EK-SBC02-UG-001

SBC-11/21 PLUS
Single-Board Computer
User's Guide

Prepared by Educational Services
of
Digital Equipment Corporation

First Edition, March 1984

The manuscript for this book was created using a DIGITAL Word Processing System and, via a translation program, was automatically typeset on DIGITAL's DECset Integrated Publishing System. Book production was done by Educational Services Development and Publishing in Marlboro, MA.

CONTENTS

Page

PREFACE

CHAPTER 1 INTRODUCTION

1.1 INTRODUCTION ... 1-1
1.2 SPECIFICATIONS ... 1-3
1.2.1 Physical ... 1-3
1.2.2 Power Requirements ... 1-3
1.2.3 Bus Loading ... 1-3
1.2.4 Environmental ... 1-4
1.3 BACKPLANE PIN IDENTIFICATION ... 1-4
1.4 RELATED DOCUMENTS ... 1-7

CHAPTER 2 INSTALLATION

2.1 INTRODUCTION ... 2-1
2.2 SELECTING OPERATIONAL FEATURES ... 2-1
2.2.1 Battery Backup ... 2-1
2.2.2 Wake-Up Circuit ... 2-8
2.2.3 Starting Address ... 2-8
2.2.4 Interrupts ... 2-8
2.2.5 Parallel I/O ... 2-11
2.2.6 Serial I/O ... 2-12
2.2.7 Memories ... 2-16
2.2.7.1 Memory Maps ... 2-16
2.2.7.2 PROMs/EPROMs/EEPROMs ... 2-16
2.2.7.3 RAMs ... 2-16
2.3 SELECTING BACKPLANES AND OPTIONS ... 2-22
2.4 POWER SUPPLY ... 2-22
2.5 EXTERNAL CABLES ... 2-22
2.5.1 Parallel I/O Interface (J3) ... 2-23
2.5.2 Serial Line Interfaces (J1 and J2) ... 2-25
2.6 VERIFYING OPERATION ... 2-28
2.6.1 Macro-ODT Option ... 2-28
2.6.2 Loopback Connectors ... 2-28
2.6.3 Verification Procedure ... 2-28

CHAPTER 3 OPTIONS

3.1 INTRODUCTION ... 3-1
3.2 SUPPORTED OPTIONS ... 3-1
3.2.1 Hardware Options ... 3-1
3.2.2 Software Options ... 3-5
3.2.2.1 RT-11 Operating System ... 3-5
3.2.2.2 MicroPower/Pascal Operating System ... 3-6

CONTENTS (Cont)

Page

CHAPTER 4 MACRO-ODT

4.1 INTRODUCTION 4-1
4.2 INSTALLATION AND CONFIGURATION 4-1
4.3 ENTRY CONDITIONS 4-1
4.3.1 Macro-ODT Input Sequence 4-1
4.3.2 Macro-ODT Output Sequence 4-2
4.4 MACRO-ODT COMMANDS 4-2
4.4.1 /(ASCII 057) Slash 4-2
4.4.2 <CR> (ASCII 15) Carriage Return 4-5
4.4.3 <LF> (ASCII 12) Line Feed 4-5
4.4.4 R (ASCII 122) Internal Register Designator 4-5
4.4.5 S (ASCII 123) Processor Status Word (PSW) 4-6
4.4.6 G (ASCII 107) Go 4-6
4.4.7 P (ASCII 120) Proceed 4-6
4.4.8 DD, DX, DY Bootstraps 4-6
4.4.9 X (ASCII 130) Diagnostics 4-8
4.5 INITIALIZATION 4-8
4.6 WARNINGS AND PROGRAMMING HINTS 4-8
4.6.1 Error Decoding 4-8
4.6.2 ODT Stack Warning 4-8
4.6.3 Addresses to Avoid 4-8
4.6.4 CPU Priority 4-9
4.6.5 Terminal Related Problems 4-9
4.6.6 Spurious Halts 4-9
4.6.7 Serial I/O Protocol 4-9
4.6.8 Interrupt Vector Initialization 4-9
4.6.9 Boot ROM Address Scheme 4-9

CHAPTER 5 SYSTEM ARCHITECTURE

5.1 INTRODUCTION 5-1
5.2 MICROPROCESSOR ARCHITECTURE 5-1
5.2.1 Registers 5-1
5.2.1.1 General Registers 5-1
5.2.1.2 Status Register 5-1
5.2.2 Hardware Stack 5-2
5.2.3 Interrupts 5-2
5.3 DMA (DIRECT MEMORY ACCESS) 5-4
5.4 MEMORY ORGANIZATION 5-4
5.5 POWER-UP/POWER-DOWN FACILITY 5-4

CHAPTER 6 PROGRAMMING INFORMATION

6.1 INTRODUCTION 6-1
6.2 ASYNCHRONOUS SERIAL LINE UNITS 6-1
6.2.1 Data Baud Rates 6-1
6.2.2 Interrupts 6-7

CONTENTS (Cont)

		Page
6.3	PROGRAMMING THE PARALLEL I/O INTERFACE	6-7
6.3.1	Modes of Operation	6-7
6.3.1.1	Port C Register	6-10
6.3.1.2	Mode 0 Basic Input/Output	6-10
6.3.1.3	Port A and B Registers	6-11
6.3.1.4	Port C Register in Mode 0	6-11
6.3.1.5	Mode 1 (Strobed Input/Output)	6-11
6.3.1.6	Mode 2 (Strobed Bidirectional I/O)	6-18
6.3.2	Control Word Register	6-22
6.3.2.1	Mode Selection	6-22
6.3.2.2	Setting Bits in Port C	6-22
6.3.3	Parallel I/O Initialization	6-25
6.3.4	Parallel I/O Handshaking	6-25
CHAPTER 7	**ADDRESSING MODES AND INSTRUCTION SET**	
7.1	INTRODUCTION	7-1
7.2	ADDRESSING MODES	7-1
7.2.1	Single Operand Addressing	7-3
7.2.2	Double Operand Addressing	7-3
7.2.3	Direct Addressing	7-4
7.2.3.1	Register Mode (Mode 0)	7-6
7.2.3.2	Autoincrement Mode (Mode 2)	7-8
7.2.3.3	Autodecrement Mode (Mode 4)	7-10
7.2.3.4	Index Mode (Mode 6)	7-12
7.2.4	Deferred (Indirect) Addressing	7-14
7.2.5	Use of the PC as a General-Purpose Register	7-17
7.2.5.1	Immediate Mode	7-18
7.2.5.2	Absolute Addressing	7-19
7.2.5.3	Relative Addressing	7-20
7.2.5.4	Relative Deferred Addressing	7-21
7.2.6	Use of the Stack Pointer as a General-Purpose Register	7-22
7.3	INSTRUCTION SET	7-22
7.3.1	Instruction Formats	7-23
7.3.2	List of Instructions	7-26
7.3.3	Single Operand Instructions	7-28
7.3.3.1	General	7-29
7.3.3.2	Shifts and Rotates	7-32
7.3.3.3	Multiple Precision	7-37
7.3.3.4	PS Word Operators	7-40
7.3.4	Double Operand Instructions	7-41
7.3.4.1	General	7-41
7.3.4.2	Logical	7-44
7.3.5	Program Control Instructions	7-46
7.3.5.1	Branches	7-46
7.3.5.2	Signed Conditional Branches	7-51
7.3.5.3	Unsigned Conditional Branches	7-53
7.3.5.4	Jump and Subroutine Instructions	7-55

CONTENTS (Cont)

		Page
7.3.5.5	Traps	7-60
7.3.5.6	Reserved Instruction Traps	7-64
7.3.5.7	HALT Interrupt	7-64
7.3.5.8	Trace Trap	7-64
7.3.5.9	Power Failure Interrupt	7-64
7.3.5.10	Interrupts	7-64
7.3.5.11	Special Cases (T-bit)	7-65
7.3.6	Miscellaneous Instructions	7-65
7.3.7	Condition Code Operators	7-67
CHAPTER 8	**THEORY OF OPERATION**	
8.1	INTRODUCTION	8-1
8.2	MICROPROCESSOR	8-1
8.2.1	Microprocessor Initialization	8-1
8.2.1.1	RESET Instruction	8-4
8.2.1.2	Power-up Input (PUP)	8-4
8.2.2	Clock Input (−TCLK)	8-5
8.2.3	Ready Input (READY)	8-5
8.2.4	Microprocessor Control Signals	8-5
8.2.4.1	Row Address Strobe (RAS)	8-5
8.2.4.2	Column Address Strobe (CAS)	8-5
8.2.4.3	Priority In (PI)	8-5
8.2.4.4	Read/Write (R/−WHB and R/−WLB)	8-5
8.2.4.5	Select Output Flags (SEL0 and SEL1)	8-5
8.2.4.6	Bus Clear (BCLR)	8-6
8.2.4.7	Clock Out (COUT)	8-6
8.2.5	Microprocessor Transactions	8-6
8.2.5.1	Fetch/Read	8-6
8.2.5.2	Write	8-8
8.2.5.3	IAK	8-8
8.2.5.4	DMA	8-8
8.2.5.5	ASPI	8-8
8.2.5.6	NOP	8-8
8.3	MODE REGISTER CONTROL	8-13
8.4	INTERRUPT CONTROL	8-14
8.4.1	Interrupt Control Logic	8-16
8.4.2	Ready Logic	8-17
8.4.3	IAK Data In (IAKDIN)	8-18
8.4.4	HALT Interrupt	8-19
8.4.5	Power Fail (−PFAIL)	8-19
8.4.6	Local	8-20
8.4.7	External	8-20
8.4.8	DMA Interrupt	8-20
8.5	DC004 PROTOCOL	8-20
8.6	ADDRESS LATCH	8-21
8.7	MEMORY ADDRESS DECODE	8-21
8.8	RAM MEMORY	8-21
8.9	ROM/RAM MEMORY SOCKETS	8-21

CONTENTS (Cont)

		Page
8.10	SERIAL LINE INTERFACE UNITS	8-23
8.11	PARALLEL I/O INTERFACE	8-25
8.12	POWER-UP	8-27
8.13	CLOCK	8-28
8.14	CLOCK CONTROL	8-29
8.15	DMA	8-29
8.16	TSYNC	8-31
8.17	READ/WRITE	8-31
8.18	REPLY TIME-OUT	8-33
8.19	BUS CONTROL	8-33
CHAPTER 9	**LSI-11 BUS**	
9.1	INTRODUCTION	9-1
9.2	SBC-11/21 PLUS SINGLE-BOARD COMPUTER	9-2
9.3	MASTER/SLAVE RELATIONSHIP	9-2
9.4	DATA TRANSFER BUS CYCLES	9-3
9.4.1	Bus Cycle Protocol	9-4
9.4.2	Direct Memory Access	9-8
9.5	INTERRUPTS	9-12
9.5.1	Device Priority	9-12
9.5.2	Interrupt Protocol	9-12
9.6	CONTROL FUNCTIONS	9-14
9.6.1	Halt	9-14
9.6.2	Initialization	9-14
9.6.3	Power Status	9-14
9.6.4	Power-Up/Power-Down Protocol	9-15
9.7	LSI-11 BUS ELECTRICAL CHARACTERISTICS	9-16
9.8	MODULE CONTACT FINGER IDENTIFICATION	9-16
APPENDIX A	**INSTRUCTION TIMING**	
APPENDIX B	**PROGRAMMING DIFFERENCE LIST**	
APPENDIX C	**SOFTWARE DEVELOPMENT**	
APPENDIX D	**MACRO-ODT ROM LISTING FOR KXT11-A2 OPTION**	
APPENDIX E	**MACRO-ODT ROM LISTING FOR KXT11-A5 OPTION**	
APPENDIX F	**SBC-11/21 PLUS SCHEMATICS**	
APPENDIX G	**GLOSSARY**	
INDEX		

FIGURES

Figure No.	Title	Page
1-1	KXT11-AB (M7676) SBC-11/21 PLUS Module	1-1
2-1	SBC-11/21 PLUS Module Layout	2-2
2-2	Interrupt Configurations	2-10
2-3	Time-out During LSI-11 Bus Interrupt Acknowledge	2-11
2-4	Parallel I/O Configuration	2-12
2-5	Socket Sets A and B Interconnection	2-17
2-6	Memory Configuration	2-18
2-7	Memory Maps	2-19
2-8	30-Pin Parallel I/O Connector	2-24
2-9	10-Pin Serial Line Unit Connector	2-26
2-10	BC20N-05 Null Modem Cable	2-27
2-11	BC21B-05 Modem Cable	2-27
5-1	Registers and Processor Status Word	5-2
5-2	Memory Maps	5-6
6-1	Serial Line Unit (SLU) Interface	6-2
6-2	Serial Line Unit Register Bit Maps	6-4
6-3	Parallel I/O Interface	6-8
6-4	Parallel I/O Flowchart	6-9
6-5	Mode 0 Port A or B Bit Assignments	6-11
6-6	Mode 0 Port C Bit Assignments	6-12
6-7	Mode 1 Port C Bit Assignments	6-15
6-8	Mode 2 Port C Bit Assignments	6-20
6-9	Mode 1 Input Data Handshaking Sequence	6-26
6-10	Mode 1 Strobed Input Timing	6-26
6-11	Mode 1 Output Data Handshaking Sequence	6-27
6-12	Mode 1 Port A Strobed Output Timing	6-28
6-13	Mode 1 Port B Strobed Output Timing	6-28
6-14	Mode 2 Port A Bidirectional Timing	6-29
7-1	Single Operand Addressing	7-3
7-2	Double Operand Addressing	7-3
7-3	Mode 0 Register	7-5
7-4	Mode 2 Autoincrement	7-5
7-5	Mode 4 Autodecrement	7-5
7-6	Mode 6 Index	7-5
7-7	INC R3	7-6
7-8	ADD R2,R4	7-7
7-9	COMB R4	7-7
7-10	CLR (R5)+	7-8
7-11	CLRB (R5)+	7-9
7-12	ADD (R2)+,R4	7-9
7-13	INC –(R0)	7-10
7-14	INCB –(R0)	7-11
7-15	ADD –(R3),R0	7-11
7-16	CLR 200(R4)	7-12
7-17	COMB 200(R1)	7-13
7-18	ADD 30(R2),20(R5)	7-13
7-19	Mode 1 Register Deferred	7-14
7-20	Mode 3 Autoincrement Deferred	7-14

FIGURES (Cont)

Figure No.	Title	Page
7-21	Mode 5 Autodecrement Deferred	7-15
7-22	Mode 7 Index Deferred	7-15
7-23	CLR @R5	7-15
7-24	INC @(R2)+	7-16
7-25	COM @ –(R0)	7-16
7-26	ADD @1000(R2),R1	7-17
7-27	ADD #10,R0	7-19
7-28	CLR @#1100	7-19
7-29	ADD @#2000,R3	7-20
7-30	INC A	7-21
7-31	CLR @A	7-22
7-32	Byte Instructions	7-25
7-33	Multiple Precision	7-37
8-1	SBC-11/21 PLUS Functional Block Diagram	8-2
8-2	SBC-11/21 PLUS Microprocessor	8-4
8-3	Fetch/Read Transaction	8-7
8-4	Write Transaction	8-9
8-5	IAK Transaction	8-10
8-6	DMA Transaction	8-11
8-7	ASPI Transaction	8-12
8-8	BUS NOP Transaction	8-12
8-9	Mode Register Control	8-13
8-10	SBC-11/21 PLUS Interrupt Control	8-14
8-11	Interrupt Control Logic	8-16
8-12	Ready	8-18
8-13	IAKDIN	8-18
8-14	HALT Interrupt	8-19
8-15	Memory Maps	8-22
8-16	RAM Memory	8-23
8-17	Serial Line Interface Units	8-24
8-18	Parallel I/O Interface	8-26
8-19	Power-up	8-27
8-20	Clock	8-28
8-21	Clock Control	8-29
8-22	DMA	8-30
8-23	TSYNC	8-31
8-24	Read/Write	8-32
8-25	Reply Time-out	8-33
8-26	Bus Control	8-34
9-1	DATI Bus Cycle	9-6
9-2	DATI Bus Cycle Timing	9-7
9-3	DATO or DATOB Bus Cycle	9-9
9-4	DATO or DATOB Bus Cycle Timing	9-10
9-5	DMA Protocol	9-11
9-6	DMA Request/Grant Timing	9-12
9-7	Interrupt Request/Acknowledge Sequence	9-13
9-8	Power-Up/Power-Down Timing	9-15
9-9	Double-Height Module Contact Finger Identification	9-16

FIGURES (Cont)

Figure No.	Title	Page
C-1	Overview of Software Development	C-4
C-2	Application Overview	C-5
C-3	Monitor Program	C-8
C-4	Load Map	C-11
C-5	Power-up Task	C-11
C-6	Power Fail Recovery	C-12
C-7	SLU Diagnostic Task	C-12
C-8	RAM Diagnostic Task	C-13
C-9	ROM Diagnostic Task	C-13
C-10	Parallel I/O Diagnostic Task	C-14
C-11	Control Task	C-14
C-12	Power Fail Task	C-15
H-1	SBC-11/21 PLUS Memory Maps	H-3

TABLES

Table No.	Title	Page
1-1	SBC-11/21 PLUS Module Backplane Pin Identification	1-5
1-2	Related Documentation	1-7
2-1	Configuration Pin Definitions	2-3
2-2	Configuration Pin Functions	2-5
2-3	Standard Factory Configuration	2-7
2-4	Mode Register Configuration	2-9
2-5	Mode 0 Buffer Configuration (No Handshake)	2-13
2-6	Mode 1 Buffer Configuration (Strobed I/O)	2-14
2-7	Mode 2 Buffer Configuration and Handshake	2-15
2-8	SLU1 BREAK Detection	2-15
2-9	Memory Map Configurations	2-20
2-10	Socket Set A Configuration for EPROM/PROM/EEPROM	2-20
2-11	Socket Set B Configuration for EPROM/PROM/EEPROM	2-21
2-12	Socket Set A Configuration for RAM	2-22
2-13	Socket Set B Configuration for RAM	2-22
2-14	EIA Slew Rate Resistor Values	2-26
2-15	Diagnostic Fault Indicators	2-29
4-1	Macro-ODT Commands	4-3
4-2	Macro-ODT States and Valid Input Characters	4-4
5-1	Processor Status Word Bit Descriptions	5-3
5-2	PSW Interrupt Levels	5-3
5-3	SBC-11/21 PLUS Interrupts	5-5
6-1	Serial Line Unit Register Addresses	6-3
6-2	Receiver Control and Status Bit Descriptions	6-4
6-3	Receiver Data Buffer Bit Descriptions	6-4
6-4	Transmitter Control and Status Bit Descriptions	6-4
6-5	Transmitter Data Buffer Bit Descriptions	6-7
6-6	Parallel I/O Register Addresses	6-8

TABLES (Cont)

Table No.	Title	Page
6-7	Mode 0 Configuration	6-10
6-8	Mode 0 Port A or B Bit Descriptions	6-11
6-9	Mode 0 Port C Bit Descriptions	6-12
6-10	Port C Control Signals in Mode 1	6-13
6-11	Combinations of Mode 1	6-14
6-12	Mode 1 Port C Bit Descriptions	6-15
6-13	Mode 1 Configuration	6-18
6-14	Port C Control Signals in Mode 2	6-19
6-15	Mode 2 Port C Bit Descriptions	6-20
6-16	Mode 2 Configuration	6-21
6-17	Control Register Mode Selection Bit Functions	6-22
6-18	Control Words for Mode Selection	6-23
6-19	Control Register Bit Set/Reset Functions	6-24
6-20	Interrupt Set/Reset Control Words	6-24
6-21	Mode 1 Input Handshaking Signals	6-25
6-22	Mode 1 Output Handshaking Signals	6-27
6-23	Mode 2 Bidirectional Handshaking Signals	6-29
7-1	Sample SBC-11/21 PLUS Instructions	7-4
7-2	Direct Addressing Modes	7-4
7-3	Indirect Addressing Modes	7-14
7-4	Register Addressing Modes	7-18
7-5	SBC-11/21 PLUS Instruction Set	7-26
8-1	Start Address Configurations	8-14
8-2	Designated Interrupts	8-17
8-3	Serial Line Unit Registers	8-25
8-4	PPI Addressable Registers	8-26
9-1	Signal Assignments	9-3
9-2	Data Transfer Operations	9-4
9-3	Bus Signals Used in Data Transfer Operations	9-4
9-4	Bus Pin Identifiers	9-17
A-1	Instruction Timing	A-1
B-1	SBC-11/21 PLUS, LSI-11/2, and LSI/11/23 Comparisons	B-1
B-2	Illegal Address Traps	B-6
H-1	FPLA Code for the SBC-11/21 PLUS	H-2
H-2	Option 23-77C6-00 FPLA Code	H-4
H-3	Functional Equivalent Pins	H-8

PREFACE

This User's Guide provides the user with configuration, system architecture, and programming information for the SBC-11/21 PLUS single-board computer. Appendix H presents a list of the differences between the SBC-11/21 and the SBC-11/21 PLUS so that a user who is familiar with the SBC-11/21 may easily configure the SBC-11/21 PLUS.

