 DATA
6000		16	CNT0     EQU    6000H    ;8253 COUNTER 0
6001		17	CNT1     EQU    6001H    ;8253 COUNTER 1
6002		18	CNT2     EQU    6002H    ;8253 COUNTER 2
6003		19	CNTM     EQU    6003H    ;8253 MODE WORD
1001		20	CRTS     EQU    1001H    ;8275 CONTROL ADDRESS
1000		21	CRTM     EQU    1000H    ;8275 MODE ADDRESS
1401		22	INT75    EQU    1401H    ;8275 INTERRUPT CLEAR
0800		23	TPDIS    EQU    0800H    ;TOP OF DISPLAY RAM
0F80		24	BTDIS    EQU    0F80H    ;BOTTOM OF DISPLAY RAM
0FD0		25	LAST     EQU    0FD0H    ;FIRST BYTE AFTER DISPLAY
0018		26	CURBOT   EQU    18H       ;BOTTOM Y CURSOR
0050		27	LNPTH    EQU    0050H    ;LENGTH OF ONE LINE
0FE0		28	STPTR    EQU    0FE0H    ;LOCATION OF STACK POINTER
		29	
		30	; START PROGRAM
		31	; ALL VARIABLES ARE INITIALIZED BEFORE ANYTHING ELSE
		32	
0000	F3	33	DI        ;DISABLE INTERRUPTS
0001	31E00F	34	LXI    SP,STPTR    ;LOAD STACK POINTER
0004	210008	35	LXI    H,TPDIS    ;LOAD H&L WITH TOP OF DISPLAY
0007	22E30F	36	SHLD   TOPAD      ;SET TOP = TOP OF DISPLAY
000A	22E80F	37	SHLD   CURAD      ;STORE THE CURRENT ADDRESS
000D	3E00	38	MVI    A,00H      ;ZERO A
000F	32E10F	39	STA    CURSY      ;ZERO CURSOR Y POINTER
0012	32E20F	40	STA    CURSX      ;ZERO CURSOR X POINTER
0015	32EB0F	41	STA    KBCHR      ;ZERO KBD CHARACTER
0018	32E70F	42	STA    USCHR      ;ZERO USART CHAR BUFFER
001B	32EA0F	43	STA    KEYDWN     ;ZERO KEY DOWN

# APPLICATIONS

```

001E 32ED0F      44      STA      KEYOK      ;ZERO KEYOK
0021 32EE0F      45      STA      ESCP       ;ZERO ESCAPE
0024 C39800      46      JMP      LPKBD      ;JUMP AND SET EVERYTHING UP
;
47
48      ;THIS JUMP VECTOR IS LOCATED AT THE RST 5.5 LOCATION
49      ;OF THE 8085. IT IS USED TO READ THE 8275 STATUS AND
50      ;READ THE KEYBOARD. THIS ROUTINE IS EXECUTED ONCE EVERY
51      ;16.667 MILLISECONDS.
52
002C           53      ORG      002CH
002C C36701      54      JMP      FRAME
;
55
56      ;THIS ROUTINE IS LOCATED AT THE RST 6.5 LOCATION OF THE
57      ;8085 AND IS USED TO LOAD THE DATA TO BE DISPLAYED INTO
58      ;THE 8275. THIS ROUTINE IS EXECUTED ONCE EVERY 617 MICROSECONDS.
59
0034           60      ORG      34H
0034 F5          61      POPDAT: PUSH  PSW      ;SAVE A AND FLAGS
0035 E5          62      PUSH  H        ;SAVE H AND L
0036 D5          63      PUSH  D        ;SAVE D AND E
0037 210000     64      LXI  H,0000H  ;ZERO H AND L
003A 39          65      DAD   SP      ;PUT STACK POINTER IN H AND L
003B EB          66      XCHG      ;PUT STACK IN D AND E
003C 2AE80F     67      LHLD   CURAD   ;GET POINTER
003F F9          68      SPHL      ;PUT CURRENT LINE INTO SP
0040 3EC0        69      MVI   A,0C0H  ;SET MASK FOR SIM
0042 30          70      SIM
;
71      REPT  (LNTH/2)
72      POP   H
73      ENDM
0043 E1          74+     POP   H
0044 E1          75+     POP   H
0045 E1          76+     POP   H
0046 E1          77+     POP   H
0047 E1          78+     POP   H
0048 E1          79+     POP   H
0049 E1          80+     POP   H
004A E1          81+     POP   H
004B E1          82+     POP   H
004C E1          83+     POP   H
004D E1          84+     POP   H
004E E1          85+     POP   H
004F E1          86+     POP   H
0050 E1          87+     POP   H
0051 E1          88+     POP   H
0052 E1          89+     POP   H
0053 E1          90+     POP   H
0054 E1          91+     POP   H
0055 E1          92+     POP   H
0056 E1          93+     POP   H
0057 E1          94+     POP   H
0058 E1          95+     POP   H
0059 E1          96+     POP   H
005A E1          97+     POP   H
005B E1          98+     POP   H
005C E1          99+     POP   H
005D E1         100+    POP   H
005E E1         101+    POP   H
005F E1         102+    POP   H
0060 E1         103+    POP   H
0061 E1         104+    POP   H
0062 E1         105+    POP   H
0063 E1         106+    POP   H
0064 E1         107+    POP   H
0065 E1         108+    POP   H
0066 E1         109+    POP   H
0067 E1         110+    POP   H
0068 E1         111+    POP   H
0069 E1         112+    POP   H
006A E1         113+    POP   H
006B 0F         114      RRC
006C 30         115      SIM
006D 210000     116      LXI  H,0000H
0070 39         117      DAD   SP
0071 EB         118      XCHG      ;ADD STACK
0072 F9         119      SPHL      ;PUT STACK IN H AND L
0073 21D00F     120      LXI  H, LAST  ;RESTORE STACK
0076 EB         121      XCHG      ;PUT BOTTOM DISPLAY IN H AND L
0077 7A         122      MOV   A,D      ;SWAP REGISTERS
0078 BC         123      CMP   H        ;PUT HIGH ORDER IN A
0079 C28400     124      JNZ  KPTK     ;SEE IF SAME AS H
007C 7B         125      MOV   A,E      ;IF NOT LEAVE
007D BD         126      CMP   L        ;PUT LOW ORDER IN A
007E C28400     127      JNZ  KPTK     ;SEE IF SAME AS L
0081 210008     128      LXI  H,TPDIS ;IF NOT LEAVE
0084 22E80F     129      KPTK: SHLD CURAD ;LOAD H AND L WITH TOP OF SCREEN MEMORY
0087 3E18       130      MVI  A,18H   ;PUT BACK CURRENT ADDRESS
0089 30         131      SIM        ;SET MASK
;
;OUTPUT MASK

```

# APPLICATIONS

```

008A D1      132      POP      D      ;GET D AND E
008B E1      133      POP      H      ;GET H AND L
008C F1      134      POP      PSW     ;GET A AND FLAGS
008D FB      135      EI          ;TURN ON INTERRUPTS
008E C9      136      RET         ;GO BACK
137
138          ; THIS IS THE EXIT ROUTINE FOR THE FRAME INTERRUPT
139
008F 3E18    140  BYPASS: MVI      A,18H     ;SET MASK
0091 30      141      SIM         ;OUTPUT THE MASK
0092 C1      142      POP         B      ;GET B AND C
0093 D1      143      POP         D      ;GET D AND E
0094 E1      144      POP         H      ;GET H AND L
0095 F1      145      POP         PSW    ;GET A AND FLAGS
0096 FB      146      EI          ;ENABLE INTERRUPTS
0097 C9      147      RET         ;GO BACK
148
149          ; THIS CLEARS THE AREA OF RAM THAT IS USED
150          ; FOR KEYBOARD DEBOUNCE.
151
0098 32EF0F  152  LPKBD: STA      SHCON    ;ZERO SHIFT CONTROL
009B 32F00F  153      STA      RETLIN   ;ZERO RETURN LINE
009E 32F10F  154      STA      SCNLIN   ;ZERO SCAN LINE
155
156          ; THIS ROUTINE CLEARS THE ENTIRE SCREEN BY PUTTING
157          ; SPACE CODES (20H) IN EVERY LOCATION ON THE SCREEN.
158
00A1 210008  159      LXI      H,TPDIS   ;PUT TOP OF SCREEN IN HL
00A4 01D00F  160      LXI      B,LAST    ;PUT BOTTOM IN BC
00A7 3520    161  LOOPF: MVI      M,20H   ;PUT SPACE IN M
00A9 23      162      INX         H      ;INCREMENT POINTER
00AA 7C      163      MOV         A,H      ;GET H
00AB B8      164      CMP         B      ;SEE IF SAME AS B
00AC C2A700  165      JNZ      LOOPF    ;IF NOT LOOP AGAIN
00AF 7D      166      MOV         A,L      ;GET L
00B0 B9      167      CMP         C      ;SEE IF SAME AS C
00B1 C2A700  168      JNZ      LOOPF    ;IF NOT LOOP AGAIN
169
170          ; 8255 INITIALIZATION
171
00B4 3E8B    172      MVI      A,8BH     ;MOVE 8255 CONTROL WORD INTO A
00B6 320318  173      STA      CNWD55    ;PUT CONTROL WORD INTO 8255
174
175          ; 8251 INITIALIZATION
176
00B9 2101A0  177      LXI      H,USTF     ;GET 8251 FLAG ADDRESS
00BC 3580    178      MVI      M,80H     ;DUMMY STORE TO 8251
00BE 3600    179      MVI      M,00H     ;RESET 8251
00C0 3640    180      MVI      M,40H     ;RESET 8251
00C2 00      181      NOP          ;WAIT
00C3 35EA    182      MVI      M,0EAH   ;LOAD 8251 MODE WORD
00C5 3605    183      MVI      M,05H     ;LOAD 8251 COMMAND WORD
184
185          ; 8253 INITIALIZATION
186
00C7 3E32    187      MVI      A,32H     ;CONTROL WORD FOR 8253
00C9 320360  188      STA      CNTM     ;PUT CONTROL WORD INTO 8253
00CC 3E32    189      MVI      A,32H     ;LSB 8253
00CE 320060  190      STA      CNT0     ;PUT IT IN 8235
00D1 3E00    191      MVI      A,00H     ;MSB 8253
00D3 320060  192      STA      CNT0     ;PUT IT IN 8253
00D6 CDDC00  193      CALL    STBAUD    ;GO DO BAUD RATE
00D9 C3F900  194      JMP         IN75    ;GO DO 8275
195
196          ; THIS ROUTINE READS THE BAUD RATE SWITCHES FROM FORT C
197          ; OF THE 8255 AND LOOKS UP THE NUMBERS NEEDED TO LOAD
198          ; THE 8253 TO PROVIDE THE PROPER BAUD RATE.
199
00DC 3A0218  200  STBAUD: LDA      PORTC   ;READ BAUD RATE SWITCHES
00DF E60F    201      ANI      0FH      ;STRIP OFF 4 MSB'S
00E1 32EC0F  202      STA      BAUD     ;SAVE IT
00E4 07      203      RLC          ;MOVE BITS OVER ONE PLACE
00E5 21C505  204      LXI      H,BDLK   ;GET BAUD RATE LOOK UP TABLE
00E8 1600    205      MVI      D,00H     ;ZERO D
00EA 5F      206      MOV         E,A      ;PUT A IN E
00EB 19      207      DAD         D      ;GET OFFSET
00EC 110360  208      LXI      D,CNTM    ;POINT DE TO 8253
00EF 3EB6    209      MVI      A,0B6H   ;GET CONTROL WORD
00F1 12      210      STAX     D      ;STORE IN 8253
00F2 1B      211      DCX         D      ;POINT AT #2 COUNTER
00F3 7E      212      MOV         A,M      ;GET LSB BAUD RATE
00F4 12      213      STAX     D      ;PUT IT IN 8253
00F5 23      214      INX         H      ;POINT AT MSB BAUD RATE
00F6 7E      215      MOV         A,M      ;GET MSB BAUD RATE
00F7 12      216      STAX     D      ;PUT IT IN 8253
00F8 C9      217      RET         ;GO BACK
218

```



# APPLICATIONS

```

219 ;8275 INITIALIZATION
220
00F9 210110 221 IN75: LXI H,CRTS
00FC 3500 222 MVI M,00H ;RESET AND STOP DISPLAY
00FE 2B 223 DCX H ;HL=1000H
00FF 364F 224 MVI M,4FH ;SCREEN PARAMETER BYTE 1
0101 3658 225 MVI M,58H ;SCREEN PARAMETER BYTE 2
0103 3689 226 MVI M,89H ;SCREEN PARAMETER BYTE 3
0105 36DD 227 MVI M,0DDH ;SCREEN PARAMETER BYTE 4
0107 23 228 INX H ;HL=1001H
0108 CDB803 229 CALL LDCUR ;LOAD THE CURSOR
010B 36E0 230 MVI M,0E0H ;PRESET COUNTERS
010D 3623 231 MVI M,23H ;START DISPLAY
232 ;
233 ; THIS ROUTINE READS BOTH THE KEYBOARD AND THE USART
234 ; AND TAKES PROPER ACTION DEPENDING ON HOW THE LINE-LOCAL
235 ; SWITCH IS SET
236
010F 3E18 237 SETUP: MVI A,18H ;SET MASK
0111 30 238 SIM ;LOAD MASK
0112 FB 239 EI ;ENABLE INTERRUPTS
240 ;
241 ; READ THE USART
242 ;
0113 20 243 RXRDY: RIM ;GET LINE LOCAL
0114 E680 244 ANI 80H ;IS IT ON OR OFF?
0116 C22101 245 JNZ KEYINP ;LEAVE IF IT IS ON
0119 3A01A0 246 LDA USTF ;READ 8251 FLAGS
011C E602 247 ANI 02H ;LOOK AT RXRDY
011E C25C01 248 JNZ OK7 ;IF HAVE CHARACTER GO TO WORK
0121 3AEA0F 249 KEYINP: LDA KEYDWN ;GET KEYBOARD CHARACTER
0124 E680 250 ANI 80H ;IS IT THERE
0126 C23101 251 JNZ KEYS ;IF KEY IS PUSHED LEAVE
0129 3E00 252 MVI A,00H ;ZERO A
012B 32ED0F 253 STA KEYOK ;CLEAR KEYOK
012E C31301 254 JMP RXRDY ;LOOP AGAIN
0131 3AED0F 255 LDA KEYOK ;WAS KEY DOWN
0134 4F 256 MOV C,A ;SAVE A IN C
0135 3AEB0F 257 LDA KBCHR ;GET KEYBOARD CHARACTER
0138 B9 258 CMP C ;IS IT THE SAME AS KEYOK
0139 CA1301 259 JZ RXRDY ;IF SAME LOOP AGAIN
013C 32ED0F 260 STA KEYOK ;IF NOT SAVE IT
013F 32E70F 261 STA USCHR ;SAVE IT
0142 20 262 RIM ;GET LINE LOCAL
0143 E680 263 ANI 80H ;WHICH WAY
0145 CA4B01 264 JZ TRANS ;LEAVE IF LINE
0148 C34E02 265 JMP CHREC ;TIME TO DO SOME WORK
014B 3A01A0 266 TRANS: LDA USTF ;GET USART FLAGS
014E E601 267 ANI 01H ;READY TO TRANSMIT?
0150 CA4B01 268 JZ TRANS ;LOOP IF NOT READY
0153 3AE70F 269 LDA USCHR ;GET CHARACTER
0156 3200A0 270 STA USTD ;PUT IN USART
0159 C30F01 271 JMP SETUP ;LEAVE
015C 3A00A0 272 OK7: LDA USTD ;READ USART
015F E67F 273 ANI 07FH ;STRIP MSB
0161 32E70F 274 STA USCHR ;PUT IT IN MEMORY
0164 C34E02 275 JMP CHREC ;LEAVE
276 ;
277 ; THIS ROUTINE CHECKS THE BAUD RATE SWITCHES, RESETS THE
278 ; SCREEN POINTERS AND READS AND LOOKS UP THE KEYBOARD.
279 ;
0167 F5 280 FRAME: PUSH PSW ;SAVE A AND FLAGS
0168 E5 281 PUSH H ;SAVE H AND L
0169 D5 282 PUSH D ;SAVE D AND E
016A C5 283 PUSH B ;SAVE B AND C
016B 3A0114 284 LDA INT75 ;READ 8275 TO CLEAR INTERRUPT
285 ;
286 ; SET UP THE POINTERS
287 ;
016E 2AE30F 288 LHLD TOPAD ;LOAD TOP IN H AND L
0171 22E80F 289 SHLD CURAD ;STORE TOP IN CURRENT ADDRESS
290 ;
291 ; SET UP BAUD RATE
292 ;
0174 3A0218 293 LDA PORTC ;READ BAUD RATE SWITCHES
0177 E60F 294 ANI 0FH ;STRIP OFF 4 MSB'S
0179 47 295 MOV B,A ;SAVE IN B
017A 3AEC0F 296 LDA BAUD ;GET BAUD RATE
017D B8 297 CMP B ;SEE IF SAME AS B
017E C4DC00 298 CNZ STBAUD ;IF NOT SAME DO SOMETHING
299 ;
300 ; READ KEYBOARD
301 ;
0181 3AEA0F 302 LDA KEYDWN ;SEE IF A KEY IS DOWN
0184 E640 303 ANI 40H ;SET THE FLAGS
0186 C2C201 304 JNZ KYDOWN ;IF KEY IS DOWN JUMP AROUND
0189 CD8F01 305 CALL RDKB ;GO READ THE KEYBOARD
018C C38F00 306 JMP BYPASS ;LEAVE