NOTE

This User's Guide is for use with the SBC-11/21 PLUS module, M7676 and subsequent revisions only. This revision is identified by the circuit board 5016277 located on the module as described in Figure 1-1.

CHAPTER 1
INTRODUCTION

1.1 INTRODUCTION

The KXT11-AB (M7676) module, called the SBC-11/21 PLUS single-board computer, is shown in Figure 1-1. It is a complete computer system on an 8.5 × 5.2 inch printed circuit board that executes the well known PDP-11 instruction set (see Appendix B). The SBC-11/21 PLUS module contains 16Kb (kilobytes) of RAM, sockets for up to 32Kb of PROM or additional RAM, two serial I/O lines, twenty-four lines of parallel I/O, and a 50 Hz, 60 Hz, or 800 Hz real-time clock. In addition, the SBC-11/21 PLUS supports the complete LSI-11 bus interface that enables it to communicate with most of Digital's large family of modules (see Chapter 3). For a description refer to the *Microcomputer Interfaces* and *Microcomputers and Memories* handbooks.

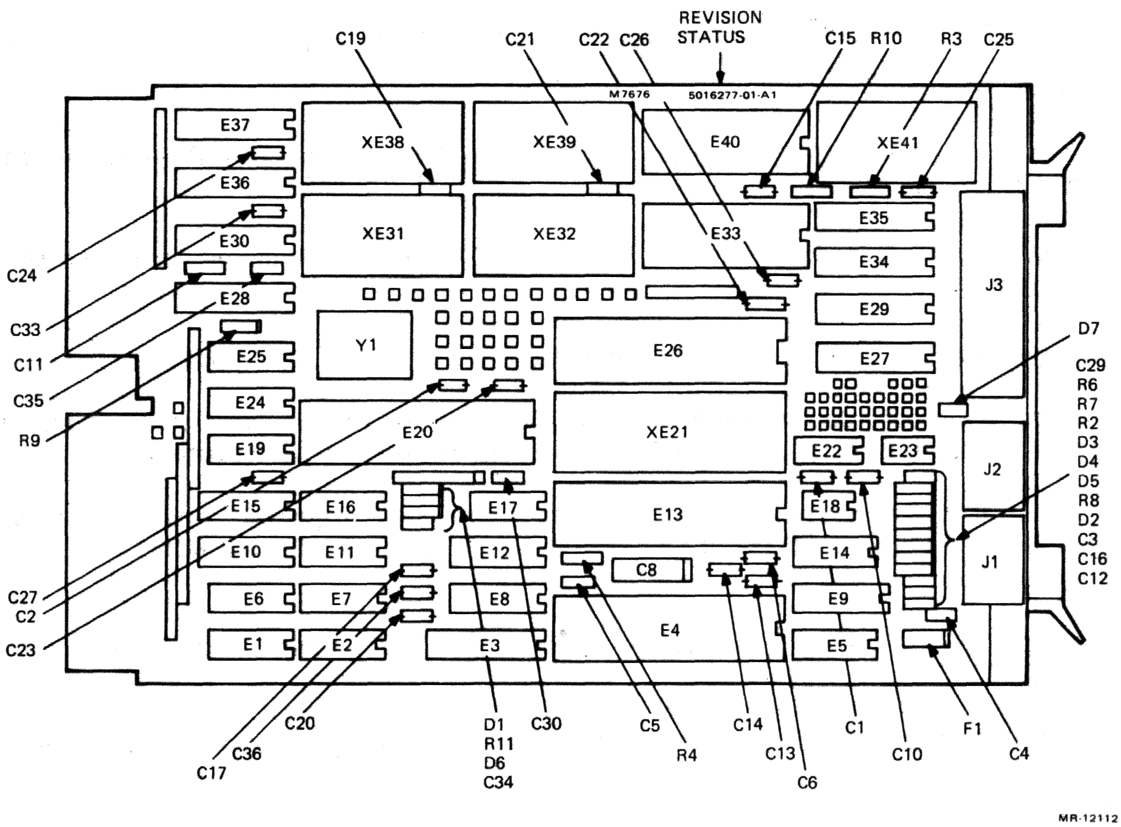


Figure 1-1 KXT11-AB (M7676) SBC-11/21 PLUS Module

The SBC-11/21 PLUS computer features the following:

- A powerful processor running the PDP-11 instruction set.
- Direct addressing of 32K, 16-bit words or 64K, 8-bit bytes (K = 1024).
- Efficient processing of 8-bit characters without the need to rotate, swap, or mask.
- On-board 16Kb of static read/write memory.
- Sockets for up to 32Kb of PROM for a wide range of memory types from many vendors. Additional RAM can also be installed in these sockets.
- Hardware memory stack for handling data, subroutines, and interrupts.
- Direct memory access for high data rate devices.
- Eight general-purpose registers for data storage, pointers, and accumulators; two are dedicated: stack pointer (SP) and program counter (PC).
- Fast on-board bus for high throughput when external memory access is not needed.
- LSI-11 bus structure that provides position dependent priority for peripheral device interfaces connected to the bus.
- Fast vectored interrupt response without device polling.
- A powerful set of instructions.
- Two serial I/O interfaces, compatible with EIA RS-232C and EIA RS-423, with software programmable baud rates over the range of 300 to 38,400 baud.
- One parallel I/O interface with two bidirectional 8-bit input/output ports and one 8-bit control port.
- Real-time clock that can be set by the user to 50 Hz, 60 Hz, or 800 Hz.
- Jumper-selected operating modes, including four memory maps, exception handling, start and restart addresses, parallel I/O configurations, and real-time clock frequency.
- Optional PROM resident Macro-ODT containing module diagnostics, bootstrap programs for mass storage devices (TU58, RX01, RX02, RL01, RL02, RX50, and RD51 devices), console communications, and on-line debugging facility.
- Supports RT-11 V5.1 or subsequent versions and MicroPower/Pascal V1.5 or subsequent versions.
- For detailed differences between the SBC-11/21 and SBC-11/21 PLUS see Appendix H.

1.2 SPECIFICATIONS

The SBC-11/21 PLUS module specifications follow:

1.2.1 Physical

Height	13.2 cm (5.2 in)
Length (includes module handle)	22.8 cm (8.9 in)
Width	1.27 cm (0.5 in)
Weight	255 g (8.5 oz) typical

1.2.2 Power Requirements

Power Supply:	
+5.0 V ± 5%	2.5 A (typical), 2.8 A (maximum)
+12.0 V ± 5%	60 mA (typical) used by on-board circuitry, 1.1 A (maximum) includes current provided to outside interface through pin 10 of the serial I/O connector
Battery Backup:	
+5.0 V ± 5%	170 mA (typical), 260 mA (maximum)

NOTE

The +12.0 V typical current is measured with no connections at pin 10 of the serial I/O connectors (fused line).

1.2.3 Bus Loading

AC Loads	2.7
DC Loads	0.5

1.2.4 Environmental

Temperature:

Storage	−40° C to 66° C (−40° F to 150° F)
Operating	5° C to 60° C (41° F to 140° F)

NOTE
The module must be brought into the operating temperature environment and allowed to stabilize before operating.

Relative Humidity:

Storage	10% to 95% (no condensation)
Operating	10% to 95% (no condensation)

Altitude:

Storage	Up to 15 km (50,000 ft)
Operating	Up to 15 km (50,000 ft) (90 mm mercury minimum)

NOTE
Lower the maximum operating temperature by 1° C (1.8° F) for each 300 m (1,000 ft) of altitude above 2.4 km (8,000 ft).

Environment:	Air must be noncaustic.
Airflow (operating):	There must be enough airflow to limit the input to output temperature rise across the module to 5° C (9° F) when the input temperature is 60° C (140° F). For operation below 55° C (131° F), there must be enough airflow to limit the input to output temperature rise across the module to 10° C (18° F) maximum.

NOTE
These are design limits. Lower temperature limits will help increase the life of the product.

1.3 BACKPLANE PIN IDENTIFICATION

Table 1-1 lists backplane pin connections for the SBC-11/21 PLUS module, pin identification and signal names unique to the SBC-11/21 PLUS module, and standard LSI-11 bus backplane names assigned to each pin. Although the signal names may differ, the module is completely LSI-11 bus compatible with the exception of bus refresh transaction (BREF) which is not performed by the SBC-11/21 PLUS. Signals STOP L, SRUN L, and START L are not used on the LSI-11 bus. These are TTL level signals unique to the SBC-11/21 PLUS.

Table 1-1 SBC-11/21 PLUS Module Backplane Pin Identification

Backplane Pin	SBC-11/21 PLUS Signal Function	LSI-11 Bus Signal Name
Side 1 (Component Side)		
AA1	Bus terminator	BIRQ5 L
AB1	Bus terminator	BIRQ6 L
AC1	Bus terminator	BDAL16 L
AD1	Bus terminator	BDAL17 L
AE1	STOP L	SSPARE1
AF1	SRUN L	SSPARE2
AH1	Not connected	SSPARE3
AJ1	GND	GND
AK1	Not connected	MSPAREA
AL1	GND	MSPAREA
AM1	GND	GND
AN1	BDMR L	BDMR L
AP1	BHALT L	BHALT L
AR1	Bus terminator	BREF L
AS1	Not connected	+12B
AT1	GND	GND
AU1	Not connected	PSPARE1
AV1	+5 VB (battery)	+5B
BA1	BDCOK H	BDCOK H
BB1	BPOK H	BPOK H
BC1	Bus terminator	SSPARE4
BD1	Bus terminator	SSPARE5
BE1	Bus terminator	SSPARE6
BF1	Bus terminator	SSPARE7
BH1	START L	SSPARE8
BJ1	GND	
BK1	Not connected	MSPAREB
BL1	Not connected	MSPAREB
BM1	GND	
BN1	BSACK L	BSACK L
BP1	Bus terminator	BIRQ7 L
BR1	BEVNT L	BEVNT L
BS1	Not connected	+12B
BT1	GND	GND
BU1	Not connected	PSPARE2
BV1	+5 V	+5 V

Table 1-1 SBC-11/21 PLUS Module Backplane Pin Identification (Cont)

Backplane Pin	SBC-11/21 PLUS Signal Function	LSI-11 Bus Signal Name
Side 2 (Solder Side)		
AA2	+5 V	+5 V
AB2	Not connected	−12 V
AC2	GND	GND
AD2	+12 V	+12 V
AE2	BDOUT L	BDOUT L
AF2	BRPLY L	BRPLY L
AH2	BDIN L	BDIN L
AJ2	BSYNC L	BSYNC L
AK2	BWTBT L	BWTBT L
AL2	BIRQ4 L	BIRQ4 L
AM2	Not connected	BIAKI L
AN2	BIAKO L	BIAKO L
AP2	BBS7 L	BBS7 L
AR2	Not connected	BDMGI L
AS2	BDMGO L	BDMGO L
AT2	BINIT L	BINIT L
AU2	BDAL0 L	BDAL0 L
AV2	BDAL1 L	BDAL1 L
BA2	+5 V	+5 V
BB2	Not connected	−12 V
BC2	GND	GND
BD2	Not connected	+12 V
BE2	BDAL2 L	BDAL2 L
BF2	BDAL3 L	BDAL3 L
BH2	BDAL4 L	BDAL4 L
BJ2	BDAL5 L	BDAL5 L
BK2	BDAL6 L	BDAL6 L
BL2	BDAL7 L	BDAL7 L
BM2	BDAL8 L	BDAL8 L
BN2	BDAL9 L	BDAL9 L
BP2	BDAL10 L	BDAL10 L
BR2	BDAL11 L	BDAL11 L
BS2	BDAL12 L	BDAL12 L
BT2	BDAL13 L	BDAL13 L
BU2	BDAL14 L	BDAL14 L
BV2	BDAL15 L	BDAL15 L

1.4 RELATED DOCUMENTS

This User's Guide is the primary reference document for the SBC-11/21 PLUS. Important information about other LSI-11 bus compatible products may be found in the publications listed in Table 1-2.

Table 1-2 Related Documentation

Title	Document Number
Microcomputers and Memories Handbook, 1982 Edition	EB-20912-20
Microcomputer Interfaces Handbook, 1980 Edition	EB-20175-20
PDP-11 Bus Handbook, 1979 Edition	EB-17525-20
Guide to RT-11 Documentation	AA-5285G-TC
MicroPower/Pascal RT-11	QJ029GZ

These documents can be ordered from:

Digital Equipment Corporation
Printing and Circulation Services
444 Whitney Street
Northboro, MA 01532

Attention: Communications Services (NR2/M15)
Customer Services Section

CHAPTER 2
INSTALLATION

2.1 INTRODUCTION
The installation of the SBC-11/21 PLUS single-board computer module is discussed in this chapter. The following five items, which are an integral part of the installation procedure, are covered in detail.

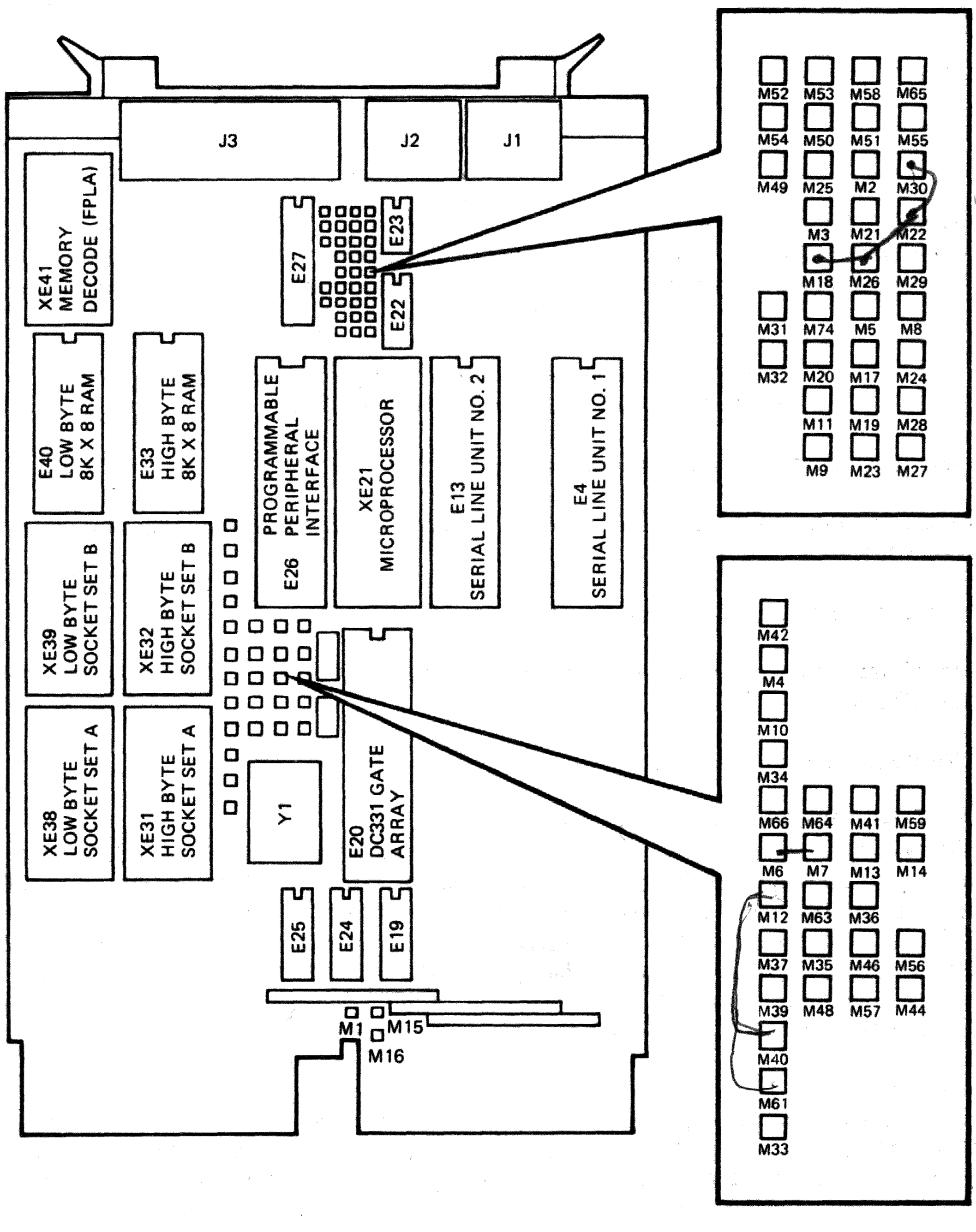
NOTE
It is best to leave the factory configuration as is until module performance has been verified.

1. Installing jumpers to select operational features.
2. Selecting and mounting an LSI-11 bus structured backplane and adding any required LSI-11 bus options.
3. Selecting and connecting an appropriate power supply.
4. Providing appropriate cables to connect external devices to the serial line and parallel I/O interfaces.
5. Verifying operation of the module.

2.2 SELECTING OPERATIONAL FEATURES
The module has sixty-one wirewrap pins with which the user configures the module for the operating modes necessary to meet any requirements. This is done by either installing or removing jumper wires between the wirewrap pins. The locations and identification numbers of the wirewrap pins are illustrated in Figure 2-1. Table 2-1 defines the wirewrap pins, and Table 2-2 lists the pin functions by the features they support. The selectable features are battery backup, power-up, starting address, interrupts, parallel I/O buffers, and memory maps. Detailed requirements for each of these configurations are described in the following paragraphs. The standard factory configuration is described in Table 2-3.

2.2.1 Battery Backup
The user can select battery backup mode to maintain a +5 Vdc battery supply to: (1) the 16Kb of on-board static RAM, (2) 24-pin devices in socket set A, and (3) 28-pin or 24-pin devices in socket set B. The +5 Vdc battery supply is provided through the LSI-11 bus pin AV1. A maximum of 2mA is required. This supply is connected to wirewrap pin M16.

To enable battery backup, the jumper wire between M1 and M15 is removed and a jumper wire is installed between M16 and M15. This provides battery backed up power for the 16Kb of on-board static RAM. To enable battery backup of 24-pin devices in socket set A, a jumper wire is installed between M7 and M41. To enable battery backup of 28-pin devices in socket set B, a jumper wire between M59 and M41 is installed. To enable battery backup of 24-pin devices in socket set B, a jumper wire between M66 and M34 is installed.



172000

Figure 2-1 SBC-11/21 PLUS Module Layout

Table 2-1 Configuration Pin Definitions

Pin	Schematic Sheet Number	Description
M1	5	System +5 V power (+5 VNCR)
M2	2	System GND
M3	2	High logic level (+3 Vdc)
M4	3	Wake up circuit diode, anode side
M5	3	Receive side of BHALT line transceiver
M6	3	Wake up circuit diode, cathode side (+5 VNCR)
M7	5	Socket set A, high and low byte, pin 26
M8	3	BREAK request clock line
M9	6	High logic level (+3 Vdc)
M10	5	Address line 14
M11	6	System GND
M12	5	High logic level (+5 VNCR)
M13	5	Socket set B, high and low byte, pin 1
M14	5	Socket set A, high and low byte, pin 1
M15	5	+5 Vdc power distribution to support static RAM
M16	5	Battery backup +5 Vdc power source
M17	6	Serial line unit (SLU) 1 BREAK detect, interrupt request output
M18	2	High logic level (+3 Vdc)
M19	6	60 Hz real-time clock output
M20	3	Transmit side of BHALT line transceiver
M21	2	Memory map select (MSB)
M22	1	Start address control (TDAL 15)
M23	6	Transmit side of BEVNT line transceiver
M24	5	System GND
M25	2	Memory map select (LSB)
M26	1	Start address control (TDAL 14)
M27	6	50 Hz real-time clock output
M28	6	800 Hz real-time clock output
M29	1	System GND
M30	1	Start address control (TDAL 13)
M31	3	System GND
M32	1	System GND
M33	5	Socket set B, high byte, pin 27
M34	5	High logic level (+5 VCR)
M35	5	Socket set B, low byte, pin 23

Table 2-1 Configuration Pin Definitions (Cont)

Pin	Schematic Sheet Number	Description
M36	5	High logic level (+5 VNCR)
M37	5	Socket set A, low byte, pin 27
M39	5	Socket set A, high byte, pin 27
M40	5	Socket set A, low byte, pin 23
M41	5	High logic level (+5 VCR)
M42	5	Address line 15
M44	5	Address line 12
M46	5	High logic level for PROMs (+5 VNCR)
M48	5	Socket set B, low byte, pin 27
M49	7	Port B buffer direction control
M50	2	High logic level (+3 Vdc)
M51	7	System GND
M52	7	Port A buffer direction control
M53	7	Port C buffered output, to J3 pin 7
M54	7	Port C PC6 output (E26 pin 11)
M55	5	System GND
M56	5	High byte write strobe (–WHB)
M57	5	Low byte write strobe (–WLB)
M58	7	Port C PC4 output (E26 pin 13)
M59	5	Socket set B, high and low byte, pin 28
M61	5	Socket set A, high byte, pin 23
M63	5	Socket set B, high byte, pin 23
M64	5	RAM, high and low byte, pin 26
M65	7	Port C buffered output, to J3 pin 5
M66	5	Socket set B, high and low byte, pin 26
M74	1	–CTMER interrupt enable

Table 2-2 Configuration Pin Functions

Pin	Function	Description
	Battery backup	
M16		Battery backup +5 Vdc power source
M15		+5 Vdc power distribution to support static RAM
M1		System +5 V power (+5 VNCR)
M36		High logic level (+5 VNCR)
M41		High logic level (+5 VCR)
M34		High logic level (+5 VCR)
M14		Socket set A, high and low byte, pin 1
M13		Socket set B, high and low byte, pin 1
M7		Socket set A, high and low byte, pin 26
M66		Socket set B, high and low byte, pin 26
M59		Socket set B, high and low byte, pin 28
M64		RAM, high and low byte, pin 26
M4		Wake-up circuit diode, anode side
	Power-up (wake-up circuitry)	
M6		System +5 V power, wake-up circuit diode, cathode side (+5 VNCR)
M4		Wake-up circuit diode, anode side
	Serial line unit (SLU) 1	
M31		System GND
M20		Transmit side of BHALT line transceiver
M17		Serial line unit (SLU) 1 BREAK detect
	Serial line unit (SLU) 2	
M23		Transmit side of BEVNT line transceiver
M27		50 Hz real-time clock output
M19		60 Hz real-time clock output
M28		800 Hz real-time clock output
M11		System GND
M9		High logic level (+3 Vdc)
	Memory map decoder	
M3		High logic level (+3 Vdc)
M25		Memory map select (LSB)
M21		Memory map select (MSB)
M2		System GND
	Start address (mode register)	

Table 2-2 Configuration Pin Functions (Cont)

Pin	Function	Description
M30		Start address control (TDAL 13)
M26		Start address control (TDAL 14)
M22		Start address control (TDAL 15)
M18		High logic level (+3 Vdc)
M29		System GND
	Nonmaskable interrupt and trap to the restart address	
M74		HALT request line
M32		System GND
	BHALT interrupt (level 7, maskable)	
M20		Transmit side of BHALT line transceiver
M31		System GND
M24		System GND
M17		Serial line unit (SLU) 1 BREAK detect, interrupt request output
M5		Receive side of BHALT line transceiver
M8		BREAK request clock line
	Memory	
M14		Socket set A, high and low byte, pin 1
M13		Socket set B, high and low byte, pin 1
M7		Socket set A, high and low byte, pin 26
M66		Socket set B, high and low byte, pin 26
M59		Socket set B, high and low byte, pin 28
M64		RAM, high and low byte, pin 26
M61		Socket set A, high byte, pin 23
M40		Socket set A, low byte, pin 23
M63		Socket set B, high byte, pin 23
M35		Socket set B, low byte, pin 23
M39		Socket set A, high byte, pin 27
M37		Socket set A, low byte, pin 27
M33		Socket set B, high byte, pin 27
M48		Socket set B, low byte, pin 27
M44		Address line 12
M56		High byte write strobe (−WHB)
M57		Low byte write strobe (−WLB)
M1		High logic level (+5 VNCR)
M6		High logic level (+5 VNCR)
M36		High logic level (+5 VNCR)
M12		High logic level (+5 VNCR)
M46		High logic level (+5 VNCR)
M34		High logic level (+5 VCR)
M15		High logic level (+5 VCR)
M41		High logic level (+5 VCR)

Table 2-2 Configuration Pin Functions (Cont)

Pin	Function	Description
	Parallel input/output	
M49		Port B buffer direction control
M51		System GND
M55		System GND
M65		Port C buffered output, to J3 pin 5
M53		Port C buffered output, to J3 pin 7
M58		Port C PC4 output (8255A-5 pin 13)
M54		Port C PC6 output (8255A-5 pin 11)
M50		High logic level (+3 Vdc)
M52		Port A buffer direction control

Table 2-3 Standard Factory Configuration

Function		Jumpers Installed Between
No Battery backup		M1 to M15
Wake-up circuit enabled		–
Start Address* 10000		M22 to M18
Restart address 10004		M26 to M29
		M30 to M26
Memories:		
Map 0		M64 and M7
		M25 to M21
		M21 to M2
Skt A contains 2K × 8 EPROM	skt A	M7 to M6
		M61 to M12
		M12 to M40
Skt B contains 8K × 8 SRAM	skt B	M59 to M36
		M33 to M56
		M48 to M57
		M66 to M4
		M63 to M44
		M35 to M44

* Before use with Macro-ODT, the start address must be changed to 172000 as described in Table 2-4.

Table 2-3 Standard Factory Configuration (Cont)

Function	Jumpers Installed Between
Interrupts:	
SLU1 BREAK asserts HALT and is received as level 7 interrupt (Vector 140)	M17 to M20 M5 to M8 M74 to M32
60 Hz Real time clock asserts BEVNT	M19 to M23
Parallel I/O Port A receive Port B transmit	M52 to M50 M49 to M51 M65 to M58

If the battery backup option is enabled, the wake-up circuitry must also be enabled (see Paragraph 2.2.2) for all RAMs on the board. The wake-up circuitry is enabled by ensuring that no jumper is installed between M4 and M6 (this is the standard factory configuration).

2.2.2 Wake-Up Circuit

The module has an on-board power wake-up circuit designed for use in systems without the LSI-11 bus power sequencing protocol or in systems with battery backup. This circuit holds the BDCOK line negated until one second after +5 V power is applied. When the module is used in an LSI-11 backplane that has a power sequencing routine, the module wake-up circuit must be disabled. To do this, a jumper wire is installed between M6 and M4. The jumper wire is removed when using power supplies without power sequencing or when the battery backup option is installed. The module requires the +5 Vdc and +12 Vdc power supplies to have a rise time of less than 50 ms.

2.2.3 Starting Address

The user selects the starting address for the microprocessor via wirewrap pins. When the module is powered up, the microprocessor loads this value into R7 (program counter) as the first fetch address. The wirewrap pins are M22, M26, M29, M30, and M18, and are defined in Table 2-1. The user can select from eight available starting addresses. Table 2-4 lists these available addresses and the jumper connections required for each address. The restart address is always the start address incremented by four. The wirewrap pin locations are shown in Figure 2-1.

2.2.4 Interrupts

The SBC-11/21 PLUS implements a multilevel interrupt system that has eleven separate interrupts. See Table 5-3 for a complete list of system interrupts. Three interrupts, CTMER, BKRQ, and REVNT, are user configurable by means of jumper wires as shown in Figure 2-2 and are discussed here.

Table 2-4 Mode Register Configuration

Start Address	Restart Address	Connect M22 to	Connect M26 to	Connect M30 to
000000	000004	M18	M29	M18
010000*	010004	M18	M29	M29
020000	020004	M29	M18	M18
040000	040004	M29	M18	M29
100000	100004	M29	M29	M18
140000	140004	M29	M29	M29
172000	172004	M18	M18	M18
173000	173004	M18	M18	M29

* Factory setting. The start address should be selected in conjunction with the memory map configuration. Figure 2-6 shows how the available start addresses fit into the memory maps.

The CTMER interrupt is at the highest level (nonmaskable). It is caused by a time-out, that is, a failure to detect RRPLY during a fetch/read, write, or IAK transaction. Such a condition could occur only if the peripheral that caused the interrupt failed to return BRPLY during the vector reading operation. See Chapter 8 for a discussion of external interrupts. Figure 2-3 describes the sequence of events that takes place during the IAK time-out.

The other two interrupts the user can select are BKRQ and BEVNT. Their vectors and priorities are described in Table 5-3. All jumper combinations, which are "electrically correct" as shown in Figure 2-2, are legal.

A description of some typical configurations follows to familiarize the user with the different combinations available.

Install jumpers between	M7 and M74
	M20 and M31
	M28 and M23
	M8 and M24

This arrangement allows the SLU1 BREAK input to set the –CTMER nonmaskable interrupt and trap to the restart address. The BHALT L bus signal is ignored. The SLU2 800 Hz line time clock and the BEVNT L bus signal enable the REVNT interrupt.

Install jumpers between	M5 and M74
	M17 and M8
	M20 and M31
	M23 and M71

This arrangement allows the BHALT L bus signal to set the –CTMER nonmaskable interrupt and trap to the restart address. The SLU1 BREAK input sets the BKRQ level 7 maskable interrupt, and only the BEVNT L bus signal enables the REVNT interrupt.

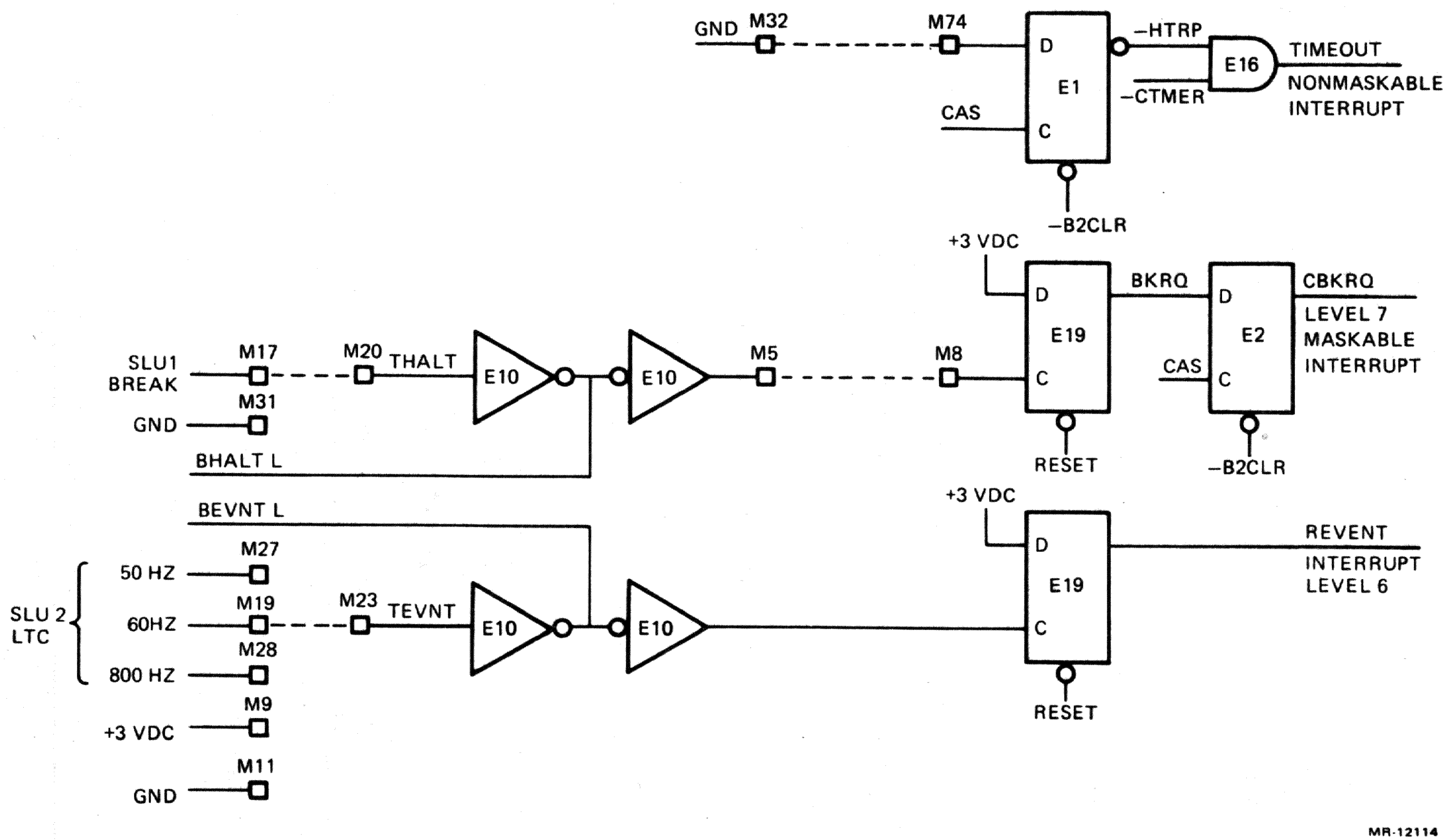


Figure 2-2 Interrupt Configurations

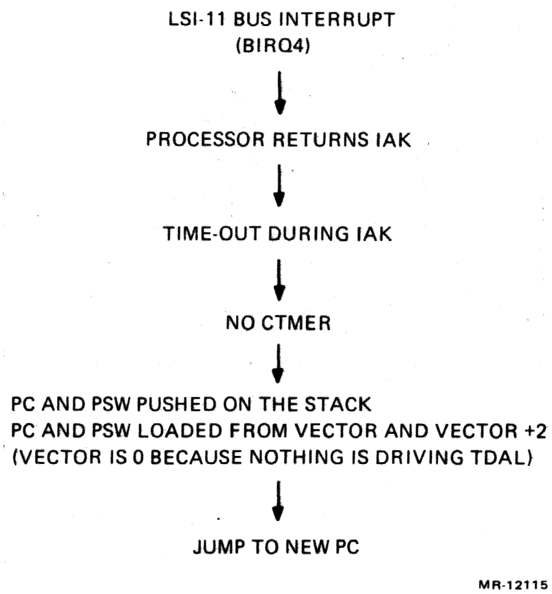


Figure 2-3 Time-out During LSI-11 Bus Interrupt Acknowledge

Install jumpers between	M74 and M32
	M17 and M20
	M5 and M8
	M23 and M9

This arrangement allows the time-out (TMER) to set the –CTMER nonmaskable interrupt for all time-outs. The SLU1 BREAK or the BHALT bus signal set the BKRQ level 7 maskable interrupt, and the BEVNT L bus line is clamped low and therefore, no interrupts can be generated by BEVNT L.

2.2.5 Parallel I/O

The parallel I/O is implemented with the 8255A-5 programmable peripheral interface (PPI) and connects to the user's interface through the J3 connector. Figure 2-4 illustrates the wirewrap pins used for the configuration of the parallel I/O. (These pins are defined in Table 2-1.) The dash lines in Figure 2-4 represent the factory configuration jumpers installed. (The wirewrap pin locations are shown in Figure 2-1.) The directions of port A and port B transceivers are dependent on the logic level connected to M49 and M52. Wirewrap pin 52 connects to port A through a 200 ns minimum rise time edge delay circuit. When M50 (+3 Vdc) is jumpered to pins M49 and M52, port A and port B buffers are inputs to the PPI from the J3 connector. When M51 (GND) is jumpered to pins M49 and M52, port A and port B buffers are outputs from the PPI to the J3 connector.

The direction of port A and port B can also be controlled by a user's program. To make this possible, M58 and M54 must be jumpered to M49 and M52. The data outputs via port C will control the voltage levels at the direction control inputs to ports A and B. The software required to do this control is discussed in Chapter 6.

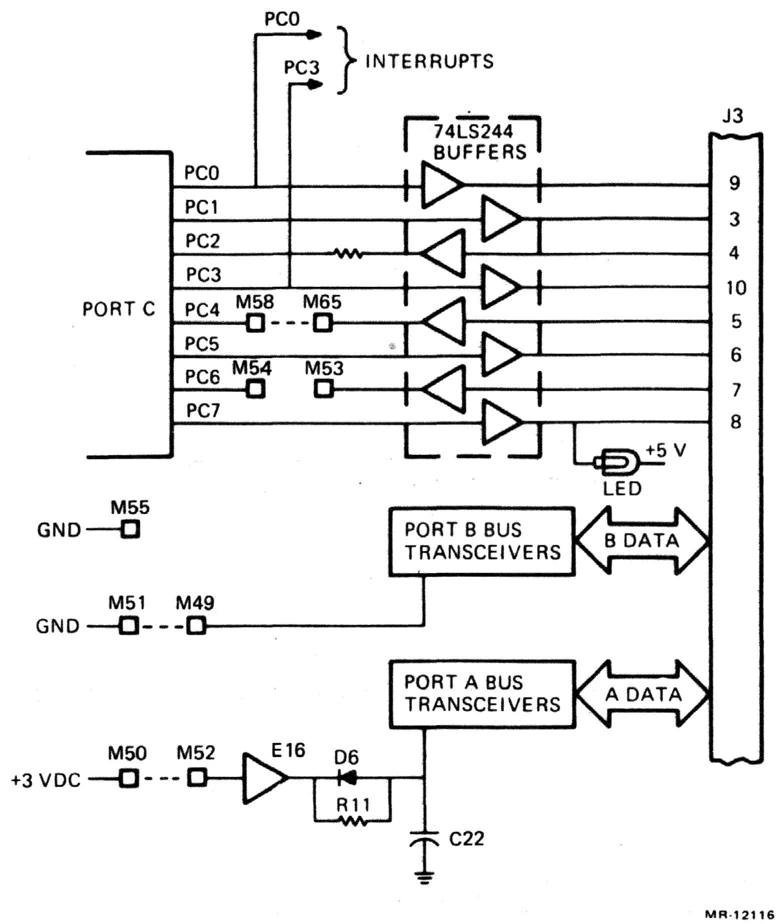


Figure 2-4 Parallel I/O Configuration

Wirewrap pins M65 and M53 can be jumpered to M49 and M52 to allow the user to control the direction of the transceivers via J3 connector pins 5 and 7. When not using wirewrap pins M58 and M65 or M54 and M53 to control the direction of ports A and B, jumpers connected between M58 and M65 and between M54 and M53 allow PC4 and PC6 to be used as inputs to the PPI from the J3 connector.

NOTE

If pins M65, M53, M58, or M54 are used for program control of port A or B, the user must ensure that the PPI and the buffer do not contend as driver output to driver output. If this condition is allowed to occur, damage to both drivers may result.

The programmable peripheral interface can function in three modes selected by software. The jumper configurations and the handshake signals for each of these modes are shown in Table 2-5, Table 2-6, and Table 2-7. See Chapter 6 for programming information.

2.2.6 Serial I/O

The jumper options relating to the serial I/O determine the interrupt response of the system and were explained in Paragraph 2.2.4. All responses to the BREAK detection by SLU1 are listed in Table 2-8.

Table 2-5 Mode 0 Buffer Configuration (No Handshake)

PPI Element	To Act as Input	To Act as Output	Program Control via Port C
Port A	M52 to M50	M52 to M51	M52 to M54 or M58
Port B	M49 to M50	M49 to M51	M49 to M54 or M58
PC7	Never an input	Always an output	
PC6	M54 to M53	Never an external output	
PC5	Never an input	Always an output	
PC4	M58 to M65	Never an external output	
PC3	Never an input	Interrupt A (vector 134) Always an output	
PC2	Always an input	Never an output	
PC1	Never an input	Always an output	
PC0	Never an input	Interrupt B (vector 130) Always an output	

Table 2-6 Mode 1 Buffer Configuration (Strobed I/O)

PPI Element	To Act as Input	To Act as Output	Program Control via Port C
Port A	M52 to M50	M52 to M51	N/A
Port B	M49 to M50	M49 to M51	M49 to M54 or M58
PC7	Never an input	Indicates buffer A full	
PC6	M54 to M53 (Acknowledge A)*	Never an external output	
PC5	Never an input	Indicates buffer A full	
PC4	M58 to M65 (Strobe A)	Never an external output	
PC3	Never an input	Interrupt A	
PC2	Strobe B in input mode Acknowledge B in output mode	Never an output	
PC1	Never an input	Buffer B full on input or output	
PC0	Never an input	Interrupt B (vector 130)	

*User's hardware acknowledges receipt of data output by port A.

Table 2-7 Mode 2 Buffer Configuration and Handshake

PPI Element	Input Signal	Output Signal
Port A	Bidirectional bus	If M52 to M54 to M53
Port B	Not used in mode 2	Not used in mode 2
PC7	Never an input	Output buffer A full
PC6	Acknowledge A	Never an output
PC5	Never an input	Input buffer A full
PC4	Strobe A (if M65 to M58)	Never an output
PC3	Never an input	Interrupt A
PC2	Always an input	Never an output
PC1	Never an input	Always an output
PC0	Never an input	Always an output

Table 2-8 SLU1 BREAK Detection

Jumper Connection*	BREAK Response
M17 to M20 M5 to M8	BHALT L signal to the LSI-11 bus and BKRQ interrupt (vector 140)
M20 to M31 M5 to M8	No response
M8 to M17 M20 to M31	BKRQ interrupt (vector 140) (no BHALT L to bus)
M17 to M74 M20 to M31 M8 to M24	CTMER interrupt (HALT trap) through restart

*Refer to Figure 2-2.