```

# APPLICATIONS

```

018F 21EF0F      307 RDKB: LXI      H,SHCON      ;POINT HL AT KEYBOARD RAM
0192 3A0218      308 LDA      PORTC    ;GET CONTROL AND SHIFT
0195 77           309 MOV      M,A      ;SAVE IN MEMORY
0196 3EFE        310 MVI     A,0FEH   ;SET UP A
0198 320018      311 LOOPK: STA     PORTA ;OUTPUT A
019B 47          312 MOV      B,A      ;SAVE A IN B
019C 3A0118      313 LDA      PORTB   ;READ KEYBOARD
019F 2F          314 CMA      ;INVERT A
01A0 B7          315 ORA      A        ;SET THE FLAGS
01A1 C2AF01      316 JNZ     SAVKEY   ;LEAVE IF KEY IS DOWN
01A4 78          317 MOV      A,B      ;GET SCAN LINE BACK
01A5 07          318 RLC      ;ROTATE IT OVER ONE
01A6 DA9801      319 JC      LOOPK    ;DO IT AGAIN
01A9 3E00        320 MVI     A,00H    ;ZERO A
01AB 32EA0F      321 STA     KEYDOWN  ;SAVE KEY DOWN
01AE C9          322 RET      ;LEAVE
01AF 23          323 SAVKEY: INX     H    ;POINT AT RETURN LINE
01B0 2F          324 CMA      ;PUT A BACK
01B1 77          325 MOV      M,A      ;SAVE RETURN LINE IN MEMORY
01B2 23          326 INX     H        ;POINT H AT SCAN LINE
01B3 70          327 MOV      M,B      ;SAVE SCAN LINE IN MEMORY
01B4 3E40        328 MVI     A,40H    ;SET A
01B6 32EA0F      329 STA     KEYDOWN  ;SAVE KEY DOWN
01B9 C9          330 RET      ;LEAVE
01BA 3E00        331 KYCHNG: MVI    A,00H ;ZERO 0
01BC 32EA0F      332 STA     KEYDOWN  ;RESET KEY DOWN
01BF C38F00      333 JMP     BYPASS   ;LEAVE
01C2 21F10F      334 KYDOWN: LXI    H,SCNLIN ;GET SCAN LINE
01C5 7E          335 MOV      A,M      ;PUT SCAN LINE IN A
01C6 320018      336 STA     PORTA    ;OUTPUT SCAN LINE TO PORT A
01C9 2B          337 DCX     H        ;POINT AT RETURN LINE
01CA 3A0118      338 LDA      PORTB   ;GET RETURN LINES
01CD B6          339 ORA      M        ;ARE THEY THE SAME?
01CE 2F          340 CMA      ;INVERT A
01CF B7          341 ORA      A        ;SET FLAGS
01D0 CABA01      342 JZ      KYCHNG   ;IF DIFFERENT KEY HAS CHANGED
01D3 3AEA0F      343 LDA     KEYDOWN  ;GET KEY DOWN
01D6 E601        344 ANI    01H      ;HAS THIS BEEN DONE BEFORE?
01D8 C28F00      345 JNZ    BYPASS   ;LEAVE IF IT HAS
01DB 3A0118      346 LDA     PORTB   ;GET RETURN LINE
01DE 06FF        347 MVI     B,0FFH  ;GET READY TO ZERO B
01E0 04          348 UP:    INR     B    ;ZERO B
01E1 0F          349 RRC     ;ROTATE A
01E2 DAE001      350 JC      UP      ;DO IT AGAIN
01E5 23          351 INX     H        ;POINT H AT SCAN LINES
01E6 7E          352 MOV      A,M      ;GET SCAN LINES
01E7 0EFF        353 MVI     C,0FFH  ;GET READY TO LOOP
01E9 0C          354 UP1:  INR     C    ;START C COUNTING
01EA 0F          355 RRC     ;ROTATE A
01EB DAE901      356 JC      UP1    ;JUMP TO LOOP
01EE 78          357 MOV      A,B      ;GET RETURN LINES
01EF 07          358 RLC     ;MOVE OVER ONCE
01F0 07          359 RLC     ;MOVE OVER TWICE
01F1 07          360 RLC     ;MOVE OVER THREE TIMES
01F2 B1          361 ORA     C        ;OR SCAN AND RETURN LINES
01F3 47          362 MOV      B,A      ;SAVE A IN B
01F4 3A0218      363 LDA     PORTC    ;GET SHIFT CONTROL
01F7 E640        364 ANI    40H      ;IS CONTROL SET
01F9 4F          365 MOV      C,A      ;SAVE A IN C
01FA 3AEF0F      366 LDA     SHCON   ;GET SHIFT CONTROL
01FD 57          367 MOV      D,A      ;SAVE A IN D
01FE E640        368 ANI    40H      ;STRIP CONTROL
0200 B1          369 ORA     C        ;SET BIT
0201 CA3E02      370 JZ      CNTDWN  ;IF SET LEAVE
0204 3A0218      371 LDA     PORTC    ;READ IT AGAIN
0207 E620        372 ANI    20H      ;STRIP SHIFT
0209 4F          373 MOV      C,A      ;SAVE A
020A 7F          374 MOV      A,D      ;GET SHIFT CONTROL
020B E620        375 ANI    20H      ;STRIP CONTROL
020D B1          376 ORA     C        ;ARE THEY THE SAME?
020E CA4702      377 JZ      SHDWN   ;IF SET LEAVE
0211 58          378 SCR:  MOV      E,B    ;PUT TARGET IN E
0212 1600        379 MVI     D,00H    ;ZERO D
0214 210705      380 LXI    H,KYLKUP ;GET LOOKUP TABLE
0217 19          381 DAD     D        ;GET OFFSET
0218 7E          382 MOV      A,M      ;GET CHARACTER
0219 47          383 MOV      B,A      ;PUT CHARACTER IN B
021A 3A0218      384 LDA     PORTC    ;GET PORTC
021D E610        385 ANI    10H      ;STRIP BIT
021F CA2E02      386 JZ      CAPLOC  ;CAPS LOCK
0222 78          387 MOV      A,B      ;GET A BACK
0223 32EB0F      388 STKEY: STA    KBCHR ;SAVE CHARACTER
0226 3EC1        389 MVI     A,0C1H  ;SET A
0228 32EA0F      390 STA     KEYDOWN  ;SAVE KEY DOWN
022B C38F00      391 JMP     BYPASS   ;LEAVE
392 ;
393 ;IF THE CAP LOCK BUTTON IS PUSHED THIS ROUTINE SEES IF
394 ;THE CHARACTER IS BETWEEN 61H AND 7AH AND IF IT IS THIS

```

# APPLICATIONS

```

395 ;ROUTINE ASSUMES THAT THE CHARACTER IS LOWER CASE ASCII
396 ;AND SUBTRACTS 20H, WHICH CONVERTS THE CHARACTER TO
397 ;UPPER CASE ASCII
398 ;
022E 78 399 CAPLOC: MOV A,B ;GET A BACK
022F FE60 400 CPI 60H ;HOW BIG IS IT?
0231 DA2302 401 JC STKEY ;LEAVE IF IT'S TOO SMALL
0234 FE7B 402 CPI 7BH ;IS IT TOO BIG
0236 D22302 403 JNC STKEY ;LEAVE IF TOO BIG
0239 D620 404 SUI 20H ;ADJUST A
023B C32302 405 JMP STKEY ;STORE THE KEY
406 ;
407 ;THE ROUTINES SHDWN AND CNTDWN SET BIT 6 AND 7 RESPECTIVLY
408 ;IN THE ACC.
409 ;
023E 3E80 410 CNTDWN: MVI A,80H ;SET BIT 7 IN A
0240 B0 411 ORA B ;OR WITH CHARACTER
0241 E6BF 412 ANI 0BFH ;MAKE SURE SHIFT IS NOT SET
0243 47 413 MOV B,A ;PUT IT BACK IN B
0244 C31102 414 JMP SCR ;GO BACK
0247 3E40 415 SHDWN: MVI A,40H ;SET BIT 6 IN A
0249 B0 416 ORA B ;OR WITH CHARACTER
024A 47 417 MOV B,A ;PUT IT BACK IN B
024B C31102 418 JMP SCR ;GO BACK
419 ;
420 ;THIS ROUTINE CHECKS FOR ESCAPE CHARACTERS, LF, CR,
421 ;FF, AND BACK SPACE
422 ;
024E 3AE0F 423 CHREC: LDA ESCP ;ESCAPE SET?
0251 FE80 424 CPI 80H ;SEE IF IT IS
0253 CA7B02 425 JZ ESSQ ;LEAVE IF IT IS
0256 3AE70F 426 LDA USCHR ;GET CHARACTER
0259 FE0A 427 CPI 0AH ;LINE FEED
025B CAF603 428 JZ LNFD ;C) TO LINE FEED
025E FE0C 429 CPI 0CH ;FORM FEED
0260 CACA03 430 JZ FMFD ;GO TO FORM FEED
0263 FE0D 431 CPI 0DH ;CR
0265 CAAD03 432 JZ CGRT ;DO A CR
0268 FE08 433 CPI 08H ;BACK SPACE
026A CA6E03 434 JZ LEFT ;DO A BACK SPACE
026D FE1B 435 CPI 1BH ;ESCAPE
026F CAA503 436 JZ ESKAP ;DO AN ESCAPE
0272 B7 437 ORA A ;CLEAR CARRY
0273 C6E0 438 ADI 0E0H ;SEE IF CHARACTER IS PRINTABLE
0275 DA7704 439 JC CHRPUT ;IF PRINTABLE DO IT
0278 C30F01 440 JMP SETUP ;GO BACK AND READ USART AGAIN
441 ;
442 ;THIS ROUTINE RESETS THE ESCAPE LOCATION AND DECODES
443 ;THE CHARACTERS FOLLOWING AN ESCAPE. THE COMMANDS ARE
444 ;COMPATABLE WITH INTELS CREDIT TEXT EDITOR
445 ;
027B 3E00 446 ESSQ: MVI A,00H ;ZERO A
027D 32E0F 447 STA ESCP ;RESET ESCP
0280 3AE70F 448 LDA USCHR ;GET CHARACTER
0283 FE42 449 CPI 42H ;DOWN
0285 CAAE02 450 JZ DOWN ;MOVE CURSOR DOWN
0288 FE45 451 CPI 45H ;CLEAR SCREEN CHARACTER
028A CACF02 452 JZ CLEAR ;CLEAR THE SCREEN
028D FE4A 453 CPI 4AH ;CLEAR REST OF SCREEN
028F CAD502 454 JZ CLRST ;GO CLEAR THE REST OF THE SCREEN
0292 FE4B 455 CPI 4BH ;CLEAR LINE CHARACTER
0294 CA2703 456 JZ CLRLIN ;GO CLEAR A LINE
0297 FE41 457 CPI 41H ;CURSOR UP CHARACTER
0299 CA3303 458 JZ UPCUR ;MOVE CURSOR UP
029C FE43 459 CPI 43H ;CURSOR RIGHT CHARACTER
029E CA4503 460 JZ RIGHT ;MOVE CURSOR TO THE RIGHT
02A1 FE44 461 CPI 44H ;CURSOR LEFT CHARACTER
02A3 CA6E03 462 JZ LEFT ;MOVE CURSOR TO THE LEFT
02A6 FE48 463 CPI 48H ;HOME CURSOR CHARACTER
02A8 CA9703 464 JZ HOME ;HOME THE CURSOR
02AB C30F01 465 JMP SETUP ;LEAVE
466 ;
467 ;THIS ROUTINE MOVES THE CURSOR DOWN ONE CHARACTER LINE
468 ;
02AE 3AE10F 469 DOWN: LDA CURSY ;PUT CURSOR Y IN A
02B1 FE18 470 CPI CURBOT ;SEE IF ON BOTTOM OF SCREEN
02B3 CA0F01 471 JZ SETUP ;LEAVE IF ON BOTTOM
02B6 3C 472 INR A ;INCREMENT Y CURSOR
02B7 32E10F 473 STA CURSY ;SAVE NEW CURSOR
02BA CDB803 474 CALL LDCUR ;LOAD THE CURSOR
02BD CDA504 475 CALL CALCU ;CALCULATE ADDRESS
02C0 7E 476 MOV A,N ;GET FIRST LOCATION OF THE LINE
02C1 FEF0 477 CPI 0E0H ;SEE IF CLEAR SCREEN CHARACTER
02C3 C20F01 478 JNZ SETUP ;LEAVE IF IT IS NOT
02C6 22E50F 479 SHLD LOC80 ;SAVE BEGINNING OF THE LINE
02C9 CD1504 480 CALL CLLINE ;CLEAR THE LINE
02CC C30F01 481 JMP SETUP ;LEAVE
482 ;