2.2.7 Memories

The memory system for the module is the LSI-11 bus, 4Kb of local RAM, and four 28-pin sockets that accept either 24-pin or 28-pin industry standard +5 V memory chips. These chips are provided by the user and can be either EEPROMs, EPROMs, PROMs, ROMs, or static RAMs. The sockets will accept 2K × 8, 4K × 8, 8K × 8, and 16K × 8 PROMs/EPROMs/EEPROMs, or 2K × 8, and 8K × 8 static RAMs.

There are two socket sets: set A which is controlled by –CSKTA and set B which is controlled by –CSKTB. Each set has a high byte socket and a low byte socket that are interconnected as shown in Figure 2-5. The wirewrap pins used to configure the memory are shown in Figure 2-6 and described in Table 2-1. The standard factory configuration of the installed jumper wires is represented by the dash lines in Figure 2-6. In addition to configuring the sockets, the user must configure the decode memory address chip to select one of the four memory maps available.

NOTE

The SBC-11/21 PLUS contains semiconductor devices that may be susceptible to damage by electrostatic charges. When handling the board and configuring the wirewrap pins, the board should be kept on a grounded conductive plane. Also, wrist straps in contact with the skin should be used to keep the operator at the same ground potential.

2.2.7.1 Memory Maps – Figure 2-7 shows the four memory maps available. The module can be configured to select the one that meets the user's requirements. Wirewrap pins M18, M21, M29, and M25 are used to select the memory map. The jumper requirements are listed in Table 2-9.

2.2.7.2 PROMs/EPROMs/EEPROMs – The 28-pin sockets accept 24-pin and 28-pin PROMs, EPROMs or EEPROMs. If 24-pin chips are selected, caution must be observed to ensure that pin 1 of the chip is placed into socket hole 3. The configuration requirements of some industry compatible PROMs/EPROMs are described in Table 2-10 and Table 2-11. The user may select chips from other vendors, however, the pin configuration must be compatible with the sockets provided. A 250 ns maximum output enable time is also required, and the maximum access time for compatible PROMs/EPROMs is 450 ns. The maximum output enable time is defined as the time from the assertion of TDIN or TDOUT by a bus master to the time the module asserts valid data onto the bus.

The user installs a jumper wire from the pin referenced by the chip type to the socket pin described in the tables. Figure 2-6 provides a reference for all signals and the socket pins associated with the wirewrap pins. These interconnections are listed separately under socket set A and socket set B, and some jumper wires are common to both socket sets. Some devices may not require a connection or installation of a jumper wire and are designated by an NC in the tables. The wirewrap pin locations are shown in Figure 2-1.

2.2.7.3 RAMs – The 28-pin sockets can also accept 24-pin static RAM chips, and caution must be observed to ensure that pin 1 of the chip is installed into socket hole 3. The configuration requirements of some industry compatible RAMs are described in Table 2-12 and Table 2-13. The user may select chips from other vendors, however, the pin configuration must be compatible with the sockets provided. The selected RAMs are required to meet the maximum output enable time and the maximum access time specified for the PROMs.

The user installs a jumper wire from the pin referenced by the chip type to the socket pin described in the tables. Figure 2-6 provides a reference for all signals and the socket pins associated with the wirewrap pins. These interconnections are listed separately under socket set A and socket set B, and some jumper wires are common to both socket sets. Some devices may not require a connection or installation of a jumper wire and are designated by an NC in the tables. The wirewrap pin locations are shown in Figure 2-1.

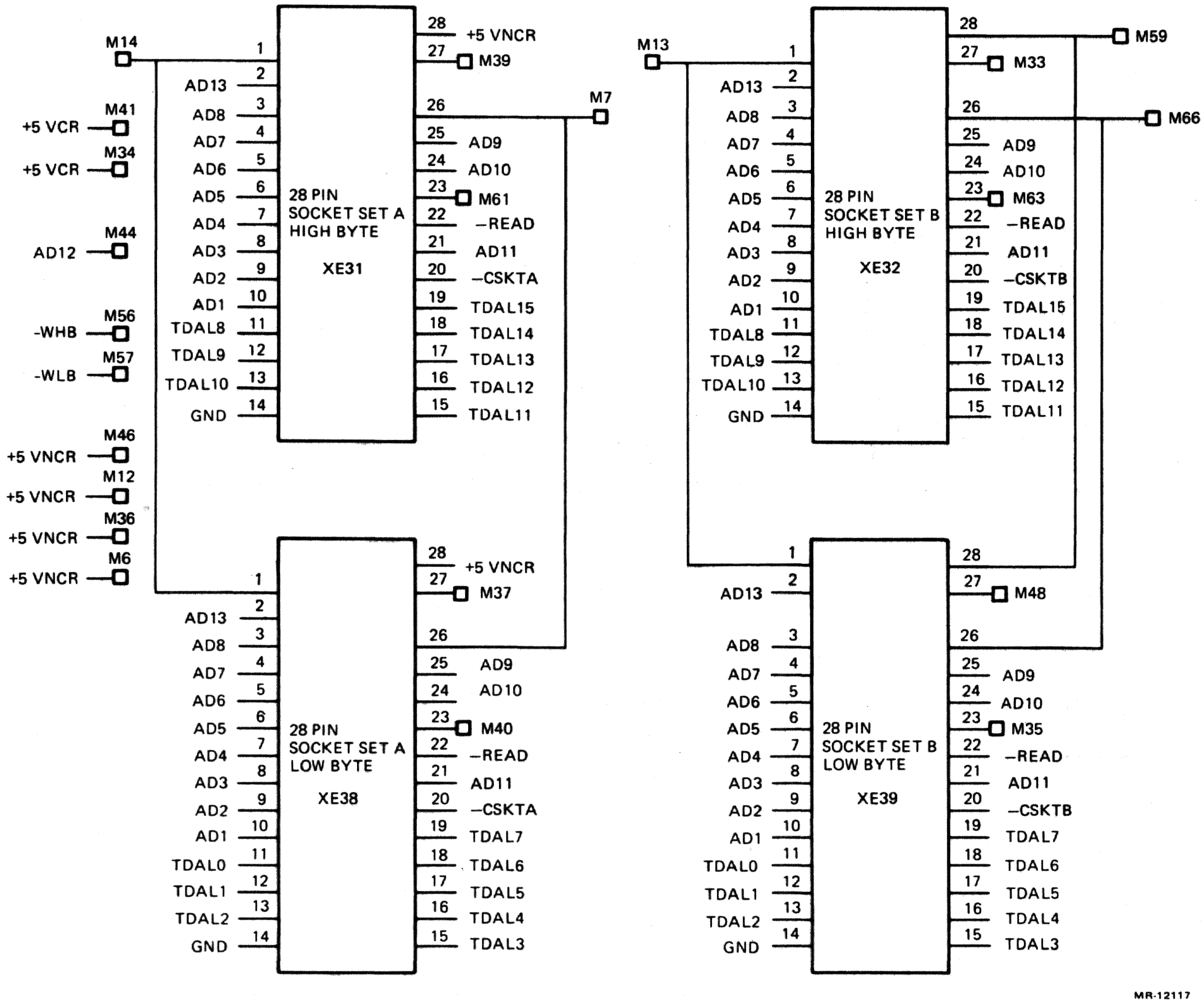


Figure 2-5 Socket Sets A and B Interconnection

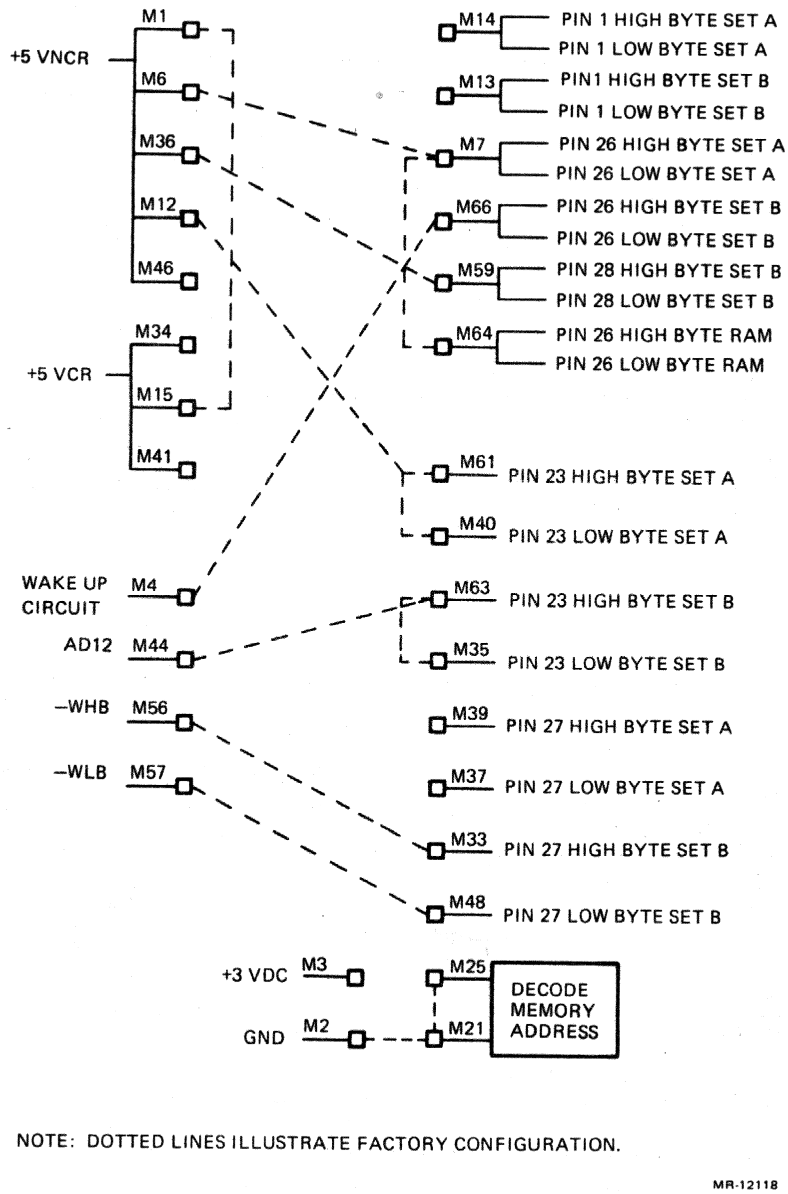
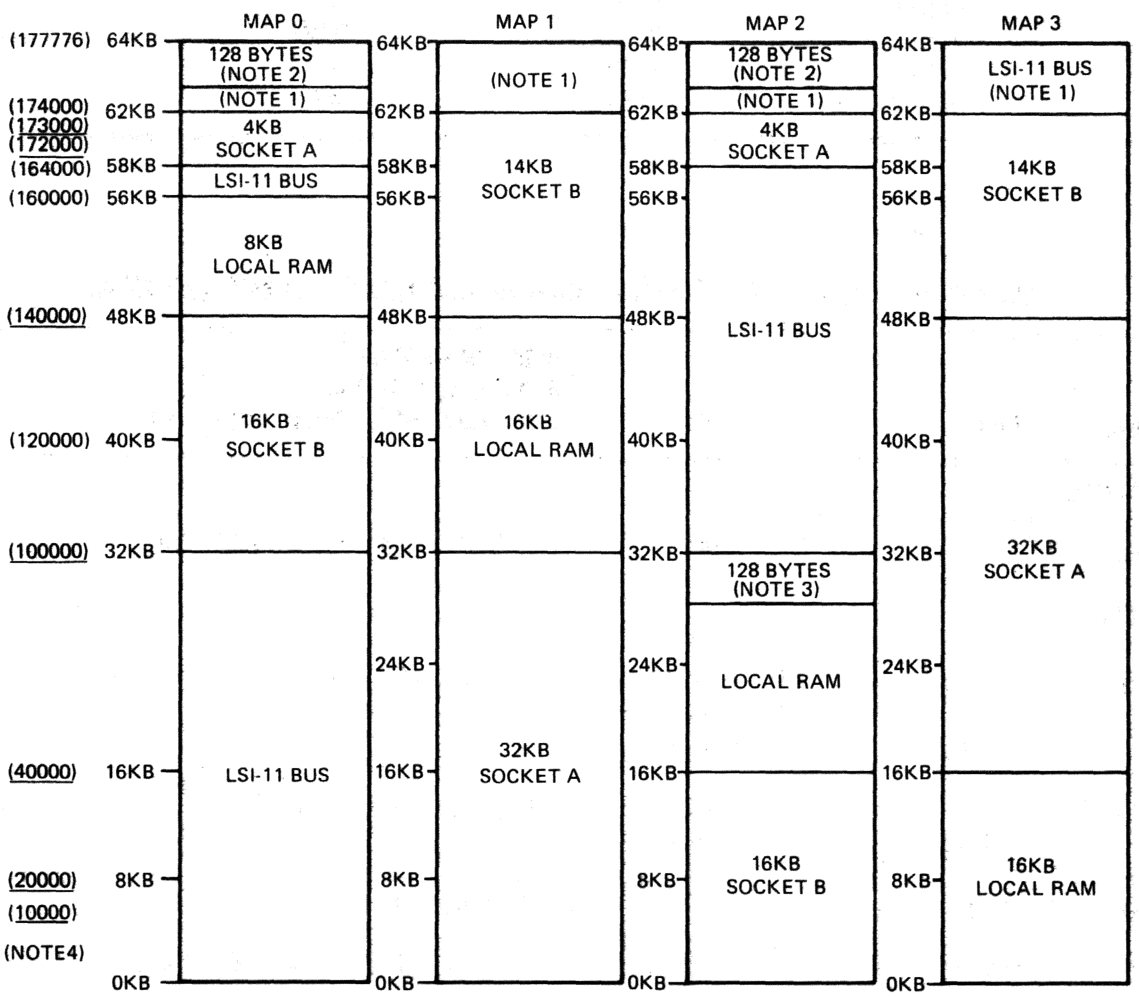


Figure 2-6 Memory Configuration



NOTES:

1. THIS SECTION CONTAINS THE LOCAL I/O ADDRESSES FOR THE SLUS AND PPI. ALL UNASSIGNED ADDRESSES ARE ASSUMED TO RESIDE ON THE LSI-11 BUS.
2. ADDRESSES 177777 - 177600 IN MAPS 0 AND 2 ARE RAM SCRATCHPAD LOCATIONS USED BY MACRO-ODT.
3. ADDRESSES 77777 - 77600 IN MAP 2 ARE ALLOCATED TO THE LSI-11 BUS.
4. UNDERLINED ADDRESSES ARE JUMPER SELECTABLE START ADDRESSES. (SEE TABLE 2-4)

MR-12119

Figure 2-7 Memory Maps

Table 2-9 Memory Map Configurations

Map Selection	Jumper M25 to	Jumper M21 to
Map 0	M21	M29
Map 1	M18	M29
Map 2	M29	M18
Map 3	M21	M18

Table 2-10 Socket Set A Configuration for EEPROMs/EPROMs/PROMs

Vendor	Parts	Pins	Size	Connect Referenced Pin to Socket A Pin M40	M37	M7	M61	M14	M39
EEPROMs									
INTEL	2815	24	2K × 8	M12	NC	M6	M12	NC	NC
EPROMs									
INTEL	2716	24	2K × 8	M12	NC	M6	M12	NC	NC
	2716-1	24	2K × 8						
	2716-2	24	2K × 8						
INTEL	2732	24	4K × 8	M44	NC	M6	M44	NC	NC
	2732A	24	4K × 8						
INTEL	2764	28	8K × 8	M44	M46	NC	M44	M36	M46
INTEL	27128	28	16K × 8	M44	M39	M10	M44	M39	M36
TI	TMS2516	24	2K × 8	M12	NC	M6	M12	NC	NC
	TMS2516-35	24	2K × 8						
Mostek	MK2716	24	2K × 8	M12	NC	M6	M12	NC	NC
Mostek	MK2764	28	8K × 8	M44	NC	NC	M44	M36	NC
PROMs									
INTEL	3632	24	4K × 8	M12	NC	M6	M12	NC	NC
INTEL	3632-1	24	4K × 8	M12	NC	M6	M12	NC	NC

NC – requires no connection.

Table 2-11 Socket Set B Configuration for EEPROMs/EPROMs/PROMs

Vendor	Parts	Pins	Size	Connect Referenced Pin to Socket B Pin M35	M48	M13	M63	M59	M66	M33
EEPROMs										
INTEL	2815	24	2K × 8	M12	NC	NC	M12	NC	M6	NC
EPROMs										
INTEL	2716	24	2K × 8	M12	NC	NC	M12	NC	M6	NC
	2716-1	24	2K × 8							
	2716-2	24	2K × 8							
INTEL	2732	24	4K × 8	M44	NC	NC	M44	NC	M6	NC
	2732A	24	4K × 8							
INTEL	2764	28	8K × 8	M44	M46	M36	M44	M36	NC	M46
INTEL	27128	28	16 × 8	M44	M33	M33	M44	M36	M10	M59
TI	TMS2516	24	2K × 8	M12	NC	NC	M12	NC	M6	NC
	TMS2516-35	24	2K × 8							
Mostek	MK2716	24	2K × 8	M12	NC	NC	M12	NC	M6	NC
Mostek	MK2764	28	8K × 8	M44	NC	M36	M44	M36	NC	NC
PROMs										
INTEL	3632	24	4K × 8	M12	NC	NC	M12	NC	M6	NC
INTEL	3632-1	24	4K × 8	M12	NC	NC	M12	NC	M6	NC

NC – requires no connection.

Table 2-12 Socket Set A Configuration for RAM

Vendor	Parts	Pins	Size	Connect Referenced Pin to Socket A Pin M40	M37	M7	M61	M14	M39
Mostek	MK4802	24	2K × 8	M57	NC	M34	M56	NC	NC
Toshiba	TMM2016P	24	2K × 8	M57	NC	M34	M56	NC	NC
	TMM2016P-1	24	2K × 8						
	TC5565P/P-1	28	8K × 8	M44	M57	M34	M44	NC	M56
	TC5565PL/PL-1								
Hitachi	HM6116P	24	2K × 8	M57	NC	M34	M56	NC	NC
	HM6264P	28	8K × 8	M44	M57	M34	M44	NC	M56

NC – requires no connection.

Table 2-13 Socket Set B Configuration for RAM

Vendor	Parts	Pins	Size	Connect Referenced Pin to Socket B Pin M35	M48	M13	M63	M59	M66	M33
Mostek	MK4802	24	2K × 8	M57	NC	NC	M56	NC	M34	NC
Toshiba	TMM2016P	24	2K × 8	M57	NC	NC	M56	NC	M34	NC
	TMM2016P-1	24	2K × 8							
	TC5565P/P-1	28	8K × 8	M44	M57	NC	M44	M41	M41	M56
	TC5565PL/PL-1									
Hitachi	HM6116P	24	2K × 8	M57	NC	NC	M56	NC	M34	NC

NC – requires no connection.

2.3 SELECTING BACKPLANES AND OPTIONS

A number of different LSI-11 bus compatible backplanes and boxes are available from Digital. The choice is defined by system requirements such as the number and type of options (described in Chapter 3), environment conditions, and packaging considerations. A list of all available backplanes and boxes is provided in the *Microcomputer Interfaces Handbook*.

2.4 POWER SUPPLY

The choice of power supply is controlled by the size of the system and packaging requirements. An important consideration is the performance of the supply during power-up and power-down. All Digital power supplies listed in the *Microcomputer Interfaces Handbook* are compatible with the LSI-11 bus protocol which allows dependable operation with no loss of data when using battery backed-up memories. Any user designed power supply must agree with the LSI-11 bus protocol.

2.5 EXTERNAL CABLES

The module has a 30-pin connector (J3) for an external interface with the programmable I/O interface and two 10-pin connectors (J1 and J2) for the external interface of the serial line units (SLUs). The location of these connectors on the module is shown in Figure 2-1. The requirements to interface with these connectors are defined in the following paragraphs.

2.5.1 Parallel I/O Interface (J3)

The module connector is a 30-pin AMP MODU connector with the I/O signals defined by Figure 2-8. The I/O signals are buffered and are capable of driving up to 50 feet (maximum) of flat ribbon or round cable with a 30-pin AMP contact housing at each end. The following list of connectors is compatible with the module connector.

AMP MODU polarized or nonpolarized contact housings for crimp snap-in pin and receptacle contacts:

Latching, polarized housings:	2-87631-6 no strain relief 87733-6 strain relief
Nonlatching, polarized housings:	1-87977-3 no strain relief 1-102184-3 strain relief
Nonlatching, nonpolarized housings:	2-87456-6 no strain relief 2-87832-7 strain relief
Receptacle contacts:	87045-3 for 30 to 26 AWG 102098-3 for 32 to 27 AWG

Mass termination connectors for flat cables:

Separate parts: (nonpolarized)	1-88378-1 connector 1-86873-2 cover 1-88340-1 strain relief cover
Separate parts: (polarized)	1-88392-1 connector 1-86373-2 cover 1-88340-1 strain relief cover
Connector and cover kits: (nonpolarized)	1-88379-1 no strain relief 1-88476-1 with strain relief
Connector and cover kits: (polarized)	1-88393-1 no strain relief 1-88478-1 with strain relief
Separate parts:	1-88392-1 connector 1-86873-2 cover 1-88340-1 strain relief cover
Latching connectors and covers: (polarized)	1-88423-1 no strain relief 1-88479-1 with strain relief
Mass modular connector system:	1-102393-3 housing for 30-26 AWG 1-102396-3 cover 1-102392-3 kit 1-102398-3 housing for 26-22 AWG 1-102396-3 cover 1-102397-3 kit

Connectors can be terminated to discrete wire in sizes 30-26 AWG, 26-24 AWG, as well as jacketed cable and bonded ribbon cable.

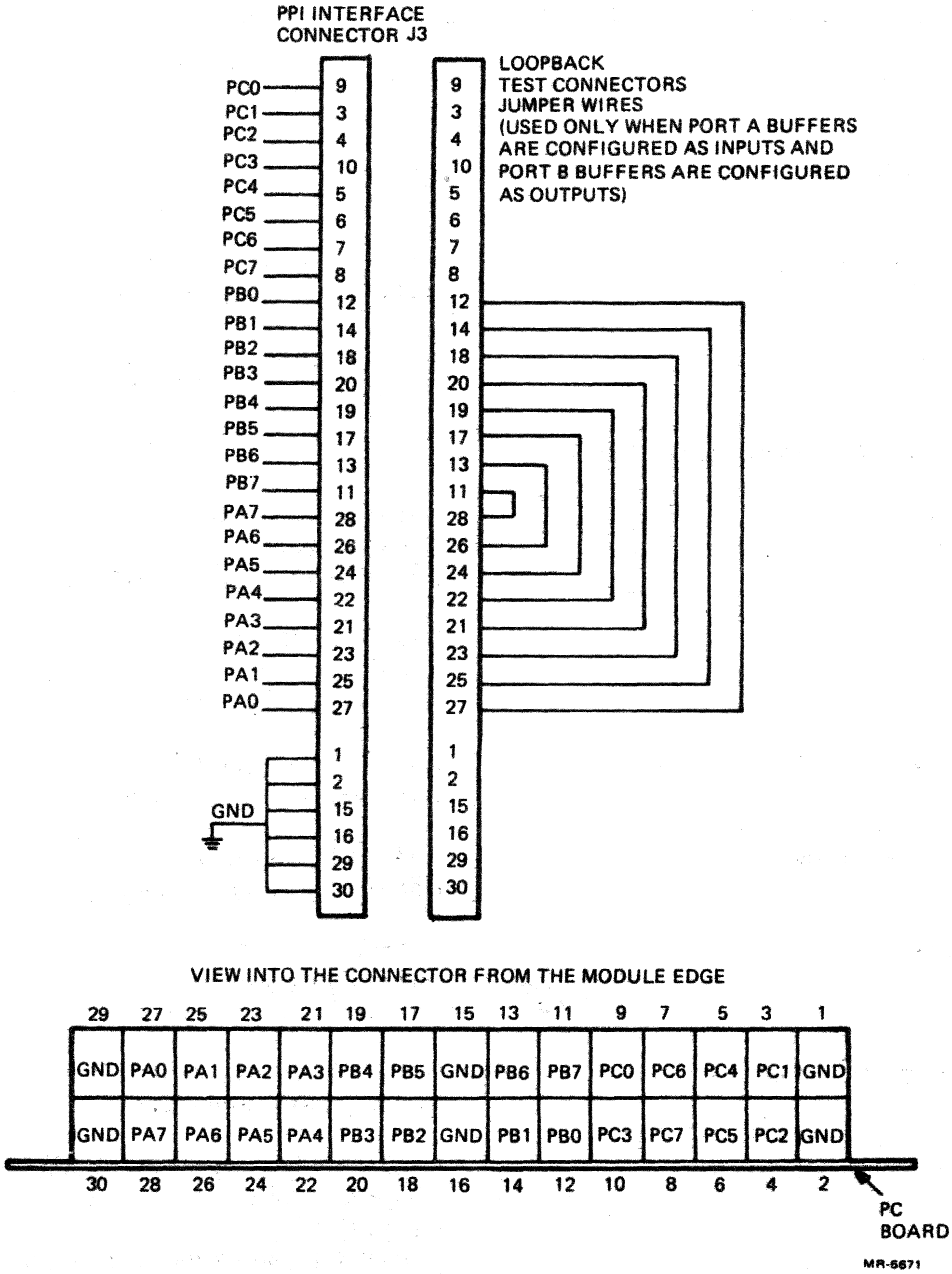


Figure 2-8 30-Pin Parallel I/O Connector

2.5.2 Serial Line Interfaces (J1 and J2)

Each serial line unit (SLU) is compatible with EIA RS-232C and EIA RS-423 serial type interfaces. SLU1 interfaces through J1, and SLU2 interfaces through J2. When a 20 mA current loop device is needed, the DLV11-KA option must be used. The option has an EIA cable (BC21A-03) that connects the converter box to the module. The box has an 8-pin Mate-N-Lok™ connector that mates with the standard 20 mA cable. The option does not support the reader run strobe and the 110 baud rate and therefore, the LA-33 or similar devices cannot be used.

The user installs a slew rate resistor determined by the operating baud rate defined in Table 2-14. The slew rate resistor is identified as R6 and its location on the module is shown in Figure 1-1.

The SLU connectors showing the signals assigned to the connector pins are illustrated in Figure 2-9. The user provides the interconnecting cables. The following list describes some standard Digital cables and also provides some information to help the user design cables.

Digital cables for the SBC-11/21 PLUS:

BC20N-05	5-foot EIA RS-232C null modem cable to directly interface with the EIA RS-232C terminal (2 × 5 pin AMP female to RS-232C female; see Figure 2-10).
BC21B-05	5-foot EIA RS-232C modem cable to interface with modems and acoustic couplers (2 × 5 pin AMP female to RS-232C male; see Figure 2-11).
BC20M-50	50-foot EIA RS-422 or RS-423 cable for high throughput transmission (19.2K baud) between two SBC-11/21 PLUS computers (2 × 5 pin AMP female to 2 × 5 pin AMP female).

When designing a cable for the SBC-11/21 PLUS, the user should consider the following points:

1. The receivers on the SBC-11/21 PLUS have differential inputs. Therefore, when designing an RS-232C or RS-423 cable, RECEIVE DATA− (pin 7 on the 2 × 5 pin AMP connector) must be tied to signal ground (pins 2, 5, or 9) in order to maintain correct EIA levels. RS-422 uses both RECEIVE DATA+ and RECEIVE DATA−.

2. To directly connect to a local EIA RS-232C terminal, it is necessary to use a null modem. To design the null modem into the cable, a user must switch RECEIVE DATA (pin 2) with TRANSMITTED DATA (pin 3) on the RS-232C male connector as shown in Figure 2-10.

3. To mate to the 2 × 5 pin connector block, the following parts are needed.

Cable receptacle	AMP PN 87133-5 DEC PN 12-14268-02
Locking clip contacts	AMP PN 87124-1 DEC PN 12-14267-00
Key pin (pin 6)	AMP PN 87179-1 DEC PN 12-15418-00

Mate-N-Lok™ is a trademark of AMP, Inc.

Table 2-14 EIA Slew Rate Resistor Values

Baud Rate	Resistor R6 (ohms)
38,400	22 kΩ*
19,200	51 kΩ
9,600	120 kΩ**
4,800	200 kΩ
2,400	430 kΩ
1,200	820 kΩ
600	1 MΩ
300	1 MΩ

*Factory installed value.
**Maximum baud rate for SLU1.

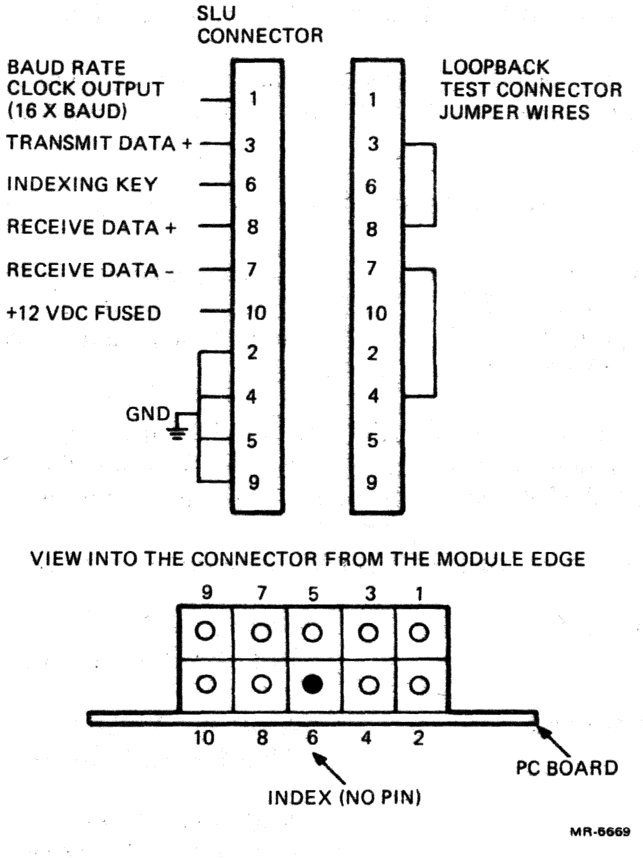


Figure 2-9 10-Pin Serial Line Unit Connector

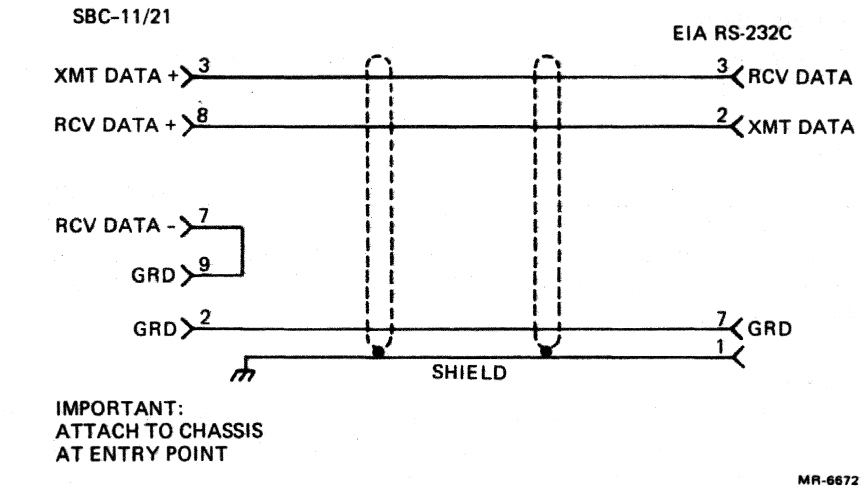


Figure 2-10 BC20N-05 Null Modem Cable

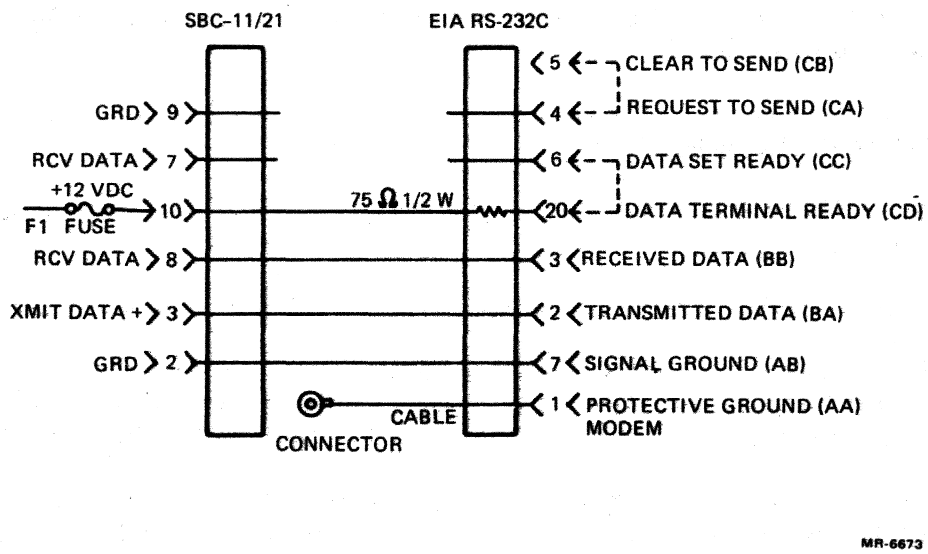


Figure 2-11 BC21B-05 Modem Cable

2.6 VERIFYING OPERATION

The SBC-11/21 PLUS single-board computer can be field tested to verify its functional operation. The Macro-ODT option and the loopback connectors support the testing of the module.

2.6.1 Macro-ODT Option

The Macro-ODT option (part number KXT11-A5) has two 24-pin, 2K × 8 PROM chips that contain the Macro-ODT code and module diagnostic programs. The Macro-ODT code is used to create communication between the module and the user via console commands. The use of ODT commands is detailed in Chapter 4. The module diagnostic programs verify that the parallel I/O and serial line unit interfaces will function with commands from the microprocessor.

2.6.2 Loopback Connectors

The loopback connectors can be made by the user for the module diagnostic tests. The 30-pin connector with the loopback jumper wires installed as shown in Figure 2-8, is used with the parallel I/O connector J3. The serial line unit connector with the loopback jumper wires installed (shown in Figure 2-9) is used with the SLU2 connector J2.

2.6.3 Verification Procedure

The module must be restored to the standard factory configuration for the test to be valid, however, the start address must be 172000 not 10000. The module can be verified using the following procedure:

1. Set the start address to 172000 as shown in Table 2-4.

2. Insert the high byte ODT ROM into socket set A, high byte socket XE31. Make sure pin 1 is inserted into socket hole 3.

3. Insert the low byte ODT ROM into socket set A, low byte socket XE38. Make sure pin 1 is inserted into socket hole 3.

4. Insert the 30-pin loopback connector (see Paragraph 2.6.2) into the module parallel I/O connector J3.

5. Insert the 10-pin loopback connector (see Paragraph 2.6.2) into the SLU2 connector J2.

6. Install the module into the LSI-11 backplane with the power turned off. An external power supply may be used to provide +5 Vdc to finger pins BV1, BA2, and AA2, +12 Vdc to finger pin AD2, and ground to finger pins BJ1, AJ1, AT1, AC2, BC2, AM1, and BM1.

7. Connect an external terminal (printer or video). The terminal must be capable of generating a 7-bit ASCII code with odd parity or 8-bit ASCII code with no parity, and baud rates of 300, 600, 1,200, 2,400, 4,800, or 9,600. The terminal is connected to SLU1 connector J1 using a Digital BC20N-05 cable or equivalent. Turn the terminal on and on-line.

8. Turn on the backplane power or enable the +5 Vdc and +12 Vdc sources. Monitor the module LED; it should light and then return to the normal off state. If the LED stays lit, there is a fault in the SLU1 circuits or the on-board RAM memory.

9. After the backplane power is turned on, press the RETURN key (carriage return) on the terminal to have the module synchronize its baud rate to that of the terminal. The module responds with the prompt character '@'.

10. To start the module diagnostic programs press the 'X' key. The diagnostic test will exercise the module including the parallel I/O and SLU2. The results of the test are printed out on the terminal. The error results are listed in Table 2-15 and indicate what area of the module contains a fault. The error code '000000' indicates a good module.

Table 2-15 Diagnostic Fault Indicators

Printout	Parallel I/O Loopback Test	Internal Serial* I/O Loopback Test	External Serial** I/O Loopback Test
000000	Passed	Passed	Passed
000001	Failed	Passed	Passed
000010	Passed	Failed	Not performed
000011	Failed	Failed	Not performed
000100	Passed	Passed	Failed
000101	Failed	Passed	Failed
000110	Not used	Not used	Not used
000111	Not used	Not used	Not used

*The internal serial I/O loopback test exercises the parallel-to-serial conversion, the serial-to-parallel conversion, and the baud rate. This test can be performed without the loopback connector.

**The external serial I/O loopback test exercises the above functions as well as the drivers, receivers, and the external signal paths.

CHAPTER 3
OPTIONS

3.1 INTRODUCTION

The SBC-11/21 PLUS is a complete single-board microcomputer that operates on the LSI-11 bus or in a standalone configuration. In some applications, it may be desirable to add optional modules to the SBC-11/21 PLUS to extend its function beyond that provided by the module itself. Paragraphs 3.2 and 3.3 list all options available. For more information see the documents listed in Paragraph 1.4 of this guide.

3.2 SUPPORTED OPTIONS

The following options are functionally compatible with the SBC-11/21 PLUS. Software diagnostics for these options run on a SBC-11/21 PLUS equipped with a mass storage device (TU58, RX01, RX02, RX50, RD51, RL01, and RL02) and the Macro-ODT option. To order diagnostics contact your Digital sales representative.

3.2.1 Hardware Options

TU58 — The TU58 is a low-cost mass memory device that is used with the SBC-11/21 PLUS by attaching it to one of the serial I/O lines. TU58 offers random access to block-formatted data on pocket size cassette media. It is ideal as a small computer systems device, as inexpensive archive mass storage, or as a software update distribution medium. A dual drive TU58 offers 512Kb of storage space, making it one of the lowest cost complete mass storage subsystems available. For mounting flexibility, the TU58 is offered both as a component level subsystem and as a fully powered 5-1/2 inch rack-mount subsystem. The TU58 interfaces with the microprocessor over an RS-423 serial line interface.

AAV11-C — The AAV11-C is a 4-channel, 12-bit digital-to-analog converter module that includes control and interfacing circuits. It has four D/A converters, a dc-dc converter that provides power to the analog circuits, and a precision voltage reference. Each channel has its own holding register that can be addressed separately and provides 12 bits of resolution. Bits 0, 1, 2, and 3 of the fourth holding register are brought out to the I/O connector so that they can be used as a 4-bit digital output register.

ADV11-C — The ADV11-C is a 12-bit successive approximation analog-to-digital converter that samples analog data at specified rates and stores the digital equivalent value for processing. The multiplexer can accommodate up to sixteen single-ended or eight quasi-differential inputs. The converter uses a patented auto-zeroing design that measures the sampled data with respect to its own offset and therefore, cancels out its own offset error.

Three reference signals are provided for self-testing any channel input. These signals consist of two dc levels and one bipolar triangular waveform. This output can be used with Digital diagnostic software to produce a data base for precise analog linearity testing.

AXV11-C	The AXV11-C functions like the ADV11-C, but also has two 12-bit digital-to-analog converters similar to those on the AAV11-C module.
DEQNA	The DEQNA is a dual height module which interfaces the LSI-11 bus to ETHERNET.
DHV11	The DHV11 option is an asynchronous multiplexer which provides eight full-duplex, asynchronous, serial data channels on an LSI-11 or Q22 bus. The option can be used in several applications which include data concentration, real-time processing, and cluster controlling.
DLV11-E	The DLV11-E is an asynchronous line interface module that interconnects the LSI-11 bus to standard serial communications lines. The module receives serial data, converts it to parallel data, and transfers it to the LSI-11 bus. It also accepts parallel data from the LSI-11 bus, converts it to serial data, and transmits it to the peripheral device. The module has jumper selectable or software selectable baud rates (50–19,200) and jumper selectable data bit formats. The DLV11-E offers full modem control for EIA/CCITT interfaces.
DLV11-F	The DLV11-F is an asynchronous line interface module that interconnects the LSI-11 bus to several types of standard serial communications lines. The module receives serial data, converts it to parallel data, and transfers it to the LSI-11 bus. It also accepts parallel data from the LSI-11 bus, converts it to serial data, and transmits it to the peripheral device. The module has jumper selectable or software selectable baud rates (50–19,200) and jumper selectable data bits. The DLV11-F supports either 20 mA current loop devices or EIA standard lines, but does not include modem control.
DLV11-J	The DLV11-J contains four independent asynchronous serial line channels that are used to interface peripheral devices to the LSI-11 bus. Each channel transmits and receives data from the peripheral device over EIA data leads (lines that do not use a control line). The module can be used with 20 mA current loop devices if a DLV11-KA adapter is used. The DLV11-J has jumper selectable baud rates from 150 to 39.4K baud.
DMV11	The multipoint DDCMP-DMV11 intelligent communications synchronous line controller is an interface device which provides efficient high-speed synchronous communications for distributed networks. The DMV11 uses LSI-11 CPUs as control or tributary stations, while requiring a minimum of main CPU resources. The following provides detailed information on the installation and operation of the DMV11.
DPV11-DA	The DPV11-DA is a single line, program-controlled, double-buffered communication device designed to interface the LSI-11 bus to a serial synchronous line. This self-contained unit can use a wide range of protocols including bit-oriented protocols (SDLC, HDLC, ADCCP, and X.25) and byte-oriented protocols (DDCMP and BISYNC). The module is used for high-speed synchronous lines such as remote batch, remote data collection, remote concentration, and communication networking. The module, compatible with EIA RS-232 and CCITT V.28 interface standards, is also compatible with EIA RS-423 and 422 electrical standards and thus, provides low-cost, local communications capability.
DRV11	The DRV11 is a parallel interface module used to interconnect the LSI-11 bus with general-purpose parallel line TTL or DTL devices. It allows program-controlled data

transfers at rates up to 40K words per second and uses LSI-11 bus interface and control logic to generate interrupts and process vector handling. The data is handled by sixteen diode clamped input lines and sixteen latched output lines. There are two 40-pin connectors on the module for user interface applications.