```

# APPLICATIONS

```

483 ;THIS ROUTINE CLEARS THE SCREEN.
484
02CF CDE403 485 CLEAR: CALL CLSCR ;GO CLEAR THE SCREEN
02D2 C30F01 486 JMP SETUP ;GO BACK
487
488 ;THIS ROUTINE CLEARS ALL LINES BENEATH THE LOCATION
489 ;OF THE CURSOR.
490
02D5 CDA504 491 CLRST: CALL CALCU ;CALCULATE ADDRESS
02D8 CDCD04 492 CALL ADX ;ADD X POSITION
02DB 01204F 493 LXI B,4F20H ;PUT SPACE AND LAST X IN B AND C
02DE 3AE20F 494 LDA CURSX ;GET X CURSOR
02E1 B8 495 CMP B ;SEE IF AT END OF LINE
02E2 CAB02 496 JZ OVR1 ;LEAVE IF X IS AT END OF LINE
02E5 3C 497 LLP: INR A ;MOVE A OVER ONE X POSITION
02E6 23 498 INX H ;INCREMENT MEMORY POINTER
02E7 71 499 MOV M,C ;PUT A SPACE IN MEMORY
02E8 B8 500 CMP B ;SEE IF A = 4FH
02E9 C2E502 501 JNZ LLP ;IF NOT LOOP AGAIN
02EC 01D00F 502 OVR1: LXI B, LAST ;PUT LAST LINE IN BC
02EF 23 503 INX H ;POINT HL TO LAST LINE
02F0 78 504 MOV A,B ;GET B
02F1 BC 505 CMP H ;SAME AS H?
02F2 C2FD02 506 JNZ CONCL ;LEAVE IF NOT
02F5 79 507 MOV A,C ;GET C
02F6 BD 508 CMP L ;SAME AS L?
02F7 C2FD02 509 JNZ CONCL ;LEAVE IF NOT
02FA 210008 510 LXI H,TPDIS ;GET TOP OF DISPLAY
02FD 3AE10F 511 CONCL: LDA CURSY ;GET Y CURSOR
0300 FE18 512 CPI CURBOT ;IS IT ON THE BOTTOM
0302 CA0F01 513 JZ SETUP ;LEAVE IF IT IS
0305 3C 514 INR A ;MOVE IT DOWN ONE LINE
0306 47 515 MOV B,A ;SAVE CURSOR IN B FOR LATER
0307 115000 516 LXI D,LENGTH ;PUT LENGTH OF ONE LINE IN D
030A 36F0 517 CLOOP: MVI M,0F0H ;PUT EOR IN MEMORY
030C 78 518 MOV A,B ;GET CURSOR Y
030D FE18 519 CPI CURBOT ;ARE WE ON THE BOTTOM
030F CA0F01 520 JZ SETUP ;LEAVE IF WE ARE
0312 3C 521 INR A ;MOVE CURSOR DOWN ONE
0313 19 522 DAD D ;GET NEXT LINE
0314 47 523 MOV B,A ;SAVE A
0315 7C 524 MOV A,H ;PUT H IN A
0316 FE0F 525 CPI 0FH ;COMPARE TO HIGH LAST
0318 C20A03 526 JNZ CLOOP ;LEAVE IF IT IS NOT
031B 7D 527 MOV A,L ;PUT L IN A
031C FED0 528 CPI 0D0H ;COMPARE TO LOW LAST
031E C20A03 529 JNZ CLOOP ;LEAVE IF IT IS NOT
0321 210008 530 LXI H,TPDIS ;PUT TOP DISPLAY IN H AND L
0324 C30A03 531 JMP CLOOP ;LOOP AGAIN
532
533 ;THIS ROUTINE CLEARS THE LINE THE CURSOR IS ON.
534
0327 CDA504 535 CLRRLIN: CALL CALCU ;CALCULATE ADDRESS
032A 22E50F 536 SHLD LOC80 ;STORE H AND L TO CLEAR LINE
032D CD1504 537 CALL CLLINE ;CLEAR THE LINE
0330 C30F01 538 JMP SETUP ;GO BACK
539
540 ;THIS ROUTINE MOVES THE CURSOR UP ONE LINE.
541
0333 3AE10F 542 UPCUR: LDA CURSY ;GET Y CURSOR
0336 FE00 543 CPI 00H ;IS IT ZERO
0338 CA0F01 544 JZ SETUP ;IF IT IS LEAVE
033B 3D 545 DCR A ;MOVE CURSOR UP
033C 32E10F 546 STA CURSY ;SAVE NEW CURSOR
033F CDB803 547 CALL LDCUR ;LOAD THE CURSOR
0342 C30F01 548 JMP SETUP ;LEAVE
549
550 ;THIS ROUTINE MOVES THE CURSOR ONE LOCATION TO THE RIGHT
551
0345 3AE20F 552 RIGHT: LDA CURSX ;GET X CURSOR
0348 FE4F 553 CPI 4FH ;IS IT ALL THE WAY OVER?
034A C26403 554 JNZ NTOVER ;IF NOT JUMP AROUND
034D 3AE10F 555 LDA CURSY ;GET Y CURSOR
0350 FE18 556 CPI CURBOT ;SEE IF ON BOTTOM
0352 CA5903 557 JZ GD18 ;IF WE ARE JUMP
0355 3C 558 INR A ;INCREMENT Y CURSOR
0356 32E10F 559 STA CURSY ;SAVE IT
0359 3E00 560 GD18: MVI A,00H ;ZERO A
035B 32E20F 561 STA CURSX ;ZERO X CURSOR
035E CDB803 562 CALL LDCUR ;LOAD THE CURSOR
0361 C30F01 563 JMP SETUP ;LEAVE
0364 3C 564 NTOVER: INR A ;INCREMENT X CURSOR
0365 32E20F 565 STA CURSX ;SAVE IT
0368 CDB803 566 CALL LDCUR ;LOAD THE CURSOR
036B C30F01 567 JMP SETUP ;LEAVE
568
569 ;THIS ROUTINE MOVES THE CURSOR LEFT ONE CHARACTER POSITION

```

# APPLICATIONS

```

570
036E 3AE20F 571 LEFT: LDA CURSX ;GET X CURSOR
0371 FE00 572 CPI 00H ;IS IT ALL THE WAY OVER
0373 C28D03 573 JNZ NOVER ;IF NOT JUMP AROUND
0376 3AE10F 574 LDA CURSY ;GET CURSOR Y
0379 FE00 575 CPI 00H ;IS IT ZERO?
037B CA0F01 576 JZ SETUP ;IF IT IS JUMP
037E 3D 577 DCR A ;MOVE CURSOR Y UP
037F 32E10F 578 STA CURSY ;SAVE IT
0382 3E4F 579 MVI A,4FH ;GET LAST X LOCATION
0384 32E20F 580 STA CURSX ;SAVE IT
0387 CDB803 581 CALL LDCUR ;LOAD THE CURSOR
038A C30F01 582 JMP SETUP
038D 3D 583 NOVER: DCR A ;ADJUST X CURSOR
038E 32E20F 584 STA CURSX ;SAVE CURSOR X
0391 CDB803 585 CALL LDCUR ;LOAD THE CURSOR
0394 C30F01 586 JMP SETUP ;LEAVE
587
588 ; THIS ROUTINE HOMES THE CURSOR.
589
0397 3E00 590 HOME: MVI A,00H ;ZERO A
0399 32E20F 591 STA CURSX ;ZERO X CURSOR
039C 32E10F 592 STA CURSY ;ZERO Y CURSOR
039F CDB803 593 CALL LDCUR ;LOAD THE CURSOR
03A2 C30F01 594 JMP SETUP ;LEAVE
595
596 ; THIS ROUTINE SETS THE ESCAPE BIT
597
03A5 3E80 598 ESKAP: MVI A,80H ;LOAD A WITH ESCAPE BIT
03A7 32EE0F 599 STA ESCP ;SET ESCAPE LOCATION
03AA C30F01 600 JMP SETUP ;GO BACK AND READ USART
601
602 ; THIS ROUTINE DOES A CR
603
03AD 3E00 604 CGRT: MVI A,00H ;ZERO A
03AF 32E20F 605 STA CURSX ;ZERO CURSOR X
03B2 CDB803 606 CALL LDCUR ;LOAD CURSOR INTO 8275
03B5 C30F01 607 JMP SETUP ;POLL USART AGAIN
608
609 ; THIS ROUTINE LOADS THE CURSOR
610
03B8 3E80 611 LDCUR: MVI A,80H ;PUT 80H INTO A
03BA 320110 612 STA CRTS ;LOAD CURSOR INTO 8275
03BD 3AE20F 613 LDA CURSX ;GET CURSOR X
03C0 320010 614 STA CRTM ;PUT IT IN 8275
03C3 3AE10F 615 LDA CURSY ;GET CURSOR Y
03C6 320010 616 STA CRTM ;PUT IT IN 8275
03C9 C9 617 RET
618
619 ; THIS ROUTINE DOES A FORM FEED
620
03CA CDE403 621 FMFD: CALL CLSCR ;CALL CLEAR SCREEN
03CD 210008 622 LXI H,TPDIS ;PUT TOP DISPLAY IN HL
03D0 22E50F 623 SHLD LOC80 ;PUT IT IN LOC80
03D3 CD1504 624 CALL CLLINE ;CLEAR TOP LINE
03D6 3E00 625 MVI A,00H ;ZERO A
03D8 32E20F 626 STA CURSX ;ZERO CURSOR X
03DB 32E10F 627 STA CURSY ;ZERO CURSOR Y
03DE CDB803 628 CALL LDCUR ;LOAD THE CURSOR
03E1 C30F01 629 JMP SETUP ;BACK TO USART
630
631 ; THIS ROUTINE CLEARS THE SCREEN BY WRITING END OF ROW
632 ; CHARACTERS INTO THE FIRST LOCATION OF ALL LINES ON
633 ; THE SCREEN.
634
03E4 3EF0 635 CLSCR: MVI A,0F0H ;PUT EOR CHARACTER IN A
03E6 0618 636 MVI B,CURBOT ;LOAD B WITH MAX Y
03E8 04 637 INR B ;GO TO MAX PLUS ONE
03E9 210008 638 LXI H,TPDIS ;LOAD H AND L WITH TOP OF RAM
03EC 115000 639 LXI D,LENGTH ;MOVE 50H = 80D INTO D AND E
03EF 77 640 MOV M,A ;MOVE EOR INTO MEMORY
03F0 19 641 DAD D ;CHANGE POINTER BY 80D
03F1 05 642 DCR B ;COUNT THE LOOPS
03F2 C2EF03 643 JNZ LOADX ;CONTINUE IF NOT ZERO
03F5 C9 644 RET ;GO BACK
645
646 ; THIS ROUTINE DOES A LINE FEED
647
03F6 CDFC03 648 LNFD: CALL LNFD1 ;CALL ROUTINE
03F9 C30F01 649 JMP SETUP ;POLL FLAGS
650
651 ; LINE FEED
652
03FC 3AE10F 653 LNFD1: LDA CURSY ;GET Y LOCATION OF CURSOR
03FF FE18 654 CPI CURBOT ;SEE IF AT BOTTOM OF SCREEN
0401 CA5304 655 JZ ONBOT ;IF WE ARE, LEAVE
0404 3C 656 INR A ;INCREMENT A
0405 32E10F 657 STA CURSY ;SAVE NEW CURSOR

```

# APPLICATIONS

```

0408 CDA504      658      CALL      CALCU      ;CALCULATE ADDRESS
040B 22E50F      659      SHLD     LOC80      ;SAVE TO CLEAR LINE
040E CD1504      660      CALL     CLLINE     ;CLEAR THE LINE
0411 CDB803      661      CALL     LDCUR      ;LOAD THE CURSOR
0414 C9           662      RET        ;LEAVE
                663      ;
                664      ; THIS ROUTINE CLEARS THE LINE WHOSE FIRST ADDRESS
                665      ; IS IN LOC80. PUSH INSTRUCTIONS ARE USED TO RAPIDLY
                666      ; CLEAR THE LINE
                667      ;
0415 F3           668      CLLINE:  DI          ;NO INTERRUPTS HERE
0416 2AE50F      669      LHL     LOC80      ;GET LOC80
0419 115000      670      LXI     D, LNTH    ;GET OFFSET
041C 19           671      DAD     D          ;ADD OFFSET
041D EB           672      XCHG    ;PUT START IN DE
041E 210000      673      LXI     H, 0000H  ;ZERO HL
0421 39           674      DAD     SP         ;GET STACK
0422 EB           675      XCHG    ;PUT STACK IN DE
0423 F9           676      SPHL   ;PUT START IN SP
0424 212020      677      LXI     H, 2020H  ;PUT SPACES IN HL
                678      ;
                679      ; NOW DO 40 PUSH INSTRUCTIONS TO CLEAR THE LINE
                680      ;
                681      REPT (LNTH/2)
                682      PUSH     H
                683      ENDM
0427 E5           684+     PUSH     H
0428 E5           685+     PUSH     H
0429 E5           686+     PUSH     H
042A E5           687+     PUSH     H
042B E5           688+     PUSH     H
042C E5           689+     PUSH     H
042D E5           690+     PUSH     H
042E E5           691+     PUSH     H
042F E5           692+     PUSH     H
0430 E5           693+     PUSH     H
0431 E5           694+     PUSH     H
0432 E5           695+     PUSH     H
0433 E5           696+     PUSH     H
0434 E5           697+     PUSH     H
0435 E5           698+     PUSH     H
0436 E5           699+     PUSH     H
0437 E5           700+     PUSH     H
0438 E5           701+     PUSH     H
0439 E5           702+     PUSH     H
043A E5           703+     PUSH     H
043B E5           704+     PUSH     H
043C E5           705+     PUSH     H
043D E5           706+     PUSH     H
043E E5           707+     PUSH     H
043F E5           708+     PUSH     H
0440 E5           709+     PUSH     H
0441 E5           710+     PUSH     H
0442 E5           711+     PUSH     H
0443 E5           712+     PUSH     H
0444 E5           713+     PUSH     H
0445 E5           714+     PUSH     H
0446 E5           715+     PUSH     H
0447 E5           716+     PUSH     H
0448 E5           717+     PUSH     H
0449 E5           718+     PUSH     H
044A E5           719+     PUSH     H
044B E5           720+     PUSH     H
044C E5           721+     PUSH     H
044D E5           722+     PUSH     H
044E E5           723+     PUSH     H
044F EB           724      XCHG    ;PUT STACK IN HL
0450 F9           725      SPHL   ;PUT IT BACK IN SP
0451 FB           726      EI        ;ENABLE INTERRUPTS
0452 C9           727      RET        ;GO BACK
                728      ;
                729      ; IF CURSOR IS ON THE BOTTOM OF THE SCREEN THIS ROUTINE
                730      ; IS USED TO IMPLEMENT THE LINE FEED
                731      ;
0453 2AE30F      732      ONBOT:  LHL     TOPAD      ;GET TOP ADDRESS
0456 22E50F      733      SHLD     LOC80      ;SAVE IT IN LOC80
0459 115000      734      LXI     D, LNTH    ;LINE LENGTH
045C 19           735      DAD     D          ;ADD HL + DE
045D 01D00F      736      LXI     B, LAST    ;GET BOTTOM LINE
0460 7C           737      MOV     A, H       ;GET H
0461 B8           738      CMP     B          ;SAME AS B
0462 C26D04      739      JNZ     ARND      ;LEAVE IF NOT SAME
0465 7D           740      MOV     A, L       ;GET L
0466 B9           741      CMP     C          ;SAME AS C
0467 C26D04      742      JNZ     ARND      ;LEAVE IF NOT SAME
046A 210008      743      LXI     H, TPDIS  ;LOAD HL WITH TOP OF DISPLAY
046D 22E30F      744      ARND:  SHLD     TOPAD      ;SAVE NEW TOP ADDRESS