DRV11-B

The DRV11-B is an interface module that uses direct memory access (DMA) to transfer data directly between the system memory and an I/O device. The interface is programmed by the processor to move variable length blocks of 8-bit or 16-bit data words to or from specified locations in the system memory. Once programmed, no processor interrupts are required. The module can transfer up to 250K 16-bit words per second in the single cycle mode and up to 500K 16-bit words per second in the burst mode. The module also allows read-modify-restore operations.

DRV11-J

The DRV11-J provides sixty-four input/outut data lines on a double-height module for the LSI-11 bus. The DRV11-J also includes an advanced interrupt structure with bit interruptability up to sixteen lines, programmable interrupt vectors, and program selection of fixed or rotating interrupt priority within the DRV11-J. The DRV11-J bit interrupts for real-time response make it especially useful for sensor I/O applications. It can also be used as a general-purpose interface to special devices, and two DRV11-Js can be connected back-to-back as a link between two LSI-11 buses.

DUV11-DA

The DUV11-DA synchronous line interface module creates a data communication line between the LSI-11 bus and a Bell 201 synchronous modem or equivalent. The module is programmable to sync characters, character length (up to 8 bits), and parity selection. The receiver logic accepts serial data for the LSI-11 bus. The transmitter logic converts the parallel LSI-11 bus data into serial data for the transmission line. The interface logic converts the TTL logic levels to the EIA voltage levels needed by the Bell 201 modems and also controls the modem for half-duplex or full-duplex operation.

DZV11-B

The DZV11-B is an asynchronous multiplexer interface module that interconnects the LSI-11 bus with up to four asynchronous serial data communications channels. The module provides EIA interface voltage levels and data set control to permit dial-up (auto answer) options with full-duplex modems such as Bell models 103, 113, 212, or equivalent. The DZV11-B does not support half-duplex operations or the secondary transmit and receive operations that are available with some modems such as Bell 202. The module has applications in data storage and collection systems where front-end systems interface to a host computer and for use in a cluster controller for terminal applications.

IBV11-A

The IBV11-A is an interface module that interconnects the LSI-11 bus with the device bus described in IEEE standard 488 1975, *Digital Interface for Programmable Instrumentation*. The IBV11-A makes a processor-controlled programmable device system possible. The module can accommodate up to fifteen IEEE-488 devices.

KMV11

The KMV11 is a medium speed intelligent single line data communications interface for LSI-11 bus based systems.

KXT11-CA

The KXT11-CA implements the LSI-11 bus (Q-Bus) specifications as a bus slave (it is incapable of arbitrating the bus), and as a DMA device (can become bus master to transfer data when the arbiter grants the bus). In this mode it serves as an intelligent peripheral or as an I/O processor module (IOP).

This module can also be used as a single-board computer (SBC) in a standalone fashion.

KWV11-C

The KWV11-C is a programmable real-time clock/counter that provides a means of determining time intervals or counting events. It can be used to generate interrupts to the processor at predetermined intervals or to establish timing between input and output events. It can also initialize the ADV11-C analog-to-digital converter by a clock counter overflow or by firing a Schmitt trigger. The clock counter has a resolution of 16 bits and can be driven by any one of five crystal-controlled frequencies (100 Hz to 1 MHz), from a line frequency input, or from a Schmitt trigger fired by an external input. The module can operate in any of four programmable modes: single interval, repeated interval, external event timing, and external event timing from zero base.

MCV11-D

The MCV11-D is an on-board battery backed CMOS memory that supports 22-bit addressing. The MCV11-DA is an 8Kb module, and the MCV11-DC is a 32Kb module. The module incorporates two nickel cadmium batteries for backup in case of a power failure.

MRV11-C

The MRV11-C is a flexible, high density ROM module used with the LSI-11 bus. The module contains sixteen 24-pin sockets which accept many of the user supplied ROM chips. The module accepts masked ROMs, fusible link PROMs, and ultraviolet erasable PROMs. It accepts several densities of ROM chips up to and including 4K × 8 chips. Using these high density chips gives the module a total capacity of 64Kb. The contents of the module can be accessed directly or window-mapped. Direct access provides total random access to all ROM locations on the module. Window-mapping provides two 2Kb windows of memory address space to access 2Kb segments of the ROM array. The segments that are seen through each window can be changed by program control.

MRV11-D

The MRV11-D is a dual height module containing sixteen 28 pin sockets. It performs with full capability on a 16-, 18- or 22-bit bus. Although this module is basically a ROM board with bootstrap capability, it supports many combinations of 24 and 28 pin ROM, PROM, EPROM, EEPROM, and SRAM devices. The MRV11-D offers page mode addressing, as well as direct mode addressing.

MSV11-D

The MSV11-D module has either 8K, 16K, or 32K × 16 bits of MOS memory. The module has an on-board memory refresh and performs the necessary LSI-11 bus cycles. The memory addressing is selected by the user by configuring switch positions. The module can use a battery backup system to maintain data when primary power is lost.

MXV11-A

The MXV11-A is a dual-height multifunction option module for the LSI-11 bus. It contains a read/write memory, provisions for read only memory, two asynchronous serial line interfaces, and a 60 Hz clock signal derived from a crystal oscillator. Read/write memory is provided with either 8Kb or 32Kb (4K or 16K words). Two 24-pin sockets are provided for +5 V read only memories. 1K × 8, 2K × 8, or 4K × 8 ROMs may be used. The sockets may also be used for 256 words of bootstrap code. The two asynchronous serial lines transmit and receive EIA-423 signal levels from 150 baud to 38.4K baud. Twenty mA active or passive current loop operation at 110 baud may be used with the DLV11-KA EIA to 20 mA converter option. The serial lines will not support the reader run function of the DLV11-KA option. The serial lines provide error indicator bits for overrun error, frame error, and parity error, but do not have modem controls. Serial line 1 may be configured to respond to a BREAK signal. The serial lines have signal level interrupt logic. Serial line 1 and serial line 0 may be used with any of many standard types of serial communication devices. The 60 Hz clock signal can be selected by a wirewrap jumper to provide real-time clock interrupts on the bus.

RL01/RL02 — The RL01 and the RL02 are random-access, mass storage subsystems that store data in fixed-length blocks on a preformatted disk cartridge. Each RL01 can store 5.24 million bytes, and each RL02 can store 10.48 million bytes.

The RLV12 disk controller interfaces RL02 and RL01 disk drives to any quad-or hex-size backplane that uses 16-, 18-, or 22-bit LSI-11 bus. One RLV12 controls up to four disk drives. The RLV12 consists of one quad-size module, a BC80M cable, a drive terminator, and drive identification hardware.

The RLV12 transfers data to and from the LSI-11 bus using direct memory access (DMA) transactions. This allows data transfers to occur without processor intervention.

RD51/RX50 — The RD51 is a 10Mbytes hard disk mass storage device and the RX50 is an 800Kb floppy disk mass storage device. Both are interfaced to the LSI-11 bus by means of an RQDX1 controller. The RQDX1 controller (M8639) is a quad-height module that is mounted in the last occupied slot in the backplane. It is a high-functionality disk controller that interfaces the RD51 fixed disk drive and the RX50 diskette drive to the extended LSI-11 bus. The controller is a direct memory access (DMA) type of interface and conforms to mass storage control protocol (MSCP). A cable (part number BC06L-1C) connects the RQDX1 controller module to the signal distribution printed circuit board, where the signals are distributed to the individual drives.

RXV11 — The RXV11 option has an interface module, cable assembly, and either a single or dual drive RX01 floppy disk. This option is a random access, mass storage device that stores data in fixed-length blocks on a preformatted flexible diskette. Each diskette can store, recover, and retrieve up to 256K, 8-bit bytes of data. The RXV11 system is rack mountable in the standard 48.3 cm (19 in) cabinet.

RXV21 — The RXV21 floppy disk option is a random access mass memory device that stores data in fixed-length blocks on a preformatted, flexible diskette. Each diskette can store and recover up to 512K 8-bit bytes of data. The RXV21 system is rack mountable and consists of an interface module, an interface cable, and either a single or dual RX02 floppy disk drive. The interface module converts the RX02 I/O bus to the LSI-11 bus structure. It controls the RX02 interrupts to the processor, decodes device addresses for register selection, and handles the data exchange between the RX02 and the processor via DMA transfers. Power for the interface module is provided by the LSI-11 bus.

TSV05 — The TSV05 is a 1600 BPI magnetic tape drive.

3.2.2 Software Options

The SBC-11/21 PLUS is functionally compatable with the following software options.

3.2.2.1 RT-11 Operating System – The RT-11 (Real Time-11) computer system is a single-user computer/operating system that serves the programming needs of both the beginning and the advanced programmer. RT-11 supports a number of programming languages, including industry standard FORTRAN, BASIC, and for more advanced users the PDP-11 assembly language, MACRO-11. RT-11 also provides a comprehensive set of operating commands for controlling system operations. Both the single job (SJ) and the foreground background (FB) monitors are supported by the SBC-11/21 PLUS.

3.2.2.2 MicroPower/Pascal Operating System – MicroPower/Pascal is a software product for developing dedicated, real-time applications that run on Digital's 16-bit microcomputers. MicroPower/Pascal applications are created in a host development/target runtime environment and are ROMmable. The host is either a PDP-11 running RT-11 or RSX-11M, or a VAX running VMS. The target is any PDP/LSI-11 Q-Bus based system. MicroPower/Pascal consists of an extended PASCAL compiler, a modular, standalone runtime system (kernel, language features, device handlers, and system processes), build and load utilities, and a symbolic debugger.

3.3 UNSUPPORTED OPTIONS

A list of LSI-11 bus options that are not guaranteed to be functionally compatible with the SBC-11/21 PLUS and are unsupported are listed below. Their diagnostics are not available.

AAV11-A	KXT11-AA
ADV11-A	LAV11
BDV11-AA/BA	LPV11
DA11-MS/QQ/QU	MRV11-AA/BA/VA
DAV11-A/B	MSV11-E/P
DRL11-SN	MSV11-L
DUV11-E/F	MXV11-BF
DUV25	NCV11-A
DW11	REV11
DWV11-A	RKV11
FEPTC-BA	RLV11
FPF11	TEV11
IPV12	TRV11
KD11-F	TSV11
KD11-HA	VMV11-A
KDF11-AB/AC/BB	VK170
KDF11-BC/P	VSV11
KDJ11-A	VTV01-A
KPV11-A	VTV30-H

CHAPTER 4
MACRO-ODT

4.1 INTRODUCTION
The Macro-ODT is the KXT11-A5 option available to users of the SBC-11/21 PLUS single-board computer. The option has a complete listing of the firmware and two 24-pin, 2K × 8 PROM chips that contain the Macro-ODT firmware. The chips are installed on the module using the PROM sockets.

Macro-ODT allows the user to:

1. Examine and deposit data in memory or general registers.
2. Examine or change the processor status word (PSW).
3. Start the execution of the program.
4. Restart the execution of a halted program.
5. Bootstrap programs from a mass storage device (TU58 cassette, RX01/RX02 floppy disk, RD51 disk, RX50 floppy disk, or RL01/RL02 disk).
6. Run a diagnostic test for on-board devices.

4.2 INSTALLATION AND CONFIGURATION
The installation and configuration of the KXT11-A5 option is described in detail in Chapter 2 of this User's Guide, and the user should refer to it for installation and start-up instructions.

4.3 ENTRY CONDITIONS
Macro-ODT is entered:

1. On power-up.
2. Via the BREAK key on the console terminal.
3. On execution of a HALT instruction.
4. On assertion of the BHALT L signal on the LSI-11 bus.
5. When accessing nonexistent memory (i.e., a bus time-out).

4.3.1 Macro-ODT Input Sequence
When entering Macro-ODT, the RBUF register is read using a DATI, and the character present in the buffer is ignored. This is done so that erroneous characters or user program characters are not interpreted by Macro-ODT as commands.

The input sequence for Macro-ODT follows.

1. Read and ignore character in RBUF.

2. Output a <CR> <LF> to the terminal.

3. Output contents of PC (program counter R7) in six digits to terminal if ODT is entered via a BREAK, BHALT, or HALT instruction or trying to fetch an instruction from nonexistent memory. Output a '?' to the terminal if ODT is entered via a bus time-out.

4. Output a <CR> <LF> to the terminal.

5. Output the prompt character (@) to the terminal.

6. Enter a wait loop for terminal input. The done flag, bit 7 in RCSR, is tested using a DATI. If it is zero, the test continues.

7. If RCSR bit 7 is a one, the low byte of RBUF is read using a DATI.

4.3.2 Macro-ODT Output Sequence

The output sequence for ODT follows.

1. Test XCSR bit 7 (done flag) using a DATI and, if it is a zero, continue testing.

2. If XCSR bit 7 is a one, write character to low byte of XBUF using a DATI followed by a DATO (high byte is ignored by interface).

4.4 MACRO-ODT COMMANDS

Table 4-1 lists the Macro-ODT commands. The commands are a subset of ODT-11 and use the same command character. The Macro-ODT internal states are listed in Table 4-2. Only specific characters are recognized as valid inputs for each state; other inputs produce a '?' response.

The parity bit, bit 7, on all input characters is ignored by Macro-ODT, and if the input character is echoed, the state of the parity is copied to the output buffer (XBUF). Output characters internally generated by ODT (e.g., <CR>) have the parity bit equal to zero. All input characters are echoed. Only uppercase command characters are recognized.

NOTE
The use of ODT commands creates a dialogue between the user and the microcomputer. All the characters typed by the user are underlined and the system response is not underlined in the examples given in this User's Guide.

4.4.1 / (ASCII 057) Slash

The '/' command is used to open an on-board module address, LSI-11 bus address, processor register, or processor status word and must normally be preceded by other characters that specify a location. In response to '/', Macro-ODT prints the contents of the location (i.e., six characters) and a space (ASCII 40). After printing is complete, Macro-ODT waits for either new data for that location or a valid close command (<CR> or <LF> if memory is accessed). The space character is issued so that the location's contents and possible new contents entered by the user are legible on the terminal.

Example: @001000/12525<SPACE>

where:

@	=	Macro-ODT prompt character.
001000	=	octal location in the LSI-11 bus address space wanted by the user (leading zeros are not required).
/	=	command to open and print contents of location.
012525	=	contents of octal location 1000.
<SPACE>	=	space character generated by Macro-ODT.

If the user issues a '/' immediately after a prompt character, the system prints ? <CR> <LF> because a location is not open.

Table 4-1 Macro-ODT Commands

Command	Symbol	Function
Slash	/	Prints the contents of a specified location.
Carriage return	<CR>	Closes an open location.
Line feed	<LF>	Closes an open location and opens the next location. This command cannot be used with the general registers.
Internal register designator	Rx	Opens a specific processor register. (x = 0–7,S)
Processor status word designator	S	Opens the PSW; must follow R command.
Go	G	Starts the execution of a program.
Proceed	P	Resumes the execution of a program.
Boot from device	D	Loads and runs programs from floppy diskettes or TU58 cassettes.
Execute diagnostics	X	Runs SBC-11/21 PLUS module verification diagnostics.

Table 4-2 Macro-ODT States and Valid Input Characters

State	Example of Terminal Output	Valid Input
1	@	0–7 P X D
2	@R	0–7 S
3	@1000/ 123456	0–7 <CR>
4	@R1/123456	0–7 <CR> <LF>
5	@1000	0–7 / G
6	@R1 or @RS	/
7	@1000/ 123456 1000	0–7 <CR> <LF>
8	@R1/ 123456 1000	0–7 <CR>
9*	@DY	0 1 <CR>
10*	@DX	0 1 <CR>
11*	@DD	0 1 <CR>

*Do not enter zero or one followed by <CR>.

4.4.2 <CR> (ASCII 15) Carriage Return

The <CR> command is used to close an open location. If a location's contents are to be changed, the user should precede the <CR> with the new data. If no change is needed, <CR> closes the location without modifying its contents.

Example: @R1/004321<SPACE> <CR> <CR> <LF>
@

Processor register R1 was opened, and no change was needed so the user issued <CR>. In response to the <CR>, Macro-ODT printed <CR> <LF>@.

Example: @R1/004321<SPACE> 1234 <CR> <CR> <LF>
@

In this example, the user wanted to change R1. The new data, 1234, was entered before issuing the <CR>. Macro-ODT deposited the new data into the open location and then printed <CR> <LF> @. Macro-ODT echoes the <CR> entered by the user before it prints <CR> <LF> @.

Example: @1000/012525<SPACE> 1234 <CR> <CR> <LF>
@

where:
first line = new data, 1234, entered into location 1000. The location is closed with <CR>.

4.4.3 <LF> (ASCII 12) Line Feed

The <LF> command is used to close an open location and then open the next contiguous location. LSI-11 bus addresses are incremented by two. If a processor register is open and an <LF> command is issued, the register is closed and any data that was typed in before the <LF> will not enter the register. ODT prints the error message <CR> ? <CR> <LF>. If the open location's contents are to be changed, the new data should precede the <LF>. If no data is entered, the location is closed without being modified.

Example: @1000/123456<SPACE> <LF> <CR> <LF>
@1002/054321<SPACE>

In this example, the user entered <LF> with no data preceding it. In response, Macro-ODT closed location 1000 and then opened location 1002.

4.4.4 R (ASCII 122) Internal Register Designator

When followed by a register number, 0 to 7, or PSW designator, 'S', the R designator will open that specific processor register.

Example: @R0/054321<SPACE>

or

@R7/000123<SPACE> 456 <CR> <CR> <LF>
@

If more than one character is typed (numeral or 'S') after the 'R', Macro-ODT uses all the characters as the register designator.

Example: @R00007/000123<SPACE> <CR> <CR> <LF>
@

4.4.5 S (ASCII 123) Processor Status Word (PSW)

The S designator opens the PSW and must be used after the user has entered the R register designator.

Example: @RS/100377<SPACE> 0 <CR> <CR> <LF>
@RS/000010<SPACE>

The T-bit filter prevents the user from setting the T-bit via Macro-ODT. The T-bit can be cleared by any write to the PSW. When the filter is disabled, the T-bit can be set by loading the PSW to set bit 4 to a one. This is normally not considered desirable. The T-bit filter can be disabled by setting bit 15 of location 167772 to a one.

The PRIORITY 7 filter prevents the user from setting a priority level of 7 via Macro-ODT. Operation at priority level 7 masks out (disables) the BREAK interrupt and makes it impossible to return to Macro-ODT. This operation is normally unacceptable. If required, the PRIORITY 7 filter can be disabled by setting bit 7 of location 167772 to a one. With the filter disabled, a priority level of 7 is selected by writing 340 into the PSW.

4.4.6 G (ASCII 107) Go

The G command is used to start program execution at a location entered immediately before the 'G' in the command string.

Example: @200G

The Macro-ODT sequence for a G command, after echoing the command character, follows.

1. Load R7 (PC) with the entered data. (In the previous example, R7 is equal to 200 and that is where program execution starts.)

2. The PSW is cleared to zero.

3. The LSI-11 bus is initialized by the processor's asserting BINIT L for 17 μs minimum and then negates BINIT L.

4. The user program starts execution at the location specified.

The user is warned that the G command clears the PSW to permit clock interrupts to be acknowledged. Failure to load the address of the clock service routine into the clock vector address (100) may cause unpredictable results.

4.4.7 P (ASCII 120) Proceed

The P command is used to restart execution of a program. No programmer visible machine state is changed using this command.

Example: @P

Program execution restarts at the address pointed to by R7. After 'P' is echoed, Macro-ODT exits, and the program restarts execution.

4.4.8 DD, DX, DY Bootstraps

The D command is used to bootstrap a standalone program or XXDP+ diagnostics from an RX01 or RX02 floppy diskette or a TU58 tape cassette. The next character after the D command determines the type of device being booted. A numerical character, either zero or one, is used to specify a selected drive or unit of the device being booted. If <CR> is typed instead of zero or one, unit 0 is assumed.

Examples: Boot unit 0 of TU58 device:

@DD<CR>

Boot unit 1 of RX01 device:

@DX1

Boot unit 0 of RX02 device:

@DY0

Boot unit 0 of RD51 or RX50:

@DU0

Boot unit 0 of RL:

@DL0

NOTE
Do not type both unit number and <CR>.