```

# APPLICATIONS

```

0470 CD1504      745      CALL      CLLINE      ;CLEAR LINE
0473 CDB803      746      CALL      LDCUR       ;LOAD THE CURSOR
0476 C9           747      RET
0477             748      ;
0477             749      ;THIS ROUTINE PUTS A CHARACTER ON THE SCREEN AND
0477             750      ;INCREMENTS THE X CURSOR POSITION. A LINE FEED IS
0477             751      ;INSERTED IF THE INCREMENTED CURSOR EQUALS 81D
0477             752      ;
0477 CDA504      753      CHRPUT: CALL      CALCU       ;CALCULATE SCREEN POSITION
047A 7E           754      MOV      A,M         ;GET FIRST CHARACTER
047B FEF0        755      CPI      0F0H       ;IS IT A CLEAR LINE
047D 22E50F      756      SHLD   LOC00       ;SAVE LINE TO CLEAR
0480 CC1504      757      CZ       CLLINE     ;CLEAR LINE
0483 2AE50F      758      LHL    LOC00       ;GET LINE
0486 CDCD04      759      CALL   ADX        ;ADD CURSOR X
0489 3AE70F      760      LDA    USC'R      ;GET CHARACTER
048C 77           761      MOV    M,A        ;PUT IT ON SCREEN
048D 3AE20F      762      LDA    CURSX     ;GET CURSOR X
0490 3C           763      INR   A          ;INCREMENT CURSOR X
0491 FE50        764      CPI    LN'GH     ;HAS IT GONE TOO FAR?
0493 C29C04      765      JNZ   OK1        ;IF NOT GOOD
0496 CDFC03      766      CALL   LNF'D1    ;DO A LINE FEED
0499 C3AD03      767      JMP   CGRT       ;DO A CR
049C 32E20F      768      STA   CURSX     ;SAVE CURSOR
049F CDB803      769      CALL   LDCUR     ;LOAD THE CURSOR
04A2 C30F01      770      JMP   SETUP     ;LEAVE
04A2             771      ;
04A2             772      ;THIS ROUTINE TAKES THE TOP ADDRESS AND THE Y CURSOR
04A2             773      ;LOCATION AND CALCULATES THE ADDRESS OF THE LINE
04A2             774      ;THAT THE CURSOR IS ON. THE RESULT IS RETURNED IN H
04A2             775      ;AND L AND ALL REGISTERS ARE USED.
04A2             776      ;
04A5 21D504      777      CALCU: LXI   H,LINTAB ;GET LINE TABLE INTO H AND L
04A8 3AE10F      778      LDA    CURSY     ;GET CURSOR INTO A
04AB 07           779      RLC      ;SET UP A FOR LOOKUP TABLE
04AC 0600        780      MVI   B,00H     ;ZERO B
04AE 4F           781      MOV    C,A      ;PUT CURSOR INTO A
04AF 09           782      DAD   B         ;ADD LINE TABLE TO Y CURSOR
04B0 7E           783      MOV    A,M      ;PUT LOW LINE TABLE INTO A
04B1 4F           784      MOV    C,A      ;PUT LOW LINE TABLE INTO C
04B2 23           785      INX   H         ;CHANGE MEMORY POINTER
04B3 7E           786      MOV    A,M      ;PUT HIGH LINE TABLE INTO A
04B4 47           787      MOV    B,A      ;PUT HIGH LINE TABLE INTO B
04B5 2100F8      788      LXI   H,0F800H ;TWO'S COMPLEMENT SCREEN LOCATION
04B8 09           789      DAD   B         ;SUBTRACT OFFSET
04B9 EB           790      XCHG  ;SAVE HL IN DE
04BA 2AE30F      791      LHL    TOPAD     ;GET TOP ADDRESS IN H AND L
04BD 19           792      DAD   D         ;GET DISPLACED ADDRESS
04BE EB           793      XCHG  ;SAVE IT IN D
04BF 2130F0      794      LXI   H,0F030H ;TWO'S COMPLEMENT SCREEN LOCATION
04C2 19           795      DAD   D         ;SEE IF WE ARE OFF THE SCREEN
04C3 DAC804      796      JC     FIX      ;IF WE ARE FIX IT
04C6 EB           797      XCHG  ;GET DISPLACED ADDRESS BACK
04C7 C9           798      RET      ;GO BACK
04C8 2130F8      799      FIX:  LXI   H,0F830H ;SCREEN BOUNDRY
04CB 19           800      DAD   D         ;ADJUST SCREEN
04CC C9           801      RET      ;GO BACK
04CC             802      ;
04CC             803      ;THIS ROUTINE ADDS THE X CURSOR LOCATION TO THE ADDRESS
04CC             804      ;THAT IS IN THE H AND L REGISTERS AND RETURNS THE RESULT
04CC             805      ;IN H AND L
04CC             806      ;
04CD 3AE20F      807      ADX:  LDA    CURSX     ;GET CURSOR
04D0 0600        808      MVI   B,00H     ;ZERO B
04D2 4F           809      MOV    C,A      ;PUT CURSOR X IN C
04D3 09           810      DAD   B         ;ADD CURSOR X TO H AND L
04D4 C9           811      RET      ;LEAVE
04D4             812      ;
04D4             813      ;THIS TABLE CONTAINS THE OFFSET ADDRESSES FOR EACH
04D4             814      ;OF THE 25 DISPLAYED LINES.
04D4             815      ;
0000             816      LINTAB: LNNUM SET 0
04D5 0008        821+   DW    TPDIS+(LN'GH*LNNUM)
0001             822+   LNNUM SET (LNNUM+1)
04D7 5008        823+   DW    TPDIS+(LN'GH*LNNUM)
0002             824+   LNNUM SET (LNNUM+1)
04D9 A008        825+   DW    TPDIS+(LN'GH*LNNUM)
0003             826+   LNNUM SET (LNNUM+1)
04DB F008        827+   DW    TPDIS+(LN'GH*LNNUM)
0004             828+   LNNUM SET (LNNUM+1)
04DD 4009        829+   DW    TPDIS+(LN'GH*LNNUM)
0005             830+   LNNUM SET (LNNUM+1)
04DF 9009        831+   DW    TPDIS+(LN'GH*LNNUM)

```

# APPLICATIONS

0006		832+	LINNUM SET (LINNUM+1)	
04E1	E009	833+	DW TPDIS+(LNGTH*LINNUM)	
0007		834+	LINNUM SET (LINNUM+1)	
04E3	300A	835+	DW TPDIS+(LNGTH*LINNUM)	
0008		836+	LINNUM SET (LINNUM+1)	
04E5	800A	837+	DW TPDIS+(LNGTH*LINNUM)	
0009		838+	LINNUM SET (LINNUM+1)	
04E7	D00A	839+	DW TPDIS+(LNGTH*LINNUM)	
000A		840+	LINNUM SET (LINNUM+1)	
04E9	200B	841+	DW TPDIS+(LNGTH*LINNUM)	
000B		842+	LINNUM SET (LINNUM+1)	
04EB	700B	843+	DW TPDIS+(LNGTH*LINNUM)	
000C		844+	LINNUM SET (LINNUM+1)	
04ED	C00B	845+	DW TPDIS+(LNGTH*LINNUM)	
000D		846+	LINNUM SET (LINNUM+1)	
04EF	100C	847+	DW TPDIS+(LNGTH*LINNUM)	
000E		848+	LINNUM SET (LINNUM+1)	
04F1	600C	849+	DW TPDIS+(LNGTH*LINNUM)	
000F		850+	LINNUM SET (LINNUM+1)	
04F3	800C	851+	DW TPDIS+(LNGTH*LINNUM)	
0010		852+	LINNUM SET (LINNUM+1)	
04F5	000D	853+	DW TPDIS+(LNGTH*LINNUM)	
0011		854+	LINNUM SET (LINNUM+1)	
04F7	500D	855+	DW TPDIS+(LNGTH*LINNUM)	
0012		856+	LINNUM SET (LINNUM+1)	
04F9	A00D	857+	DW TPDIS+(LNGTH*LINNUM)	
0013		858+	LINNUM SET (LINNUM+1)	
04FB	F00D	859+	DW TPDIS+(LNGTH*LINNUM)	
0014		860+	LINNUM SET (LINNUM+1)	
04FD	400E	861+	DW TPDIS+(LNGTH*LINNUM)	
0015		862+	LINNUM SET (LINNUM+1)	
04FF	900E	863+	DW TPDIS+(LNGTH*LINNUM)	
0016		864+	LINNUM SET (LINNUM+1)	
0501	E00E	865+	DW TPDIS+(LNGTH*LINNUM)	
0017		866+	LINNUM SET (LINNUM+1)	
0503	300F	867+	DW TPDIS+(LNGTH*LINNUM)	
0018		868+	LINNUM SET (LINNUM+1)	
0505	800F	869+	DW TPDIS+(LNGTH*LINNUM)	
0019		870+	LINNUM SET (LINNUM+1)	
		871	;	
		872	;KEYBOARD LOOKUP TABLE	
		873	;THIS TABLE CONTAINS ALL THE ASCII CHARACTERS	
		874	;THAT ARE TRANSMITTED BY THE TERMINAL	
		875	;THE CHARACTERS ARE ORGANIZED SO THAT BITS 0,1 AND 2	
		876	;ARE THE SCAN LINES, BITS 3,4 AND 5 ARE THE RETURN LINES	
		877	;BIT 6 IS SHIFT AND BIT 7 IS CONTROL	
		878	;	
0507	38	879	KYLKUP: DB 38H,39H ;8 AND 9	
0508	39			
0509	30	880	DB 30H,2DH ;0 AND -	
050A	2D			
050B	3D	881	DB 3DH,5CH ;= AND \	
050C	5C			
050D	08	882	DB 08H,00H ;BS AND BREAK	
050E	00			
050F	75	883	DB 75H,69H ;LOWER CASE U AND I	
0510	69			
0511	6F	884	DB 6FH,70H ;LOWER CASE O AND P	
0512	70			
0513	5B	885	DB 5BH,5CH ;[ AND \	
0514	5C			
0515	0A	886	DB 0AH,7FH ;LF AND DELETE	
0516	7F			
0517	6A	887	DB 6AH,6BH ;LOWER CASE J AND K	
0518	6B			
0519	6C	888	DB 6CH,3BH ;LOWER CASE L AND ;	
051A	3B			
051B	27	889	DB 27H,00H ;' AND NOTHING	
051C	00			
051D	0D	890	DB 0DH,37H ;CR AND 7	
051E	37			
051F	6D	891	DB 6DH,2CH ;LOWER CASE M AND COMMA	
0520	2C			
0521	2E	892	DB 2EH,2FH ;PERIOD AND SLASH	
0522	2F			
0523	00	893	DB 00H,00H ;BLANK AND NOTHING	
0524	00			
0525	00	894	DB 00H,00H ;NOTHING AND NOTHING	
0526	00			
0527	00	895	DB 00H,61H ;NOTHING AND LOWER CASE A	
0528	61			
0529	7A	896	DB 7AH,78H ;LOWER CASE Z AND X	
052A	78			
052B	63	897	DB 63H,76H ;LOWER CASE C AND V	
052C	76			
052D	62	898	DB 62H,6EH ;LOWER CASE B AND N	
052E	6E			



# APPLICATIONS

052F 79	899	DB	79H,00H	;LOWER CASE Y AND NOTHING
0530 00				
0531 00	900	DB	00H,20H	;NOTHING AND SPACE
0532 20				
0533 64	901	DB	64H,66H	;LOWER CASE D AND F
0534 66				
0535 67	902	DB	67H,68H	;LOWER CASE G AND H
0536 68				
0537 00	903	DB	00H,71H	;TAB AND LOWER CASE Q
0538 71				
0539 77	904	DB	77H,73H	;LOWER CASE W AND S
053A 73				
053B 65	905	DB	65H,72H	;LOWER CASE E AND R
053C 72				
053D 74	906	DB	74H,00H	;LOWER CASE T AND NOTHING
053E 00				
053F 1B	907	DB	1BH,31H	;ESCAPE AND I
0540 31				
0541 32	908	DB	32H,33H	; 2 AND 3
0542 33				
0543 34	909	DB	34H,35H	; 4 AND 5
0544 35				
0545 36	910	DB	36H,00H	; 6 AND NOTHING
0546 00				
0547 2A	911	DB	2AH,28H	; * AND )
0548 28				
0549 29	912	DB	29H,5FH	; ( AND -
054A 5F				
054B 2B	913	DB	2BH,00H	; + AND NOTHING
054C 00				
054D 08	914	DB	08H,00H	; BS AND BREAK
054E 00				
054F 55	915	DB	55H,49H	; U AND I
0550 49				
0551 4F	916	DB	4FH,50H	; O AND P
0552 50				
0553 5D	917	DB	5DH,00H	;   AND NO CHARACTER
0554 00				
0555 0A	918	DB	0AH,7FH	; LF AND DELETE
0556 7F				
0557 4A	919	DB	4AH,4BH	; J AND K
0558 4B				
0559 4C	920	DB	4CH,3AH	; L AND :
055A 3A				
055B 22	921	DB	22H,00H	; " AND NO CHARACTER
055C 00				
055D 0D	922	DB	0DH,26H	; CR AND &
055E 26				
055F 4D	923	DB	4DH,3CH	; M AND <
0560 3C				
0561 3E	924	DB	3EH,3FH	; > AND ?
0562 3F				
0563 00	925	DB	00H,00H	; BLANK AND NOTHING
0564 00				
0565 00	926	DB	00H,00H	; NOTHING AND NOTHING
0566 00				
0567 00	927	DB	00H,41H	; NOTHING AND A
0568 41				
0569 5A	928	DB	5AH,58H	; Z AND X
056A 58				
056B 43	929	DB	43H,56H	; C AND V
056C 56				
056D 42	930	DB	42H,4EH	; B AND N
056E 4E				
056F 59	931	DB	59H,00H	; Y AND NOTHING
0570 00				
0571 00	932	DB	00H,20H	; NO CHARACTER AND SPACE
0572 20				
0573 44	933	DB	44H,46H	; D AND F
0574 46				
0575 47	934	DB	47H,48H	; G AND H
0576 48				
0577 00	935	DB	00H,51H	; TAB AND Q
0578 51				
0579 57	936	DB	57H,53H	; W AND S
057A 53				
057B 45	937	DB	45H,52H	; E AND R
057C 52				
057D 54	938	DB	54H,00H	; T AND NO CONNECTION
057E 00				
057F 1B	939	DB	1BH,21H	; ESCAPE AND !
0580 21				
0581 40	940	DB	40H,23H	; @ AND #
0582 23				
0583 24	941	DB	24H,25H	; \$ AND %
0584 25				
0585 5E	942	DB	5EH,00H	; ^ AND NO CONNECTION

# APPLICATIONS

0586 00				
	943	;		
	944	;	THIS IS WHERE THE CONTROL CHARACTERS ARE LOOKED UP	
	945	;		
0587 00	946	DB	00H,00H	;NOTHING
0588 00				
0589 00	947	DB	00H,00H	;NOTHING
058A 00				
058B 00	948	DB	00H,00H	;NOTHING
058C 00				
058D 00	949	DB	00H,00H	;NOTHING
058E 00				
058F 15	950	DB	15H,09H	;CONTROL U AND I
0590 09				
0591 0F	951	DB	0FH,10H	;CONTROL O AND P
0592 10				
0593 0B	952	DB	0BH,0CH	;CONTROL [ AND \
0594 0C				
0595 0A	953	DB	0AH,7FH	;LF AND DELETE
0596 7F				
0597 0A	954	DB	0AH,0BH	;CONTROL J AND K
0598 0B				
0599 0C	955	DB	0CH,00H	;CONTROL L AND NOTHING
059A 00				
059B 00	956	DB	00H,00H	;NOTHING
059C 00				
059D 0D	957	DB	0DH,00H	;CR AND NOTHING
059E 00				
059F 0D	958	DB	0DH,00H	;CONTROL M AND COMMA
05A0 00				
05A1 00	959	DB	00H,00H	;NOTHING
05A2 00				
05A3 00	960	DB	00H,00H	;NOTHING
05A4 00				
05A5 00	961	DB	00H,00H	;NOTHING AND NOTHING
05A6 00				
05A7 1A	962	DB	1AH,18H	;CONTROL Z AND X
05A8 18				
05A9 03	963	DB	03H,16H	;CONTROL C AND V
05AA 16				
05AB 02	964	DB	02H,0EH	;CONTROL B AND N
05AC 0E				
05AD 19	965	DB	19H,00H	;CONTROL Y AND NOTHING
05AE 00				
05AF 00	966	DB	00H,20H	;NOTHING AND SPACE
05B0 20				
05B1 04	967	DB	04H,06H	;CONTROL D AND F
05B2 06				
05B3 07	968	DB	07H,08H	;CONTROL G AND H
05B4 08				
05B5 00	969	DB	00H,11H	;NOTHING AND CONTROL Q
05B6 11				
05B7 17	970	DB	17H,13H	;CONTROL W AND S
05B8 13				
05B9 06	971	DB	06H,12H	;CONTROL E AND R
05BA 12				
05BB 14	972	DB	14H,00H	;CONTROL W AND NOTHING
05BC 00				
05BD 1B	973	DB	1BH,1DH	;ESCAPE AND HOME(CREDIT)
05BE 1D				
05BF 1E	974	DB	1EH,1CH	;CURSOR UP AND DOWN(CREDIT)
05C0 1C				
05C1 14	975	DB	14H,1FH	;CURSOR RIGHT AND LEFT(CREDIT)
05C2 1F				
05C3 00	976	DB	00H,00H	;NOTHING
05C4 00				
	977	;		
	978	;	LOOK UP TABLE FOR 8253 BAUD RATE GENERATOR	
	979	;		
	980	BDLK: DB	00H,05H,69H,03H	;75 AND 110 BAUD
05C5 00				
05C6 05				
05C7 69				
05C8 03				
05C9 80	981	DB	80H,02H,40H,01H	;150 AND 300 BAUD
05CA 02				
05CB 40				
05CC 01				
05CD A0	982	DB	0A0H,00H	;600 BAUD
05CE 00				
05CF 50	983	DB	50H,00H	;1200 BAUD
05D0 00				
05D1 28	984	DB	28H,00H	;2400 BAUD
05D2 00				
05D3 14	985	DB	14H,00H	;4800 BAUD
05D4 00				
05D5 0A	986	DB	0AH,00H	;9600 BAUD
05D6 00				

# APPLICATIONS

```

987
988 ;DATA AREA
989 ;
0FE1 990 ;ORG 0FE1H
0001 991 CURSY: DS 1
0001 992 CURSX: DS 1
0002 993 TOPAD: DS 2
0002 994 LOC80: DS 2
0001 995 USCHR: DS 1
0002 996 CURAD: DS 2
0001 997 KEYDWN: DS 1
0001 998 KBCHR: DS 1
0001 999 BAUD: DS 1
0001 1000 KEYOK: DS 1
0001 1001 ESCP: DS 1
0001 1002 SHCON: DS 1
0001 1003 RETLIN: DS 1
0001 1004 SCNLIN: DS 1
1005 END

```

## PUBLIC SYMBOLS

## EXTERNAL SYMBOLS

### USER SYMBOLS

ADX	A 04CD	ARND	A 046D	BAUD	A 0FEC	BDLK	A 05C5	BTDIS	A 0F80	BYPASS	A 008F
CAPLOC	A 022E	CGRT	A 03AD	CHREC	A 024E	CHRPUR	A 0477	CLEAR	A 02CF	CLLINE	A 0415
CLRLIN	A 0327	CLRST	A 02D5	CLSCR	A 03E4	CNT0	A 5000	CNT1	A 5001	CNT2	A 5002
CNIM	A 5003	CNWD55	A 1803	CONCL	A 02FD	CRTM	A 1000	CRTS	A 1001	CURAD	A 0FE8
CURSX	A 0FE2	CURSY	A 0FE1	DOWN	A 02AE	ESCP	A 0FEE	ESKAP	A 03A5	ESSQ	A 027B
FMFD	A 03CA	FRAME	A 0167	GD18	A 0359	HOME	A 0397	IN75	A 00F9	INT75	A 1401
KEYDWN	A 0FEA	KEYINP	A 0121	KEYOK	A 0FED	KEYS	A 0131	KPTK	A 0084	KYCHNG	A 01BA
KYLKUP	A 0507	LAST	A 0FD0	LDCUR	A 03B8	LEFT	A 036E	LINNUM	A 0019	LINTAB	A 04D5
LNFD	A 03F6	LNFD1	A 03FC	LNPTH	A 0050	LOADX	A 03EF	LOC80	A 0FE5	LOOPF	A 00A7
LPKBD	A 0098	NOVER	A 038D	NTOVER	A 0364	OK1	A 049C	OK7	A 015C	ONEBOT	A 0453
POPDAT	A 0034	PORTA	A 1800	PORTB	A 1801	PORTC	A 1802	RDKB	A 018F	RETLIN	A 0FE0
RXRDY	A 0113	SAVKEY	A 01AF	SCNLIN	A 0FF1	SCR	A 0211	SETUP	A 010F	SHCON	A 0FEF
STBAUD	A 00DC	STKEY	A 0223	STPTR	A 0FE0	TOPAD	A 0FE3	TPDIS	A 0800	TRANS	A 014B
UP1	A 01E9	UPCUR	A 0333	USCHR	A 0FE7	USTD	A A000	USTF	A A001		

ASSEMBLY COMPLETE, NO ERRORS

# Appendix

CONTENTS

1. Introduction	1
2. Methodology	5
3. Results	10
4. Discussion	15
5. Conclusion	20
6. Appendix A	25
7. Appendix B	30
8. Appendix C	35
9. Appendix D	40
10. Appendix E	45
11. Appendix F	50
12. Appendix G	55
13. Appendix H	60
14. Appendix I	65
15. Appendix J	70
16. Appendix K	75
17. Appendix L	80
18. Appendix M	85
19. Appendix N	90
20. Appendix O	95
21. Appendix P	100
22. Appendix Q	105
23. Appendix R	110
24. Appendix S	115
25. Appendix T	120
26. Appendix U	125
27. Appendix V	130
28. Appendix W	135
29. Appendix X	140
30. Appendix Y	145
31. Appendix Z	150

# INTEL PERIPHERAL COMPONENTS

## Data Communications

	Description
8251A	Programmable Communication Interface
8256	MUART—Multifunctional Asynchronous Receiver/Transmitter
8273	Programmable HDLC/SDLC Protocol Controller
8274	Dual Channel Multiprotocol Controller
Ethernet	Local Area Network Communications
8291	GPIO (IEEE 488) Talker/Listener
8292	GPIO Controller
8293	GPIO Transceiver

## Magnetics

7220	Bubble Memory Controller
7230	Current Pulse Generator for Bubble Memories
7242	Dual Formatter/Sense Amp for Bubble Memories
7250	Coil Pre-Driver for Bubble Memories
7254	Quad VMOS Drive Transistors for Bubble Memories

## Slave Processors (Universal Peripheral Interface)

8041A	Universal Peripheral Interface 1K ROM
8042	Universal Peripheral Interface 2K ROM
RUPI	Remote Universal Peripheral Interface 4K ROM
8741A	Universal Peripheral Interface 1K EPROM
8742	Universal Peripheral Interface 2K EPROM

## Special Function Slave Processors

8231A	Arithmetic Processing Unit
8232	Floating Point Processor
8243	I/O Expander for 8041A
8278	Keyboard Controller
8294	Data Encryption Unit
8295	Dot Matrix Printer Controller

## DRAM Memory Controllers

8202A	4K/16K Dynamic RAM Controller
8203	16K/64K Dynamic RAM Controller
8206	Error Correction Unit

## Timer Counters/Parallel I/O/ Keyboard Controllers

8253	Programmable Interval Timer
8254	High Performance Programmable Interval Timer

## Description

8255	Programmable Peripheral Interface (see also 8155, MCS 85 support; 8256 Data Com. for parallel I/O)
8279	Keyboard/Display Interface (see also 8278—Slave Processors)

## Math Processors

8231A	Arithmetic Processing Unit
8232	Floating Point Processor

## Floppy Disk Controllers

8271	Programmable Floppy Disk Controller
8272	Single/Double Density FDC

## Display Controllers

8275	Programmable CRT Controller
8276	Small System CRT Controller
GDC	Graphics Display Controller (see also 8279 in Timer Counters/Parallel I/O/Keyboard Controllers)

## DMA Controllers/Interrupt Controllers

8237	High Performance Programmable DMA Controller
8257	Programmable DMA Controller
8259A	Programmable Interrupt Controller (see also 8256 Data Com. section for interrupt controllers)

## MCS 80 Bipolar Support

8216/8226	4-Bit Parallel Bidirectional Bus Driver
8218	Bus Controller
8224	Clock Generator
8228/8238	System Controller and Bus Driver

## MCS 85 Bipolar Support

8212	8-Bit Input/Output Port
8219	Bus Controller

## iAPX 88, 86 Bipolar Support

8282/8283	Octal Latches
8284A	Clock Generator
8286/8287	Octal Bus Transceivers
8288	Bus Controller
8289	Bus Arbiter

## Muxed, Memory/I/O Components for MCS 85, iAPX 88

8155/8156	2048-Bit Static MOS RAM with I/O Ports and Timer
8355/8755	16,384-Bit ROM with I/O



## U.S. AND CANADIAN SALES OFFICES

3085 Bowers Avenue  
Santa Clara, California 95051  
Tel: (408) 987-8080  
TWX: 910-338-0026  
TELEX: 34-6372

### ALABAMA

Intel Corp.  
303 Williams Avenue, S.W.  
Suite 1422  
Huntsville 35801  
Tel: (205) 533-9353

### ARIZONA

Intel Corp.  
10210 N. 25th Avenue, Suite 11  
Phoenix 85021  
Tel: (602) 969-4980

### CALIFORNIA

Intel Corp.  
7670 Opportunity Rd.  
Suite 135  
San Diego 92111  
Tel: (714) 268-3563

Intel Corp.\*  
2000 East 4th Street  
Suite 100  
Santa Ana 92705  
Tel: (714) 835-9842  
TWX: 910-595-1114

Intel Corp.\*  
5530 Corbin Ave.  
Suite 120  
Tarzana 91356  
Tel: (213) 708-0333  
TWX: 910-495-2045

Intel Corp.\*  
3375 Scott Blvd.  
Santa Clara 95051  
Tel: (408) 987-8086  
TWX: 910-338-9279  
910-338-0255

Earle Associates, Inc.  
4617 Ruffner Street  
Suite 202  
San Diego 92111  
Tel: (714) 278-5441

Mac-I  
2576 Shattuck Ave.  
Suite 4B  
Berkeley, CA 94704

Mac-I  
558 Valley Way  
Calaveras Business Park  
Milpitas 95035  
Tel: (408) 946-8885

Mac-I  
P.O. Box 8763  
Fountain Valley 92708  
Tel: (714) 839-3341

Mac-I  
25 Village Parkway  
Santa Monica 90409  
Tel: (213) 452-7611

Mac-I  
20121 Ventura Blvd., Suite 240E  
Woodland Hills 91364  
Tel: (213) 347-5900

### COLORADO

Intel Corp.\*  
850 S. Cherry Street  
Suite 720  
Denver 80222  
Tel: (303) 321-8086  
TWX: 910-931-2289

### CONNECTICUT

Intel Corp.  
36 Padanaram Road  
Danbury 06810  
Tel: (203) 792-8366  
TWX: 710-456-1199

EMC Corp.  
48 Purnell Place  
Manchester 06040  
Tel: (203) 646-8085

### FLORIDA

Intel Corp.  
1500 N.W. 82nd Street, Suite 104  
Ft. Lauderdale 33309  
Tel: (305) 771-0600  
TWX: 510-956-9407

Intel Corp.  
500 N. Maitland, Suite 205  
Maitland 32751  
Tel: (305) 828-2393  
TWX: 810-853-9219

### GEORGIA

Intel Corp.  
3300 Holcomb Bridge Rd.  
Norcross 30092  
Tel: (404) 449-0541

### ILLINOIS

Intel Corp.\*  
2550 Goff Road, Suite 815  
Rolling Meadows 80008  
Tel: (312) 981-7200  
TWX: 910-651-5881

### INDIANA

Intel Corp.  
9101 Wesleyan Road  
Suite 204  
Indianapolis 46268  
Tel: (317) 875-0623

### IOWA

Intel Corp.  
St. Andrews Building  
1930 St. Andrews Drive N.E.  
Cedar Rapids 52402  
Tel: (319) 393-5510

### KANSAS

Intel Corp.  
9393 W. 110th St., Ste. 265  
Overland Park 66210  
Tel: (913) 642-8080