To boot a diskette drive, ODT expects the RXV11 or RXV21 controller CSR address to be configured for 177170. To boot the TU58, it must be connected to SLU2 and its' baud rate set for 38,400.

Any error detected during the execution of a boot command will cause a halt at one of many addresses in the boot section of the ROM, with the PC contents printed on the console. The actual addresses and the specific errors they represent are given in the listing provided with the option.

Some errors, however, are not reported. If a TU58 is not connected to SLU2 or if baud rates are incompatible, or if the RL is not loaded, no error indication is given after using the applicable boot command, and the program waits forever. This is also true when booting from floppy diskettes when the drive power is off. In either condition, the user can use <BREAK> to return to ODT prompt level (@).

The D command performs the following operations.

1. If there is no RAM memory at address 0, the D command will cause a halt.

2. The command initializes the LSI-11 bus by asserting BINIT L for 17 μs minimum.

3. It reads block 0 (the first 512 bytes) from the selected mass storage device into memory locations 000–777.

4. It reads location 0 and if it is 240, it loads R1 register with the CSR address of the booted device, loads R0 register with the selected unit or drive number, and jumps to location 0.

5. If the device is an RX01/RX02 or a TU58 and if the content of location 0 is 260, the mass storage device contains a standalone program. Macro-ODT interprets the contents of locations 2, 4, and 6 as a RADIX-50 encoded six character file name. Macro-ODT assumes that the mass storage device is an RT-11 file structured volume and searches the directory of the volume for the file name provided by locations 2, 4, and 6. When the file is found, the complete file is loaded into contiguous memory starting at location 0. The R0 register is loaded with the number of the unit or drive, and the R1 register is loaded with the CSR address of the booted device. The stack pointer (SP) is loaded with the contents of location 42 and the program counter (PC) is loaded with the contents of location 40. The program starts execution.

6. If the content of location 0 is not 240 or 260, the device does not contain a valid boot block. The boot command is aborted, and the SBC-11/21 PLUS is initialized as if a power-up occurred.

4.4.9 X (ASCII 130) Diagnostics

After typing the letter 'X', there is a three-second delay before an octal number is displayed. This command is described in detail in Chapter 2.

4.5 INITIALIZATION

When it is necessary to reinitialize the system without removing power, the user enters 173000G from the console in response to the '@' prompt. After a delay, the user types a carriage return to resynchronize the terminal as shown in the following example.

Example: @173000G

After a delay of at least one second, the user types <CR> to resynchronize.

4.6 WARNINGS AND PROGRAMMING HINTS

The following warnings and programming hints are provided to help the user operate Macro-ODT.

4.6.1 Error Decoding

When an '@' appears unexpectedly, it is good practice for the user to examine the word at 167774. This is an error word that indicates the cause of entry to ODT. A HALT instruction, BREAK, or trying to fetch from nonexistent memory will appear as 100000. Other attempted bus transactions to nonexistent memory will appear as 000200, or, if accessed by the stack pointer R6, as 000201.

4.6.2 ODT Stack Warning

While performing its various functions, Macro-ODT requires two words of user stack. It will push and pop internal information there. Therefore, it is necessary that the user always provide two more words than those necessary for the correct execution of the application program. If desirable, these two words can be given back when the program is completely debugged and operating within its own ROMs without ODT.

For correct program operation, R6 should always contain a valid even RAM memory address. Failure to observe this rule will cause unpredictable results.

4.6.3 Addresses to Avoid

Because the firmware uses the top of the SBC-11/21 PLUS on-board RAM as its scratchpad, the user should not write to any address above 167642 unless specifically defined in this User's Guide.

The vector at 140 controls the BREAK interrupt. Changing locations 140 and 142 could result in the inability to suspend program execution.

4.6.4 CPU Priority
When the PSW is set to 340, the BREAK key will have no effect and will not invoke Macro-ODT. Running at a level 6 priority (PSW set to 300) is acceptable for most programming needs. This will disable all interrupts except for BREAK.

4.6.5 Terminal Related Problems
Macro-ODT echoes every character typed in response to the '@' prompt. Some intelligent terminals also respond to control characters as commands. The results may include loss of communication.

4.6.6 Spurious Halts
When the last word of an instruction is all zeros and causes a bus time-out, Macro-ODT will interpret it as a HALT instruction and print the contents of PC on the terminal before issuing the '@' prompt.

4.6.7 Serial I/O Protocol
The Macro-ODT operates the serial line interface in full-duplex mode, and each character is echoed by the microprocessor to the terminal. Programmed I/O methods are used instead of interrupts. When the Macro-ODT firmware is busy printing a multicharacter message using the transmit side of the interface, the firmware is not monitoring the receive side for incoming characters. Any characters coming in at this time are lost. The interface may set the overrun error bit, but the Macro-ODT does not check this bit, and those characters are not recognized. All peripherals communicating with the Macro-ODT through this interface must observe this protocol.

4.6.8 Interrupt Vector Initialization
On power-up, Macro-ODT initializes the LTC interrupt vector (REVNT at 100) and the BREAK interrupt vector (BKRQ at 140). Other vectors are not initialized and may contain erroneous data.

4.6.9 Boot Rom Address Scheme
The code for the SBC-11/21 PLUS resides in locations 164000–173776 (octal) when maps 0 or 2 are selected. The code for addresses 164000–167776 is physically coded into the UPPER half of the 2K × 8 PROMs used for the KXT11-A5s', and the code for addresses 170000–173776 is coded into the LOWER half of the 2K × 8 PROMs.

CHAPTER 5
SYSTEM ARCHITECTURE

5.1 INTRODUCTION
This chapter describes the architecture of the microprocessor, memory organization, and power-up method. The microprocessor architecture describes the registers, hardware stack, interrupts, and direct memory access (DMA) mechanism. The memory organization describes byte or word addressing and memory mapping. The power-up procedure and initialization are also described.

5.2 MICROPROCESSOR ARCHITECTURE
The SBC-11/21 PLUS microprocessor executes a subset of the PDP-11 instruction set. It has eight high-speed general-purpose registers that are used as accumulators, address pointers, index registers, and for other special functions. The microprocessor executes single and double operand instructions using either 16-bit words or 8-bit bytes. The direct memory access (DMA) function transfers data directly from the LSI-11 bus to the on-board I/O devices and memory while the program continues to run.

5.2.1 Registers
As shown in Figure 5-1, the microprocessor contains a number of internal registers that are used for many purposes. The registers are divided into two groups:

1. General
2. Status

5.2.1.1 General Registers – The microprocessor contains eight 16-bit general-purpose registers that can perform many functions. These registers operate as accumulators, index registers, autoincrement registers, autodecrement registers, or as stack pointers for temporary storage of data. Arithmetic operations can be performed from one general register to another, from one memory location or device register to another, or between memory locations or a device register and a general register.

Registers R6 and R7 are dedicated. R6 is the stack pointer (SP) and contains the location (address) of the last entry in the stack. Register R7 is the processor program counter (PC) and contains the address of the next instruction to be executed. It is normally used for addressing purposes only and not as an accumulator.

5.2.1.2 Status Register – The PSW contains information on the current processor status. This information includes the current processor priority, the condition codes describing the arithmetic or logic results of the last instruction, and an indicator for detecting the execution of an instruction to be trapped during program debugging. Figure 5-1 shows the PSW format; Table 5-1 lists status word bit descriptions. Certain instructions allow programmed control of condition code bits and loading and storing (moving) the processor status. Not all instructions affect the condition codes in an obvious way. See Chapter 7 for details on specific instructions.

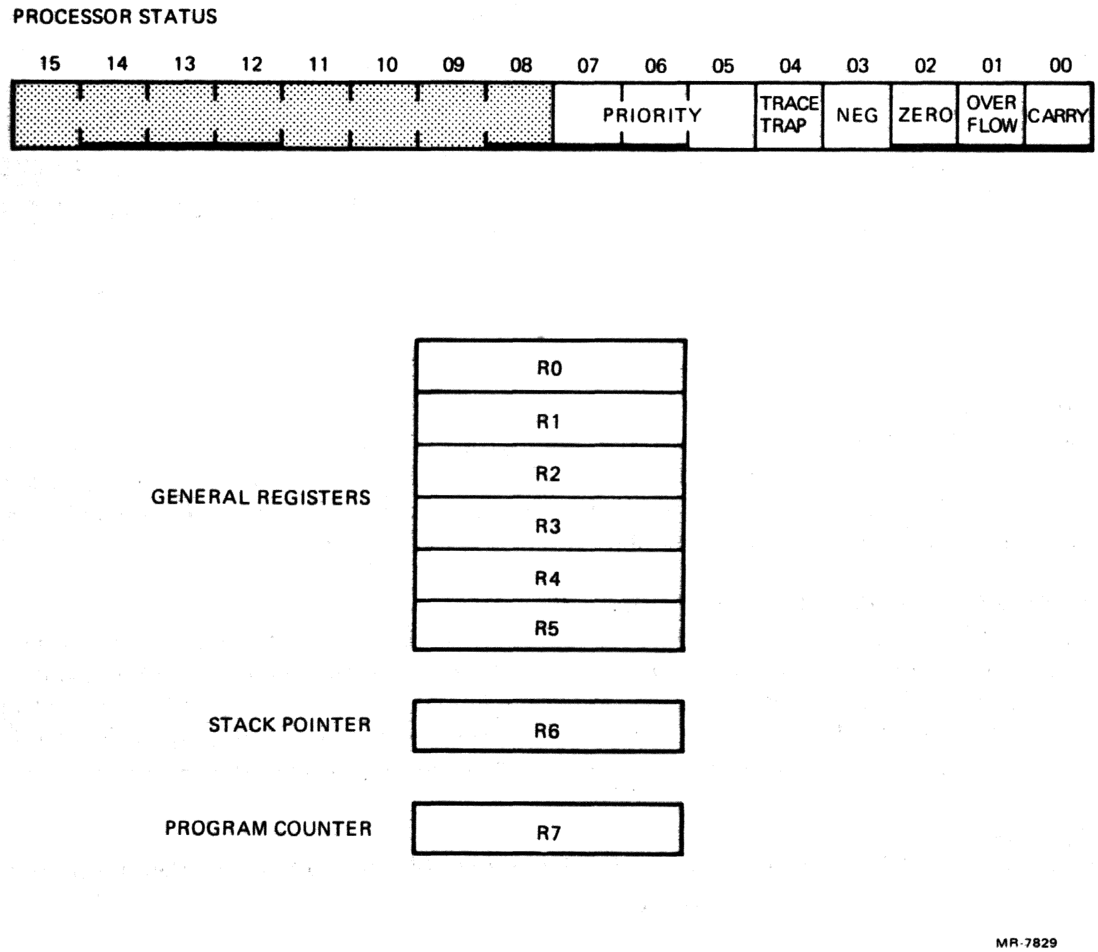


Figure 5-1 Registers and Processor Status Word

5.2.2 Hardware Stack

The hardware stack is part of the basic design architecture of the SBC-11/21 PLUS. It is an area of memory used by the programmer or by the operating system for temporary storage and linkage. It is controlled on a LIFO (last in/first out) basis; items are recovered in the reverse of the order they were stored. The stack starts at the highest location reserved for it (376 octal at power-up) and expands linearly downward to a lower address as items are added to the stack.

It is not necessary to keep track of the actual locations into which data is being stacked. This is done automatically through the use of the stack pointer. Register R6 always contains the memory address where the last item is stored in the stack. Instructions associated with subroutine linkage and interrupt service automatically use R6 as the hardware stack pointer. For this reason, R6 is often referred to as the system SP. The hardware stack is organized in full word units only.

5.2.3 Interrupts

Interrupts are requests, made by peripheral devices, that cause the processor to temporarily suspend its present program execution to service the requesting device. A device can interrupt the processor only when its priority is higher than the processor priority indicated by PSW<7:5>, as shown in Table 5-2.

SBC-11/21 PLUS supports a vectored interrupt structure with priority on four levels. In addition, it supports two nonmaskable interrupts: power fail and HALT.

Table 5-1 Processor Status Word Bit Descriptions

Bits	Name	Description
15–08	N/A	These bits are not accessible to the programmer and contain no valid information.
07–05	Priority	These bits define the current priority level of the microprocessor program, and only interrupts with a higher priority are recognized by the microprocessor. Table 5-2 describes the microprocessor interrupt levels as functions of bits 5–7.
04	Trace	When set, this bit allows the microprocessor to trap to locations 14 and 16 after an instruction is executed. It can only be set by executing an RTI or RTT instruction with the correct PSW on the stack. The trace bit allows programs to be single stepped and is useful for debugging.
03	Condition code N	This bit is set when an instruction causes the result to be negative.
02	Condition code Z	This bit is set when an instruction causes the result to be zero.
01	Condition code V	This bit is set when an instruction causes an overflow condition.
00	Condition code C	This bit is set when an instruction causes a carryout of the most significant bit.

Table 5-2 PSW Interrupt Levels

Microprocessor Priority	Interrupt Levels Acknowledged	PSW Bits 7	6	5
Level 7	Nonmaskable interrupt	1	1	1
Level 6	7	1	1	0
Level 5	7,6	1	0	1
Level 4	7,6,5	1	0	0
Level 0–3	7,6,5,4	0	X	X

Every interrupt except HALT is associated with an interrupt vector. The interrupt vector is a pair of words: the next PC (address of that device's service routine) and the next PSW (priority with which the routine must be executed). Upon interrupt, the current PC and PSW are saved on the stack, and the new PC and PSW are loaded from the vector address.

Up to sixty-four vectors may reside in the first 256 memory locations (octal 374 is the highest vector location). The vector address is provided by the interrupting device (external vector address) or generated internally by the microprocessor.

NOTE

The power fail interrupt uses interrupt vector address 24. The HALT interrupt is not associated with a vector. It pushes the PC and PSW on the stack and immediately goes to the restart address with PSW 340.

The SBC-11/21 PLUS has eleven interrupt sources. Nine of these are maskable; two are nonmaskable. An interrupt request can occur at any time, but it is not acknowledged until the completion of the current instruction. This lets the microprocessor execute a program until the interrupt occurs and then vector to the service routine for the interrupt. After the service routine is completed, a return from interrupt instruction (RTI) is executed. The microprocessor then pops the top two words, the original PC and PSW, from the system stack, and the interrupted program is continued.

Table 5-3 lists the eleven interrupt sources with their priorities. For a device to be serviced, its priority level must be higher than the current microprocessor level. When two devices with equal priority numbers request an interrupt at the same time, the device nearest to the top of the table is serviced first.

When an interrupt is requested by several LSI-11 bus devices at the same time, the device electrically nearest to the SBC-11/21 PLUS is serviced first.

5.3 DMA (DIRECT MEMORY ACCESS)

DMA allows the programmer to implement block transfers by specifying the direction of transfer, the starting address in memory, the number of words, and any additional parameters that an external device requires. SBC-11/21 PLUS does not have an on-board DMA interface but it can support DMA transfers for external devices via the LSI-11 bus interface. A typical device using the DMA mechanism is the RX02 double-density floppy diskette. User designed devices can also be connected to the SBC-11/21 PLUS DMA facility. See Chapter 9 for more information.

5.4 MEMORY ORGANIZATION

The SBC-11/21 PLUS memory uses on-board memory and LSI-11 bus memory. The memory map configurations and the types of on-board memory chips are described in Chapter 2. The memory maps are illustrated in Figure 5-2. Addresses from 0 to 376 octal are reserved for vector locations, and addresses from 60Kb to 64Kb are reserved for I/O devices.

The address space of the SBC-11/21 PLUS module is 64Kb. A 16-bit word is two 8-bit bytes with bits 0–7 representing the low byte and bits 8–15 representing the high byte. Words are always addressed by even numbers. The bytes are addressed by either even or odd numbers. The high bytes are stored in the odd numbered locations, and the low bytes are stored in the even numbered locations.

5.5 POWER-UP/POWER-DOWN FACILITY

The SBC-11/21 PLUS has facilities for an automatic program start-up when power is turned on and for orderly shutdown, without loss of data, when power is turned off or lost. This is done with a combination of hardware features and software.

Table 5-3 SBC-11/21 Interrupts

Interrupt Source	Control Signal	Priority Level	Vector Address**
HALT	−CTMER	Nonmaskable	*
Power fail	−PFAIL	Nonmaskable	24
LSI-11 bus signal BHALT	BKRQ	7	140
LSI-11 bus signal BEVNT	REVNT	6	100
SLU2 REC	RDL2	5	120
SLU2 XMIT	XDL2	5	124
Parallel I/O B	PBRQST	5	130
Parallel I/O A	PARQST	5	134
SLU1 REC	RDL1	4	60
SLU1 XMIT	XDL1	4	64
LSI-11 bus signal BIRQ4	IRQ4	4	Read from LSI-11 bus

* The microprocessor jumps directly to the restart address with a PSW priority level 7. (RESTART is loaded into PC and 340 into PSW.)

** All vectors defined in this table are internal vectors supplied by the microprocessor except for the BIRQ4 interrupt which is read from the bus.

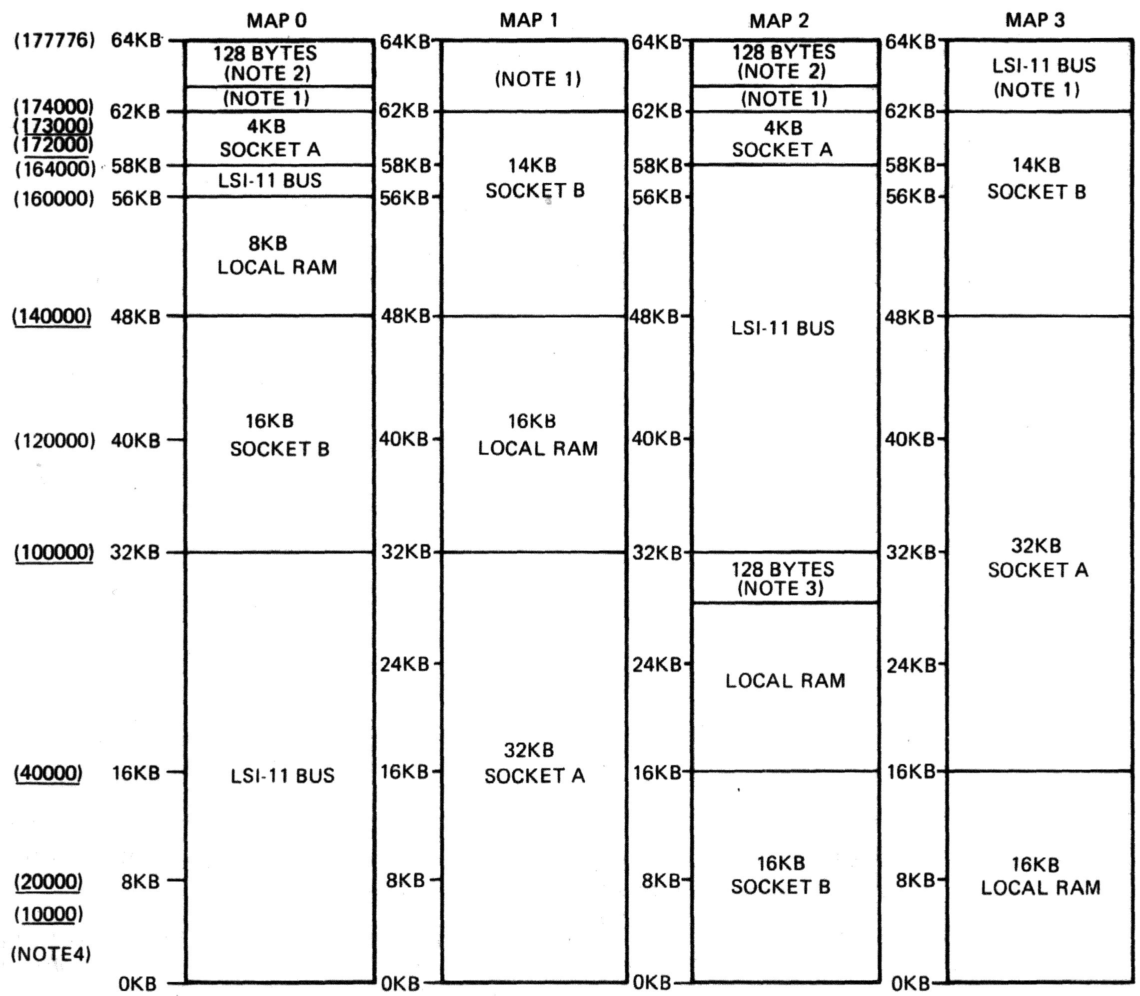
Hardware features:

- Two signal lines in the LSI-11 bus, BDCOK H and BPOK H, are used only for power-up/power-down protocol. These signals are usually generated by the power supply.
- One signal line in the LSI-11 bus, BINIT L, that resets the system.
- The vectoring on interrupt facility of the SBC-11/21 PLUS.
- Battery backup connections.