### MARYLAND

Intel Corp.\*  
7257 Parkway Drive  
Hanover 21076  
Tel: (301) 798-7500  
TWX: 710-862-1944

### MASSACHUSETTS

Intel Corp.\*  
27 Industrial Ave.  
Chelmsford 01824  
Tel: (617) 256-1800  
TWX: 710-343-6333

EMC Corp.  
381 Elliot Street  
Newton 02164  
Tel: (617) 244-4740  
TWX: 922531

### MICHIGAN

Intel Corp.\*  
26500 Northwestern Hwy.  
Suite 401  
Southfield 48075  
Tel: (313) 353-0920  
TWX: 810-244-4915

### MINNESOTA

Intel Corp.  
7401 Metro Blvd.  
Suite 355  
Edina 55435  
Tel: (612) 835-6722  
TWX: 910-576-2667

### MISSOURI

Intel Corp.  
502 Earth City Plaza  
Suite 121  
Earth City 63045  
Tel: (314) 291-1990

### NEW JERSEY

Intel Corp.\*  
Raritan Plaza  
2nd Floor  
Raritan Center  
Edison 08837  
Tel: (201) 225-3000  
TWX: 710-480-6238  
M. T. I.  
383 Route 46 West  
Fairfield, NJ 07006

### NEW MEXICO

BFA Corporation  
P.O. Box 1237  
Las Cruces 88001  
Tel: (505) 523-0801  
TWX: 910-983-0543

BFA Corporation  
3705 Westerfield, N.E.  
Albuquerque 87111  
Tel: (505) 292-1212  
TWX: 910-989-1157

### NEW YORK

Intel Corp.\*  
300 Motor Pkwy.  
Hauppauge 11787  
Tel: (516) 231-3300  
TWX: 510-227-6236

Intel Corp.  
80 Washington St.  
Poughkeepsie 12601  
Tel: (914) 473-2303  
TWX: 510-248-0080

Intel Corp.\*  
2255 Lyell Avenue  
Lower Floor East Suite  
Rochester 14606  
Tel: (716) 254-6120  
TWX: 510-253-7391

T-Squared  
4054 Newcourt Avenue  
Syracuse 13206  
Tel: (315) 463-8592  
TWX: 710-541-0554

T-Squared  
7353 Pittsburgh  
Victor Road  
Victor 14564  
Tel: (716) 924-9101  
TWX: 510-254-8542

### NORTH CAROLINA

Intel Corp.  
2306 W. Meadowview Rd.  
Suite 206  
Greensboro 27407  
Tel: (919) 294-1541

### OHIO

Intel Corp.\*  
6500 Poe Avenue  
Dayton 45414  
Tel: (513) 890-5360  
TWX: 810-450-2526

Intel Corp.\*  
Chagrin-Brainard Bldg., No. 300  
28001 Chagrin Blvd.  
Cleveland 44122  
Tel: (216) 454-2736  
TWX: 810-427-9298

### OREGON

Intel Corp.  
10700 S.W. Beaverton  
Hillside Highway  
Suite 324  
Beaverton 97005  
Tel: (503) 641-8086  
TWX: 910-467-8741

### PENNSYLVANIA

Intel Corp.\*  
510 Pennsylvania Avenue  
Fort Washington 19034  
Tel: (215) 641-1000  
TWX: 510-681-2077

Intel Corp.\*  
201 Penn Center Boulevard  
Suite 301W  
Pittsburgh 15235  
Tel: (412) 853-4970  
Q.E.D. Electronics  
300 N. York Road  
Hatboro 19040  
Tel: (215) 674-9800

### TEXAS

Intel Corp.\*  
2925 L.B.J. Freeway  
Suite 175  
Dallas 75234  
Tel: (214) 241-9521  
TWX: 910-860-5617

Intel Corp.\*  
8420 Richmond Ave.  
Suite 280  
Houston 77057  
Tel: (713) 784-3400  
TWX: 910-881-2490

Industrial Digital Systems Corp.  
5925 Sovereign  
Suite 101  
Houston 77036  
Tel: (713) 988-9421

Intel Corp.  
313 E. Anderson Lane  
Suite 314  
Austin 78752  
Tel: (512) 454-3628

### UTAH

Intel Corp. (temporary)  
3519 Lexington Dr.  
Bountiful, UT 84010  
Tel: (801) 292-2164

### VIRGINIA

Intel Corp.  
1501 Santa Rosa Road  
Suite C-7  
Richmond, VA 23268  
Tel: (804) 282-5668

### WASHINGTON

Intel Corp.  
Suite 114, Bldg. 3  
1603 118th Ave. N.E.  
Bellevue 98005  
Tel: (206) 453-8086  
TWX: 910-443-3002

### WISCONSIN

Intel Corp.  
150 S. Sunnyslope Rd.  
Brookfield 53005  
Tel: (414) 784-9050

### CANADA

Intel Semiconductor Corp.\*  
Suite 233, Bell Mews  
39 Highway 7, Bell's Corners  
Ottawa, Ontario K2H 6R2  
Tel: (613) 829-9714  
TELEX: 053-4115

Intel Semiconductor Corp.  
50 Galaxy Blvd.  
Unit 12  
Rexdale, Ontario  
M9W 4Y5  
Tel: (416) 675-2105  
TELEX: 06983574

Multitek, Inc.\*  
15 Grenfell Crescent  
Ottawa, Ontario K2G 0G3  
Tel: (613) 226-2365  
TELEX: 053-4585

Multitek, Inc.\*  
Toronto  
Tel: 1-800-267-1070  
Multitek, Inc.  
Montreal  
Tel: 1-800-267-1070

\*Field Application Location



## U.S. AND CANADIAN SERVICE OFFICES

3065 Bowers Avenue  
Santa Clara, California 95051  
Tel: (408) 987-8080  
TWX: 910-338-0028  
TELELEX: 34-6372

### CALIFORNIA

Intel Corp.  
1601 Old Bayshore Hwy.  
Suite 345  
Burlingame 94010  
Tel: (415) 692-4762  
TWX: 910-375-3310

Intel Corp.  
2000 E. 4th Street  
Suite 110  
Santa Ana 92705  
Tel: (714) 835-2670  
TWX: 910-595-2475

Intel Corp.  
7670 Opportunity Road  
San Diego 92111  
Tel: (714) 268-3563

Intel Corp.  
5530 N. Corbin Ave.  
Suite 120  
Tarzana 91356  
Tel: (213) 708-0333

### COLORADO

Intel Corp.  
650 South Cherry  
Suite 720  
Denver 80222  
Tel: (303) 321-8086  
TWX: 910-931-2289

### CONNECTICUT

Intel Corp.  
36 Padanaram Rd.  
Danbury, CT 06810  
Tel: (203) 792-8366

### FLORIDA

Intel Corp.  
1500 N.W. 62nd Street  
Suite 104  
Ft. Lauderdale 33309  
Tel: (305) 771-0800  
TWX: 510-956-9407

Intel Corp.  
500 N. Maitland Ave.  
Suite 205  
Maitland, FL 32751  
Tel: (305) 626-2393  
TWX: 810-853-9219

Intel Corp.  
5151 Adanson St.  
Orlando 32804  
Tel: (305) 829-2393

### GEORGIA

Intel Corp.  
3300 Holcomb Bridge Rd. #225  
Norcross, GA 30092  
Tel: (404) 449-0541

### ILLINOIS

Intel Corp.  
2550 Golf Road  
Suite 815  
Rolling Meadows 60008  
Tel: (312) 981-7230  
TWX: 910-253-1825

### KANSAS

Intel Corp.  
9393 W. 110th Street  
Suite 255  
Overland Park 66210  
Tel: (913) 642-8080

### MARYLAND

Intel Corp.  
7257 Parkway Drive  
Hanover 21076  
Tel: (301) 796-7500  
TWX: 710-862-1944

### MASSACHUSETTS

Intel Corp.  
27 Industrial Avenue  
Chelmsford 01824  
Tel: (617) 256-1800  
TWX: 710-343-6333

### MICHIGAN

Intel Corp.  
26500 Northwestern Hwy.  
Suite 401  
Southfield 48075  
Tel: (313) 353-0920  
TWX: 810-244-4915

### MINNESOTA

Intel Corp.  
7401 Metro Blvd.  
Suite 355  
Edina 55435  
Tel: (612) 835-6722  
TWX: 910-576-2867

### MISSOURI

Intel Corp.  
502 Earth City Plaza  
Suite 121  
Earth City 63045  
Tel: (314) 291-1990

### NEW JERSEY

Intel Corp.  
2480 Lemoine Avenue  
1st Floor  
Ft. Lee 07024  
Tel: (201) 947-6267  
TWX: 710-991-8593

### NEW YORK

Intel Corp.  
2255 Lyell Avenue  
Rochester, NY 14806  
Tel: (716) 254-6120

### NORTH CAROLINA

Intel Corp.  
2306 W. Meadowview Rd.  
Suite 206  
Greensboro, NC 27407  
Tel: (919) 294-1541

### OHIO

Intel Corp.  
Chagrin-Brainard Bldg. Suite 300  
28001 Chagrin Blvd.  
Cleveland 44122  
Tel: (216) 464-2736  
TWX: 810-427-9298

Intel Corp.  
6500 Poe Avenue  
Dayton 45414  
Tel: (513) 890-5350  
TWX: 810-450-2528

### OREGON

Intel Corp.  
10700 S.W. Beaverton-Hilldale Hwy.  
Suite 22  
Beaverton 97005  
Tel: (503) 641-8086  
TWX: 910-467-8741

### PENNSYLVANIA

Intel Corp.  
500 Pennsylvania Avenue  
Fort Washington 19034  
Tel: (215) 641-1000  
TWX: 510-681-2077

Intel Corp.  
201 Penn Center Blvd.  
Suite 301 W.  
Pittsburgh, PA 15235  
Tel: (412) 623-4670

### TEXAS

Intel Corp.  
313 E. Anderson Lane  
Suite 314  
Austin 78752  
Tel: (512) 454-8477  
TWX: 910-874-1347

Intel Corp.  
2925 L.B.J. Freeway  
Suite 175  
Dallas 75234  
Tel: (214) 241-2820  
TWX: 910-860-5617

Intel Corp.  
6420 Richmond Avenue  
Suite 280  
Houston 77057  
Tel: (713) 784-1300  
TWX: 910-881-2490

### VIRGINIA

Intel Corp.  
7700 Leesburg Pike  
Suite 412  
Falls Church 22043  
Tel: (703) 734-9707  
TWX: 710-931-0625

### WASHINGTON

Intel Corp.  
1603 116th Ave. N.E.  
Suite 114  
Bellevue 98005  
Tel: (206) 232-7823  
TWX: 910-443-3002

### WISCONSIN

Intel Corp.  
150 S. Sunnyslope Road  
Suite 148  
Brookfield 53005  
Tel: (414) 784-9060

### CANADA

Intel Corp.  
50 Galaxy Blvd.  
Unit 12  
Rexdale, Ontario  
M9W4Y5  
Tel: (416) 675-2105  
Telex: 069-89278

Intel Corp.  
39 Bells Corners  
Ottawa, Ontario  
K2H 8R2  
Tel: (613) 829-9714  
Telex: 053-4115



# INTERNATIONAL SALES AND MARKETING OFFICES

3085 Bowers Avenue  
Santa Clara, California 95051  
Tel: (408) 987-8080  
TWX: 910-338-0026  
TELEX: 34-6372

## INTERNATIONAL DISTRIBUTORS/REPRESENTATIVES

### AUSTRALIA

A.J.F. Systems & Components Pty. Ltd.  
310 Queen Street  
Melbourne  
Victoria 3000  
Tel: 879-702  
TELEX: AA 31261

Warburton Franki  
Corporate Headquarters  
372 Eastern Valley Way  
Chatswood, New South Wales 2067  
Tel: 407-3261  
TELEX: AA 21299

### AUSTRIA

Bacher Elektronische Gerate GmbH  
Rotenmullgasse 26  
A 1120 Vienna  
Tel: (222) 835646  
TELEX: 131532

Rekirsch Elektronik Gerate GmbH  
Lichtensteinstrasse 9716  
A 1090 Vienna  
Tel: (222) 347646  
TELEX: 134759

### BELGIUM

Inelco Belgium S.A.  
Ave. des Croix de Guerre 94  
B1120 Brussels  
Tel: (02) 216 01 60  
TELEX: 25441

### BRAZIL

Icotron S.A.  
0511 Av. Mutinga 3650  
6 Andar  
Piribuba Sao Paulo  
Tel: 261-0211  
TELEX: 1122274/ICOTBR

### CHILE

DIN  
Av. Vic. MacKenna 204  
Casilla 6055  
Santiago  
Tel: 227 564  
TELEX: 3520003

### CHINA

C.M. Technologies  
525 University Avenue  
Suite A-40  
Palo Alto, CA 94301  
Tel: (415) 326-9150  
Schmidt & Co. Ltd.  
Wing On Centre, 28th Floor  
Connaught Road  
Hong Kong  
Tel: 011-852-5-455-644  
TELEX: 74766 SCHMC HX

### COLOMBIA

International Computer Machines  
Carrera 7 No. 72-34  
Apdo. Aereo 19403  
Bogota 1  
Tel: 211-7282  
TELEX: 44495 TOYOCC

### CYPRUS

Cyprus Eltron Electronics  
P.O. Box 5393  
Nicosia  
Tel: 21-27982

### DENMARK

ITT Multi Komponent  
Fabrysparken 31  
DK-2600 Glostrup  
Tel: (02) 45 66 45  
TX: 33355  
Scandinavian Semiconductor  
Supply A/S  
Nannasgade 18  
DK-2200 Copenhagen  
Tel: (01) 63 50 90  
TELEX: 19037

### FINLAND

Oy Fintronic AB  
Melkonkatu 24 A  
SF-00210  
Helsinki 21  
Tel: (0) 692 80 22  
TELEX: 124 224 Ftron SF

### FRANCE

Caldia S.A.\*  
53, Rue Charles Freret  
F-94250 Gentilly  
Tel: (01) 546 13 13  
TELEX: 200 485

### FEUTRIER

Rue des Trois Lacs  
F-42270 St. Priest-en-Jarez  
Tel: 33 (77) 74 67 33  
TELEX: 300 0 21

### Metrologic\*

La Tour d'Asnieres  
1, Avenue Laurent Cely  
92508-Asnieres  
Tel: (1) 791 44 44  
TELEX: 611 448

### Tekelec Airtronic\*

Cite des Bruyeres  
Rue Carle Vernet  
F-92230 Sevres  
Tel: (01) 534 75 35  
TELEX: 204552

### GERMANY

Electronic 2000 Vertriebs GmbH  
Neumarkter Strasse 75  
D-8000 Munich 80  
Tel: (89) 43 40 61  
TELEX: 522561

### Jermyn GmbH

Postfach 1180  
Schulstrasse 48  
D-6277 Camberg  
Tel: (6343) 4231  
TELEX: 484426

Kontron Elektronik GmbH  
Breslauerstrasse 2  
8057 Eching B  
D-8000 Munich  
Tel: (89) 319 011  
TELEX: 522122

Neye Enatechnik GmbH  
Schillerstrasse 14  
D-2085 Quickborn-Hamburg  
Tel: (4108) 6121  
TELEX: 213590

### GREECE

American Technical Enterprises  
P.O. Box 156  
Athens  
Tel: 30-18811271  
TX: 30-18219470

### HONG KONG

Schmidt & Co.  
Wing on Centre, 28th Floor  
Connaught Road  
Hong Kong  
Tel: 5-455-644  
TELEX: 74766 Schmc Hx

### INDIA

Micronic Devices  
104/109C, Nirmal Industrial Estate  
Sion (E)  
Bombay 400022, India  
Tel: 485-170  
TELEX: 011-5947 MDEV IN

### ISRAEL

Electronics Ltd.\*  
11 Rozans Street  
P.O. Box 39300  
Tel Aviv 61390  
Tel: (3) 47 51 51  
TELEX: 33638

### ITALY

Eledra 3S S.P.A.\*  
Viale Elvezia, 18  
I 20154 Milano  
Tel: (2) 34 97 51  
TELEX: 332332

### JAPAN

Asahi Electronics Co. Ltd.  
KMM Bldg. Room 407  
2-14-1 Aaano, Kokura  
Kita-Ku, Kitakyushu City 802  
Tel: (093) 511-6471  
TELEX: AECKY 7126-16

Hamilton-Avnet Electronics Japan Ltd.  
YU and YOU Bldg. 1-4 Horidome-Cho  
Nihonbashi Chuo-Ku, Tokyo 103  
Tel: (03) 662-9911  
TELEX: 2523774

Ryoyo Electric Corp.  
Konwa Bldg.  
1-12-22, Takuji  
Chuo-Ku, Tokyo 104  
Tel: (03) 543-7711

Tokyo Electron Ltd.  
Shin Juku, Nomura Bldg.  
26-2 Nishi-Shin Juku-ichome  
Shin Juku-Ku, Tokyo 160  
Tel: (03) 343-4411  
TELEX: 232-2220 LABEL J

### KOREA

Koram Digital  
Room 909 Woonam Bldg.  
7, 1-KA Bongre-Dong  
Chung-Ku Seoul  
Tel: 238-123  
TELEX: K23542 HANSINT

### MEXICO

Proveedora Electronica, S.A. (Proesa)  
Prof. Moctezuma Ote. 24  
Col. Romero de Terreros  
Apdo. Postal 21-139  
Mexico 21, D.F.  
Tel: 554-8300  
TELEX: 017-72402 SAULME

### NETHERLANDS

Inelco Nether. Comp. Sys. BV  
Turfstekersstraat 63  
P.O. Box 360  
NL Aalsmeer 1430  
Tel: (2977) 28855  
TELEX: 14693

Koning & Hartman  
Koperwerf 30  
P.O. Box 43220  
2544 EM's Gravenhage  
Tel: 31 (70) 210.101  
TELEX: 31528

### NEW ZEALAND

McLean Information Technology Ltd.  
P.O. Box 18-065  
Glenn Innes, Auckland, 6  
Tel: 587-037  
TELEX: NZ2763 KOSFY

### NORWAY

Nordisk Elektronisk (Norge) A/S  
Postoffice Box 122  
Smødsving 4  
1364 Hvalstad  
Tel: (2) 786 210  
TELEX: 16963

### PORTUGAL

Ditram  
Componentes E Electronica LDA  
Av. Miguel Bombarda, 133  
P1000 Lisboa  
Tel: (19) 545 313  
TELEX: 14182 Brieks-P

### SINGAPORE

General Engineers Corporation Pte. Ltd.  
Blok 3, Units 1003-1008, 10th Floor  
P.S.A. Multi-Storey Complex  
Pasir Panjang Road  
Singapore 05 11  
Tel: 271-3163  
TELEX: RS23987 GENERCO

### SOUTH AFRICA

Electronic Building Elements  
P.O. Box 4609  
Hazelwood, Pretoria 0001  
Tel: 011-27-12-46-9221  
TELEX: 30181SA

### SPAIN

Interface S.A.  
Ronda San Pedro 22, 3\*  
Barcelona 10  
Tel: (3) 301 78 51  
TX: 51508

### ITT SESA

Miguel Angel 16-6  
Madrid 10  
Tel: (1) 410.23.54  
TELEX: 27707/27461

### SWEDEN

AB Gosta Backstrom  
Box 12009  
Alstrångeråtan 22  
S-10221 Stockholm 12  
Tel: (8) 541 080  
TELEX: 10135

### Nordisk Elektronik AB

Box 27301  
S-10254 Stockholm  
Tel: (8) 635 040  
TELEX: 10547

### SWITZERLAND

Industrade AG  
Gemsenstrasse 2  
Postcheck 80 - 21190  
CH-8021 Zurich  
Tel: (1) 363 22 30  
TELEX: 66798

### TAIWAN

Taiwan Automation Co.\*  
3d Floor #75, Section 4  
Nanking East Road  
Taipei  
Tel: 771-0940  
TELEX: 11942 TAIATAO

### TURKEY

Turkelek Electronics  
Aparuk Boulevard 169  
Ankara  
Tel: 189483

### UNITED KINGDOM

Conway Microsystems Ltd.  
Market Street  
UK Bracknell, Berkshire  
Tel: 44 (344) 51654  
TELEX: 847201  
M.E.D.L.  
East Lane Road  
North Wembley  
Middlesex HA9 7PP  
Tel: 44 (01) 904-9303/908-4111  
TELEX: 28817

Jermyn Industries (Mogul)  
Vestry Estate  
Sevenoaks, Kent  
Tel: (0732) 601.44  
TELEX: 95142

Rapid Recall, Ltd.  
Rapid House/Denmark St  
High Wycombe  
Bucks, England HP11 2ER  
Tel: 44 494 26 271  
TELEX: 849439

Bytech Ltd.  
Sutton Park Avenue  
Reading, Berkshire RG1 2A  
Tel: (0734) 61 031  
TELEX: 848215

\*Field Application Location





# INTERNATIONAL SALES AND MARKETING OFFICES

3065 Bowers Avenue  
Santa Clara, California 95051  
Tel: (408) 987-8080  
TWX: 910-338-0026  
TELEX: 34-6372

## INTEL® MARKETING OFFICES

### AUSTRALIA

Intel Semiconductor Pty. Ltd.  
Suite 2, Level 15, North Point  
100 Miller Street  
North Sydney, NSW, 2060  
Tel: 450-847  
TELEX: AA 20097

### BELGIUM

Intel Corporation S.A.  
Rue du Moulin a Papier 51  
Boite 1  
B-1160 Brussels  
Tel: (02) 660 30 10  
TELEX: 24814

### DENMARK

Intel Denmark A/S\*  
Lyngbyvej 32F 2nd Floor  
DK-2100 Copenhagen East  
Tel: (01) 18 20 00  
TELEX: 19567

### FINLAND

Intel Finland OY  
Sensharikuja 3  
SF-00400 Helsinki 40  
Tel: (0) 58244 55  
TELEX: 123 332

### FRANCE

Intel Corporation, S.A.R.L.\*  
5 Place de la Balance  
Silic 223  
94528 Rungis Cedex  
Tel: (01) 987 22 21  
TELEX: 270475

### GERMANY

Intel Semiconductor GmbH\*  
Seidstrasse 27  
D-8000 Muenchen 2  
Tel: (89) 53891  
TELEX: 523 177

Intel Semiconductor GmbH  
Mainzer Strasse 75  
D-6200 Wiesbaden 1  
Tel: (6121) 70 08 74  
TELEX: 04186183

Intel Semiconductor GmbH  
Wernerstrasse 87  
P.O. Box 1460  
D-7012 Fellbach  
Tel: (711) 58 00 82  
TELEX: 7254826

Intel Semiconductor GmbH  
Hohenzollern Strasse 5  
3000 Hannover 1  
Tel: (511) 32 70 81  
TELEX: 923625

Intel Semiconductor GmbH  
Vertriebsbuero Dusseldorf  
Ober-Ratherstrasse 2  
4000 Dusseldorf 30  
Tel: (a11) 65 10 54  
TELEX: 8586977

### HONG KONG

Intel Semiconductor Ltd.  
99-105 Des Voeux Rd., Central  
18F, Unit B  
Hong Kong  
Tel: 5-450-847  
TELEX: 93869

### ISRAEL

Intel Semiconductor Ltd.\*  
P.O. Box 1859  
Haifa  
Tel: 4/524 261  
TELEX: 46511

### ITALY

Intel Corporation Italia Spa  
Milanofiori, Palazzo E  
20094 Assago (Milano)  
Tel: (02) 824 00 06  
TELEX: 315183 INTMIL

### JAPAN

Intel Japan K.K.\*  
Flower Hill-Shinmachi East Bldg.  
1-23-9 Shinmachi, Setagaya-ku  
Tokyo 154  
Tel: (03) 426-9261  
TELEX: 781-28426

### NETHERLANDS

Intel Semiconductor Nederland B.V.  
Oranjestraat 1  
3441 Ax Woerden  
Netherlands  
Tel: 31-3480-112-64  
TELEX: 47970

Intel Semiconductor B.V.  
Cometongebouw  
Westblaak 106  
3012 Km Rotterdam  
Tel: (10) 149122  
TELEX: 22283

### NORWAY

Intel Norway A/S  
P.O. Box 92  
Hvamsvien 4  
N-2013  
Skjetten  
Tel: (2) 742 420  
TELEX: 18018

### SWEDEN

Intel Sweden A.B.\*  
Box 20092  
Enighetsvagen 5  
S-16120 Bromma  
Tel: (08) 98 53 85  
TELEX: 12261

### SWITZERLAND

Intel Semiconductor A.G.  
Forchstrasse 95  
CH 8032 Zurich  
Tel: (01) 55 45 02  
TELEX: 557 89 ich ch

### UNITED KINGDOM

Intel Corporation (U.K.) Ltd.\*  
5 Hospital Street  
Nantwich, Cheshire CW5 5RE  
Tel: (0270) 626 560  
TELEX: 36620

Intel Corporation (U.K.) Ltd.  
Dorcan House  
Eldene Drive  
Swindon, Wiltshire SN3 310  
Tel: (0793) 26 101  
TELEX: 444447 INT SWN

\*Field Application Location



## U.S. AND CANADIAN DISTRIBUTORS

3065 Bowers Avenue  
Santa Clara, California 95051  
Tel: (408) 987-8080  
TWX: 910-338-0026  
TELEX: 34-6372

### ALABAMA

Arrow Electronics  
4717 University Dr.  
Suite 102 1/2 D  
Huntsville 35405  
Tel: (205) 830-1103  
†Hamilton / Avnet Electronics  
4812 Commercial Drive N.W.  
Huntsville 35805  
Tel: (205) 837-7210  
TWX: 810-726-2162  
†Pioneer/Huntsville  
1207 Putnam Drive N.W.  
Huntsville 35805  
Tel: (205) 837-9300  
TWX: 810-726-2197

### ARIZONA

†Hamilton / Avnet Electronics  
505 S. Madison Drive  
Tempe, AZ 85281  
Tel: (602) 231-5140  
TWX: 910-950-0077  
†Wyle Distribution Group  
8155 N. 24th Street  
Phoenix 85021  
Tel: (602) 995-9185  
TWX: 910-951-4282

### CALIFORNIA

Arrow Electronics, Inc.  
521 Weddell Dr.  
Sunnyvale 94086  
Tel: (408) 745-6600  
TWX: 910-339-9371  
†Avnet Electronics  
350 McCormick Avenue  
Costa Mesa 92626  
Tel: (714) 754-6051  
TWX: 910-595-1928  
Hamilton / Avnet Electronics  
1175 Bordeaux Dr.  
Sunnyvale 94086  
Tel: (408) 743-3300  
TWX: 910-339-9332  
†Hamilton / Avnet Electronics  
4545 Viewridge Ave  
San Diego 92123  
Tel: (714) 563-1989  
TWX: 910-335-1218  
†Hamilton / Avnet Electronics  
10912 W. Washington Blvd.  
Culver City 90230  
Tel: (213) 558-2193  
TWX: 910-340-6384 or 7073  
†Hamilton Electro Sales  
3170 Putman Street  
Costa Mesa 92626  
Tel: (714) 641-4109  
TWX: 910-595-2638  
†Wyle Distribution Group  
124 Maryland Street  
El Segundo 90245  
Tel: (213) 322-8100  
TWX: 910-348-7140 or 7111  
†Wyle Distribution Group  
9525 Chesapeake Dr.  
San Diego 92123  
Tel: (714) 565-9171  
TWX: 910-335-1590  
†Wyle Distribution Group  
3000 Bowers Avenue  
Santa Clara 95052  
Tel: (408) 727-2500  
TWX: 910-338-0451 or 0296  
†Wyle Distribution Group  
17872 Cowan Avenue  
Irvine 92713  
Tel: (714) 641-1600  
TWX: 910-595-1572

### COLORADO

†Wyle Distribution Group  
451 E 124th Avenue  
Thornton, CO 80241  
Tel: (303) 457-8953  
TWX: 910-936-0770  
†Hamilton / Avnet Electronics  
8765 E. Orchard Road  
Suite 708  
Englewood 80111  
Tel: (303) 740-1017  
TWX: 910-935-0787

### CONNECTICUT

†Arrow Electronics  
12 Beaumont Road  
Wallingford 06512  
Tel: (203) 265-7741  
TWX: 710-476-0182  
†Hamilton / Avnet Electronics  
Commerce Industrial Park  
Commerce Drive  
Danbury 06810  
Tel: (203) 797-2800  
TWX: 710-456-9974  
†Harvey Electronics  
112 Main Street  
Norwalk 06851  
Tel: (203) 853-1515  
TWX: 710-468-3373

### FLORIDA

†Arrow Electronics  
1001 N.W. 62nd Street  
Suite 108  
Ft. Lauderdale 33309  
Tel: (305) 776-7790  
TWX: 510-955-9456  
†Arrow Electronics  
115 Palm Bay Road, N.W.  
Suite 10, Bldg. 200  
Palm Bay 32905  
Tel: (305) 725-1480  
TWX: 510-959-6337  
†Hamilton / Avnet Electronics  
6800 Northwest 20th Ave.  
Ft. Lauderdale 33309  
Tel: (305) 971-2900  
TWX: 510-956-3097  
Hamilton / Avnet Electronics  
3197 Tech. Drive North  
St. Petersburg 33702  
Tel: (813) 576-3930  
TWX: 810-863-0374  
†Pioneer/Orlando  
6220 S. Orange Blossom Trail  
Suite 412  
Orlando 32809  
Tel: (305) 859-3600  
TWX: 810-850-0177

### GEORGIA

Arrow Electronics  
2979 Pacific Drive  
Norcross 30071  
Tel: (404) 449-8252  
TWX: 810-766-0439  
†Hamilton / Avnet Electronics  
5925 D. Peachtree Corners  
Norcross 30092  
Tel: (404) 448-1711  
TWX: 810-766-4515  
Pioneer/Georgia  
5835 B Peachtree Corners E  
Norcross 30092  
Tel: (404) 448-1711  
TWX: 810-766-4515

### ILLINOIS

Arrow Electronics  
492 Lunt Avenue  
P.O. Box 94248  
Schaumburg 60172  
Tel: (312) 893-9420  
TWX: 910-291-3544  
†Hamilton / Avnet Electronics  
3901 No. 25th Avenue  
Schiller Park 60176  
Tel: (312) 678-6310  
TWX: 910-227-0080  
Pioneer/Chicago  
1551 Carmen Drive  
Elk Grove 60007  
Tel: (312) 437-9680  
TWX: 910-222-1834

### INDIANA

Arrow Electronics  
2718 Rand Road  
Indianapolis 46241  
(317) 243-9353  
Tel: 810-341-3119  
†Hamilton / Avnet Electronics  
485 Gladie Drive  
Carmel 46032  
Tel: (317) 844-9333  
TWX: 810-260-3966

### INDIANA (Cont.)

Pioneer/Indiana  
6408 Castlepiece Drive  
Indianapolis 46250  
Tel: (317) 849-7300  
TWX: 810-260-1794  
KANSAS  
†Hamilton / Avnet Electronics  
9219 Quivera Road  
Overland Park 66215  
Tel: (913) 588-8900  
TWX: 910-743-0005  
†Component Specialties, Inc.  
8369 Nieman Road  
Lenexa 66214  
Tel: (913) 492-3555

### MARYLAND

†Hamilton / Avnet Electronics  
6822 Oak Hall Lane  
Columbia, MD 21045  
Tel: (301) 995-3500  
TWX: 710-862-1861  
Mesa  
16021 Industrial Dr.  
Gaithersburg 20760  
Tel: (301) 948-4350  
†Pioneer/Washington  
9100 Gaither Road  
Gaithersburg 20760  
Tel: (301) 948-0710  
TWX: 710-828-0545

### MASSACHUSETTS

†Hamilton / Avnet Electronics  
50 Tower Office Park  
Woburn 01801  
Tel: (617) 935-9700  
TWX: 710-393-0382  
†Arrow Electronics  
Arrow Dr.  
Woburn 01801  
Tel: (617) 933-8130  
TWX: 710-393-8770  
Harvey/Boston  
44 Hartwell Ave.  
Lexington 02173  
Tel: (617) 863-1200  
TWX: 710-326-6617

### MICHIGAN

†Arrow Electronics  
3810 Varsity Drive  
Ann Arbor 48104  
Tel: (313) 971-8220  
TWX: 810-223-6020  
†Pioneer/Michigan  
13485 Stamford  
Livonia 48150  
Tel: (313) 525-1800  
TWX: 810-242-3271  
†Hamilton / Avnet Electronics  
32487 Schoolcraft Road  
Livonia 48150  
Tel: (313) 522-4700  
TWX: 810-242-8775

### MINNESOTA

†Arrow Electronics  
5230 W. 73rd Street  
Edina 55435  
Tel: (612) 830-1800  
TWX: 910-576-3125  
†Industrial Components  
5229 Edina Industrial Blvd.  
Minneapolis 55435  
Tel: (612) 831-2866  
TWX: 910-576-3153  
Hamilton / Avnet Electronics  
10300 Bren Rd. East  
Minnetonka 55343  
Tel: (612) 932-0666  
TWX: (612) 676-2720  
†Hamilton / Avnet Electronics  
7449 Cahill Road  
Edina 55435  
Tel: (612) 941-3801  
TWX: 910-578-2720

### MISSOURI

†Arrow Electronics  
2380 Schuetz  
St. Louis, MO 63141  
Tel: (314) 567-6888  
†Hamilton / Avnet Electronics  
13743 Shoreline Ct.  
Earth City 63045  
Tel: (314) 344-1200  
TWX: 910-762-0684

### NEW HAMPSHIRE

†Arrow Electronics  
1 Perimeter Drive  
Manchester 03103  
Tel: (603) 668-6968  
TWX: 710-220-1684  
NEW JERSEY  
†Arrow Electronics  
Pleasant Valley Avenue  
Moorestown 08057  
Tel: (215) 925-1800  
TWX: 710-897-0829  
†Arrow Electronics  
285 Midland Avenue  
Saddle Brook 07662  
Tel: (201) 797-5800  
TWX: 710-988-2206

†Hamilton / Avnet Electronics  
1 Keystone Ave.  
Bldg. 38  
Cherry Hill 08003  
Tel: (609) 424-0100  
TWX: 710-940-0262

Hamilton / Avnet Electronics  
10 Industrial Road  
Fairfield 07006  
Tel: (201) 575-3390  
TWX: 710-734-4388  
†Harvey Electronics  
45 Route 46  
Pinebrook 07058  
Tel: (201) 227-1262  
TWX: 710-734-4382  
Measurement Technology Sales Corp.  
383 Route 46 W  
Fairfield, NJ 07006  
Tel: (201) 227-5552

### NEW MEXICO

†Alliance Electronics Inc.  
11030 Cochiti S.E.  
Albuquerque 87123  
Tel: (505) 292-3860  
TWX: 910-989-1151  
†Hamilton / Avnet Electronics  
2524 Baylor Drive S.E.  
Albuquerque 87119  
Tel: (505) 765-1500  
TWX: 910-969-0614

### NEW YORK

†Arrow Electronics  
900 Broad Hollow Rd.  
Farmingdale, NY 11735  
Tel: (516) 894-6800  
TWX: 510-224-6494  
†Arrow Electronics  
3000 South Winton Road  
Rochester 14623  
Tel: (716) 275-0300  
TWX: 510-253-4766  
†Arrow Electronics  
7705 Maltage Drive  
Liverpool 13088  
Tel: (315) 652-1000  
TWX: 710-545-0230  
Arrow Electronics  
20 Osar Avenue  
Hauppauge 11787  
Tel: (516) 231-1000  
TWX: 510-227-6623  
†Hamilton / Avnet Electronics  
333 Metro Park  
Rochester 14623  
Tel: (716) 475-9130  
TWX: 510-253-5470  
†Hamilton / Avnet Electronics  
16 Corporate Circle  
E. Syracuse 13057  
Tel: (315) 437-2641  
TWX: 710-541-1560  
†Hamilton / Avnet Electronics  
5 Hub Drive  
Melville, Long Island 11746  
Tel: (516) 454-6000  
TWX: 510-224-6166  
Harvey Electronics  
P.O. Box 1208  
Binghamton 13902  
Tel: (607) 748-8211  
TWX: 510-252-0893



## U.S. AND CANADIAN DISTRIBUTORS

3065 Bowers Avenue  
Santa Clara, California 95051  
Tel: (408) 987-8080  
TWX: 910-338-0026  
TELEX: 34-6372

### NEW YORK (Cont.)

†Harvey Electronics  
60 Crossways Park West  
Woodbury, Long Island 11797  
Tel: (516) 221-8920  
TWX: 510-221-2184

Harvey/Rochester  
840 Fairport Park  
Fairport 14450  
Tel: (716) 381-7070  
TWX: 510-253-7001

Measurement Technology Sales Corp.  
169 Northern Blvd.  
Greatneck 11021  
Tel: (516) 482-3500  
TWX: 510-223-0846

### NORTH CAROLINA

Arrow Electronics  
938 Burke Street  
Winston-Salem 27102  
Tel: (919) 725-8711  
TWX: 510-931-3169

†Hamilton/Avnet Electronics  
2803 Industrial Drive  
Raleigh 27609  
Tel: (919) 829-8030  
TWX: 510-928-1836

Pioneer/Carolina  
106 Industrial Ave.  
Greensboro 27406  
Tel: (919) 273-4441  
TWX: 510-925-1114

### OHIO

Arrow Electronics  
10 Knokkrest Dr.  
Reading, OH 45237  
Tel: (513) 761-5432  
TWX: 810-461-2870

Arrow Electronics  
7620 McEwen Road  
Centerville 45459  
Tel: (513) 435-5563  
TWX: 810-459-1611

Arrow Electronics  
6238 Cochran Rd.  
Solon 44139  
Tel: (216) 248-3990  
TWX: 810-427-9409

†Hamilton/Avnet Electronics  
954 Senate Drive  
Dayton 45459  
Tel: (513) 433-0610  
TWX: 910-450-2531

†Hamilton/Avnet Electronics  
4588 Emery Industrial Parkway  
Warrensville Heights 44128  
Tel: (216) 831-3500  
TWX: 810-427-9452

†Pioneer/Dayton  
4433 Interpoint Blvd.  
Dayton 45424  
Tel: (513) 236-9900  
TWX: 810-459-1622

†Pioneer/Cleveland  
4800 E. 131st Street  
Cleveland 44105  
Tel: (216) 587-3600  
TWX: 810-422-2211

### OKLAHOMA

†Components Specialties, Inc.  
7920 E. 40th Street  
Tulsa 74145  
Tel: (918) 564-2820  
TWX: 910-945-2215

### OREGON

†Almac/Strom Electronics  
6022 S.W. Nimbus, Bldg. 7  
Beaverton 97005  
Tel: (503) 841-9070  
TWX: 910-467-8743

†Hamilton/Avnet Electronics  
6024 S.W. Jean Rd.  
Bldg. C, Suite 10  
Lake Oswego 97034  
Tel: (503) 635-7848  
TWX: 910-455-8179

### PENNSYLVANIA

Arrow Electronics  
650 Seco Rd.  
Monroeville, PA 15146  
Tel: (412) 856-7000

†Arrow Electronics  
650 Seco Rd.  
Monroeville 15146  
Tel: (412) 856-7000

Pioneer/Pittsburgh  
259 Kappa Drive  
Pittsburgh 15238  
Tel: (412) 782-2300  
TWX: 710-795-3122

Pioneer/Delaware Valley  
261 Gibraltar Road  
Horsesham 19044  
Tel: (215) 674-4000  
TWX: 510-665-6778

### TEXAS

Arrow Electronics  
13715 Gama Road  
Dallas 75234  
Tel: (214) 386-7500  
TWX: 910-860-5377

Arrow Electronics, Inc.  
10700 Corporate Drive, Suite 100  
Stafford 77477  
Tel: (713) 491-4100  
TWX: 910-880-4439

Component Specialties, Inc.  
8222 Jamestown Drive  
Suite 115  
Austin 78758  
Tel: (512) 837-8922  
TWX: 910-874-1320

†Component Specialties, Inc.  
10907 Shady Trail, Suite 101  
Dallas 75220  
Tel: (214) 357-6511  
TWX: 910-861-4999

†Component Specialties, Inc.  
8181 Commerce Park Drive, Suite 700  
Houston 77036  
Tel: (713) 771-7237  
TWX: 910-881-2422

Hamilton/Avnet Electronics  
10508A Boyer Blvd.  
Austin 78757  
Tel: (512) 837-8911  
TWX: 910-874-1319

†Hamilton/Avnet Electronics  
2111 W. Walnut Hill Lane  
Irving 75062  
Tel: (214) 659-4100  
TWX: 910-860-5929

†Hamilton/Avnet Electronics  
3939 Ann Arbor Drive  
Houston 77063  
Tel: (713) 780-1771  
TWX: 910-881-5523

### UTAH

†Hamilton/Avnet Electronics  
1585 West 2100 South  
Salt Lake City 84119  
Tel: (801) 972-2800  
TWX: 910-925-4018

### WASHINGTON

†Almac/Strom Electronics  
5811 Sixth Ave. South  
Seattle 98108  
Tel: (206) 763-2300  
TWX: 910-444-2067

†Hamilton/Avnet Electronics  
14212 N.E. 21st Street  
Bellevue 98005  
Tel: (206) 453-5844  
TWX: 910-443-2469

†Wyle Distribution Group  
1760 132nd Avenue N.E.  
Bellevue 98005  
Tel: (206) 453-8300  
TWX: 910-443-2526

### WISCONSIN

†Arrow Electronics  
430 W. Rausson Avenue  
Oskosh 53154  
Tel: (414) 764-6600  
TWX: 910-262-1193

†Hamilton/Avnet Electronics  
2975 Moorland Road  
New Berlin 53151  
Tel: (414) 784-4610  
TWX: 910-262-1182

### CANADA

#### ALBERTA

†L.A. Varah Ltd.  
4742 14th Street N.E.  
Calgary T2D 6L7  
Tel: (403) 230-1235  
TWX: 038-258-97

Zentronics  
9224 27th Avenue  
Edmonton T6N 1B2  
Tel: (403) 463-3014  
Telex: 03742841

Zentronics  
3651 21st N.E.  
Calgary T2E 6T5  
Tel: (403) 230-1422

#### BRITISH COLUMBIA

†L.A. Varah Ltd.  
2077 Alberta Street  
Vancouver V5Y 1C4  
Tel: (604) 873-3211  
TWX: 610-929-1068

Zentronics  
550 Cambie St.  
Vancouver V6B 2N7  
Tel: (604) 688-2533  
TWX: 04-5077-89

#### MANITOBA

L.A. Varah  
1-1832 King Edward Street  
Winnipeg R2R 0N1  
Tel: (204) 633-8190  
TWX: 07-55-365

Zentronics  
590 Berry St.  
Winnipeg R3H 0S1  
Tel: (204) 775-8661

#### NOVA SCOTIA

Zentronics  
30 Simmonds Dr., Unit B  
Dartmouth, B3B 1R3

### ONTARIO

†Hamilton/Avnet Electronics  
6845 Rexwood Road, Units G & H  
Mississauga L4V 1M5  
Tel: (416) 677-4732  
TWX: 610-492-8867

†Hamilton/Avnet Electronics  
1735 Courtwood Crescent  
Ottawa K2C 3J2  
Tel: (613) 228-1700  
TWX: 053-4971

†L.A. Varah, Ltd.  
505 Kenora Avenue  
Hamilton L8E 3P2  
Tel: (416) 561-9311  
TWX: 061-8349

#### Zentronics

141 Catherine Street  
Ottawa K2P 1C3  
Tel: (613) 238-6411  
TWX: 053-3636

#### Zentronics

8 Kilbury Ct.  
Brampton, Ontario  
Toronto, L6T 3T4  
Tel: (416) 451-9600  
Telex: 06-976-78

#### Zentronics

564/10 Weber St., N.  
Waterloo, NAL 5C6  
Tel: (519) 884-5700

#### QUEBEC

†Hamilton/Avnet Electronics  
2670 Sabourin Street  
St. Laurent H4S 1M2  
Tel: (514) 331-6643  
TWX: 610-421-3731

#### Zentronics

5010 Rue Pare Street  
Montreal H4P 1P3  
Tel: (514) 735-5361  
TWX: 05-827-535



*Intel Corporation  
3065 Bowers Avenue  
Santa Clara, CA 95051*

*Intel International  
Rue du Moulin à Papier 51, Boite 1,  
B-1160 Brussels, Belgium*

*Intel Japan K.K.  
Flower Hill-Shinmachi East Bldg.  
1-23-9, Shinmachi, Setagayu-ku  
Tokyo 154, Japan*