Software features:

The programmer must provide power-up and power-down routines, and store their addresses at the jumper-selected start address for power-up, and at location 24 for the power-down routine.

For a detailed description of the power-up/power-down protocol, see Chapter 9.



NOTES:

1. THIS SECTION CONTAINS THE LOCAL I/O ADDRESSES FOR THE SLUS AND PPI. ALL UNASSIGNED ADDRESSES ARE ASSUMED TO RESIDE ON THE LSI-11 BUS.
2. ADDRESSES 177777 - 177600 IN MAPS 0 AND 2 ARE RAM SCRATCHPAD LOCATIONS USED BY MACRO-ODT.
3. ADDRESSES 77777 - 77600 IN MAP 2 ARE ALLOCATED TO THE LSI-11 BUS.
4. UNDERLINED ADDRESSES ARE JUMPER SELECTABLE START ADDRESSES. (SEE TABLE 2-4)

MR-12119

Figure 5-2 Memory Maps

CHAPTER 6
PROGRAMMING INFORMATION

6.1 INTRODUCTION

The SBC-11/21 PLUS has three on-board interfaces: one parallel I/O line and two serial I/O lines. These interfaces contain many programmable features that allow the user to change their operating characteristics. This chapter explains how this is done.

The SBC-11/21 PLUS also has hardware that enables the microprocessor to operate in a controlled sequence when the power is turned on and off. This hardware requires software to make it work. The basic principles of this programming are described in Appendix C.

6.2 ASYNCHRONOUS SERIAL LINE UNITS

The two serial line units (SLUs), shown in Figure 6-1, provide the means of transferring data between the microprocessor and two user connectors, J1 or J2. The user interfaces support the EIA RS-232C standard and RS-423 protocol at baud rates from 300 to 38,400.

Each SLU has four addressable registers. These four registers are listed in Table 6-1 and illustrated in Figure 6-2; their functions are described in Table 6-2, Table 6-3, Table 6-4, and Table 6-5. The registers can be accessed by the microprocessor or any DMA bus master. SLU1, with the correct software handling, can be used as a system console and is capable of initiating a hardware interrupt when BREAK is detected. The SBC-11/21 PLUS can be configured for the BREAK to cause a level 7 interrupt with an internal vector of 140, to enable the BHALT interrupt, or to request a HALT trap to the restart address. SLU2 provides three line time clocks at 50 Hz, 60 Hz, and 800 Hz, which can be wire-jumper configured to enable the BEVNT level 6 interrupt. See Chapter 2 for details on how to configure the SLUs.

6.2.1 Data Baud Rates

The serial line units transmit or receive data serially by bit and by character. Each character has ten bits; a start bit, eight bits of data, and the stop bit. Split-speed operation of the receiver and transmitter for the SLU is not supported, and the user cannot supply an external baud rate clock to the SLU. During power-up or reset, the outputs are disabled, and later, the baud rate defaults to 300.

Baud rates are programmable for 300, 600, 1,200, 2,400, 4,800, 9,600, 19,200 or 38,400 when bit 1 of the transmitter control and status register (TCSR) is set to a one. The baud rate is then selected by programming bits 5–3 of the TCSR.

The bits used for the baud rate selection are level sensitive and do not latch. Therefore, the software in control of the TCSR must use bit set and bit reset type instructions after the baud rate is written into the SLU. Each SLU provides an output at TTL levels to pin 1 of its connector (J1 or J2) at sixteen times the baud rate selected for that SLU.

The Macro-ODT option has the autobaud feature that enables SLU1 to adjust itself to the terminal's baud rate between 300 and 9,600 baud. The autobaud feature operates only when Macro-ODT is running on the system.

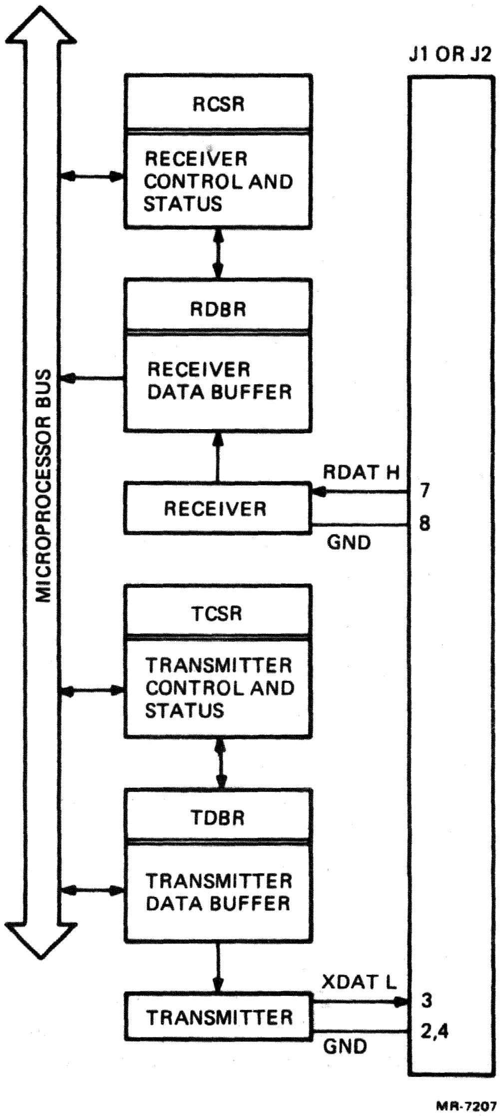


Figure 6-1 Serial Line Unit (SLU) Interface

Table 6-1 Serial Line Unit Register Addresses

Register	Description	Address	AD2	AD1
SLU1				
RCSR	Receiver control and status	177560	0	0
RDBR	Receiver data buffer	177562	0	1
TCSR	Transmitter control and status	177564	1	0
TDBR	Transmitter data buffer	177566	1	1
SLU2				
RCSR	Receiver control and status	176540	0	0
RDBR	Receiver data buffer	176542	0	1
TCSR	Transmitter control and status	176544	1	0
TDBR	Transmitter data buffer	176546	1	1

RECEIVER CONTROL AND STATUS REGISTER

15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
0	0	0	0	RCV ACT	0	0	0	RCV DONE	RCV IE	0	0	0	0	0	0

SLU 1 ADDRESS 177560
SLU 2 ADDRESS 176540

RECEIVER DATA BUFFER REGISTER

15	14	13	12	11	10	09	08	07–00
ERR	OR ERR	FR ERR	0	REC BRK	0	0	0	RECEIVED DATA BUFFER

SLU 1 ADDRESS 177562
SLU 2 ADDRESS 176542

TRANSMITTER CONTROL AND STATUS REGISTER

15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
0	0	0	0	0	0	0	0	XMIT RDY	XMIT IE	PBR SEL2	PBR SEL1	PBR SEL0	MAINT	PBR ENB	XMT BRK

SLU 1 ADDRESS 177564
SLU 2 ADDRESS 176544

TRANSMITTER DATA BUFFER REGISTER

15	14	13	12	11	10	09	08	07–00
0	0	0	0	0	0	0	0	TRANSMIT DATA BUFFER

SLU 1 ADDRESS 177566
SLU 2 ADDRESS 176546

MR-7208

Figure 6-2 Serial Line Unit Register Bit Maps

Table 6-2 Receiver Control and Status Bit Descriptions

Bits	Name	Direction	Function
12–15	Not used	Read only	Reserved for future use.
11	Receiver active	Read only	This bit is set to a one by the start bit and is cleared to a zero by the stop bit at the end of each byte. It is also cleared to a zero on power-up.
08–10	Not used	Read only	Reserved for future use.
07	Receiver done	Read only	This bit is set to a one when the byte received is transferred into the RCV data buffer. It is cleared to a zero when the RCV data buffer is read. It is also cleared to a zero on power-up.
06	Receiver interrupt enable	Read/ write	This bit is set to a one under program control. When set, it allows an interrupt request to be initiated whenever the receiver done bit is set. It is cleared to a zero by reset, power-up, or under program control. Refer to Chapter 2 for interrupt jumper configuration.
00–05	Not used	Read only	Reserved for future use.

Table 6-3 Receiver Data Buffer Bit Descriptions

Bits	Name	Direction	Function
15	Error	Read only	The bit is set to a one when the overrun error or the framing error bit is set. It is cleared to a zero when the error producing condition is removed.
14	Overrun error	Read only	The bit is set to a one when the received byte is transferred into the RCV data buffer before the RCV done bit is cleared. The overrun error indicates that the previous byte in the RCV data buffer was not cleared prior to receiving a new byte. The bit is updated when a byte is transferred into the RCV data buffer and cleared to a zero on power-up.

Table 6-3 Receiver Data Buffer Bit Descriptions (Cont)

Bits	Name	Direction	Function
13	Framing error	Read only	The bit is set to a one when the received character does not have a valid stop bit and is transferred into the RCV data buffer. The bit is cleared to a zero when a character with a valid stop bit is received and is transferred into the RCV data buffer or on power-up.
12	Not used	Read only	Reserved for future use.
11	Received break	Read only	The bit is set to a one when the received signal goes from a mark to a space and stays in the space condition for 11 bit times after serial reception starts. The bit is cleared to a zero when the received signal returns to the mark condition or on power-up.
08–10	Not used	Read only	Reserved for future use.
00–07	Received data buffer	Read only	These eight bits represent the most recent byte received. These bits are cleared to zero on power-up.

Table 6-4 Transmitter Control and Status Bit Descriptions

Bits	Name	Direction	Function
08–15	Not used	Read only	Reserved for future use.
07	Transmitter ready	Read only	The bit is set to a one when the XMIT data buffer is ready to accept a byte. The bit is cleared to a zero by writing into the XMIT data buffer. The bit is also set to a one on power-up.
06	Transmitter interrupt enable	Read/ write	This bit is set to a one under program control. When set, it allows an interrupt request to be initiated whenever the transmitter ready bit is set. The bit is cleared to a zero by reset, power-up, or under program control.

Table 6-4 Transmitter Control and Status Bit Descriptions (Cont)

Bits	Name	Direction	Function
03–05	Programmable* baud rate select	Read/ write	The condition of these bits selects the baud rate under program control provided the programmable baud rate select enable bit is set. The baud rates are selectable by setting these bits as follows. 05 04 03 Baud Rate 0 0 0 300 0 0 1 600 0 1 0 1,200 0 1 1 2,400 1 0 0 4,800 1 0 1 9,600 1 1 0 19,200 1 1 1 38,400 When the programmable baud rate select enable bit is not set, the baud rate defaults to 300.
02	Maintenance	Read/ write	This bit is controlled by the program. When set to a one the transmitter serial output is connected to the receiver serial input and disconnects the external serial input. This bit is cleared to a zero by INIT, power-up, or the program.
01	Programmable* baud rate enable	Read/ write	This bit is controlled by the program. When set to a one, bits 03–05 are used to determine the baud rate. When cleared to a zero, the baud rate will be 300 baud. This bit is cleared to a zero by INIT, power-up, or the program.
00	Transmit break	Read/ write	This bit is controlled by the program. When set to a one, the serial output is forced into the space condition. This bit is cleared by INIT, power-up, or the program.

* The transmitter programmable baud rate select and enable bits are level sensitive and are not latched. This requires that software in control of the TCSR must use bit set and clear instructions to access the TCSR once the baud rate has been written into the SLU.

Table 6-5 Transmitter Data Buffer Bit Descriptions

Bits	Name	Direction	Function
08–15	Not used	Read only	Reserved for future use.
00–07	Transmit data buffer	Read/ write	These eight bits represent the next data byte to be transmitted. These bits are cleared by power-up.

6.2.2 Interrupts

Each SLU provides both a receiver interrupt and a transmitter interrupt to request service from the on-board microprocessor. Receiver and transmitter requests can be independently enabled by software. The receiver interrupt request is enabled when the RCV interrupt enable (bit 6) of the receiver control and status register (RCSR) is set to a one.

SLU2 has a higher interrupt priority, level 5, than SLU1 which has a level 4 interrupt priority. Within each unit, the receiver has higher priority than the transmitter. SLU1 uses vector address 60 for the receiver and 64 for the transmitter. SLU2 uses vector address 120 for the receiver and 124 for the transmitter. These relationships are described in Table 5-3.

6.3 PROGRAMMING THE PARALLEL I/O INTERFACE

The parallel I/O interface, illustrated in Figure 6-3, provides a means of transferring data between the microprocessor bus and the user interface connector J3. The interface has four addressable registers for data and control. Table 6-6 describes these registers.

Port A and B registers are used only for data transfer to and from the user interface. Port C is used for both data transfer and control. The control word register is used only for control of the parallel I/O interface. The interface is programmable by using this register. In addition to software programming, the parallel interface can also be programmed by hardware (see Chapter 2).

The parallel I/O interface is complex, and understanding all its capabilities requires considerable effort. However, efficient use can be made of the parallel I/O using a subset of its capabilities. The following paragraphs are organized to help users find needed information. The flowchart in Figure 6-4 provides an overview of the following discussion on the parallel I/O interface, and helps guide users to the paragraphs of specific interest to them.

6.3.1 Modes of Operation

The interface ports can operate in three basic modes that are selected by system software setting bits in the control word register. The modes are defined as mode 0, 1, and 2 and define how the data is routed through ports A and B.

NOTE

If the bidirectional buffers are being hardwired, care must be taken to ensure that the wired direction agrees with the programmed directions of ports A and B. This is necessary to prevent driver output to driver output connections, which could damage the integrated circuits.

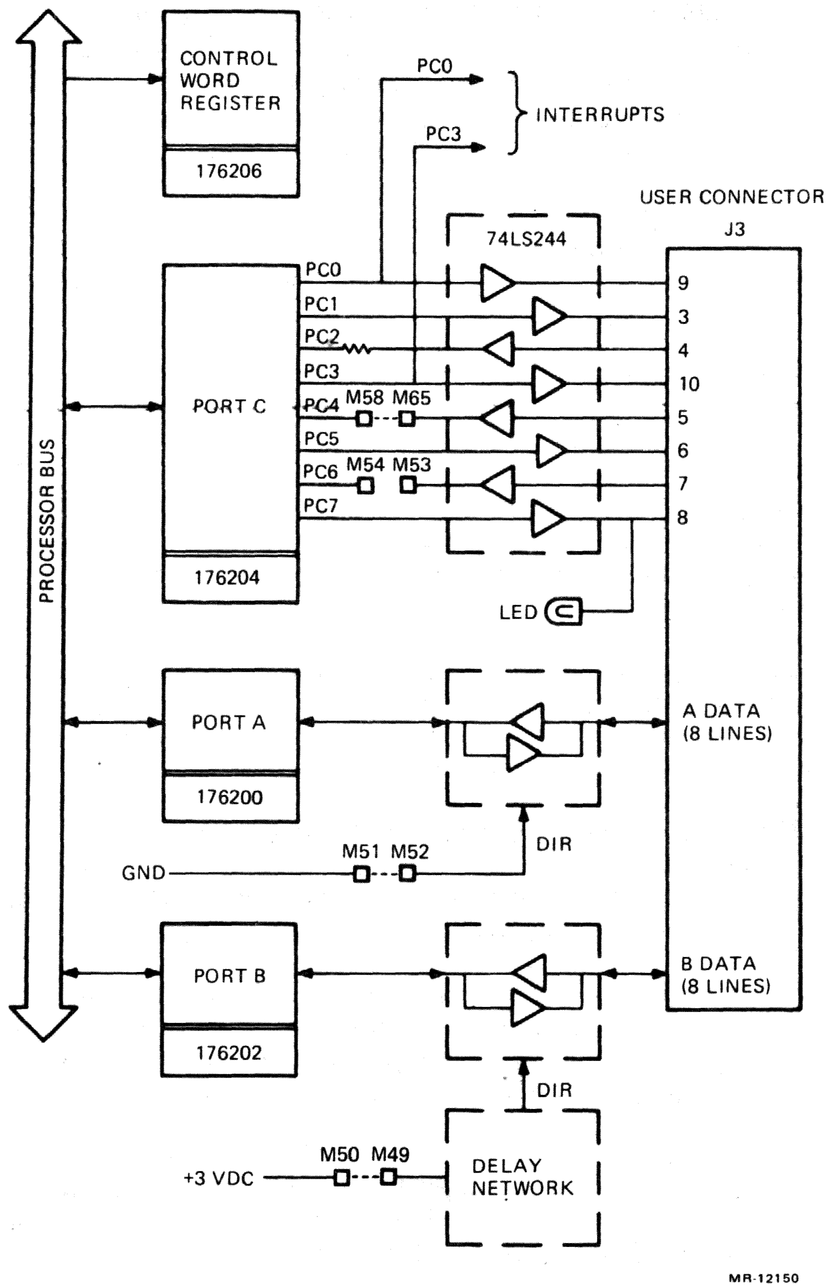


Figure 6-3 Parallel I/O Interface

Table 6-6 Parallel I/O Register Addresses

Register	Address	Status
Port A	176200	Read/write
Port B	176202	Read/write
Port C	176204	Read/write
Control word	176206	Write only

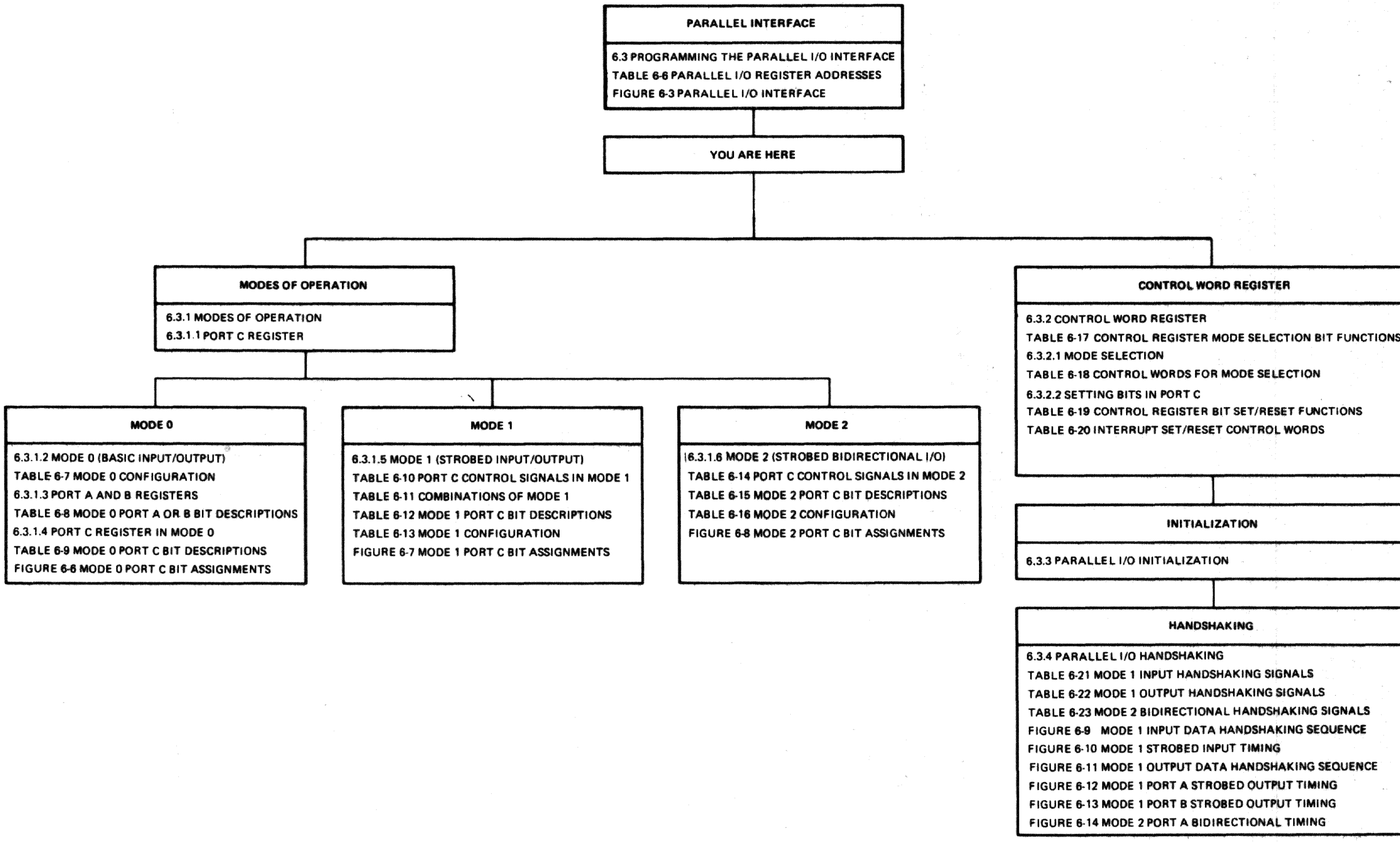


Figure 6-4 Parallel I/O Flowchart

6.3.1.1 Port C Register – The bit assignments for the port C register are dependent on the mode selected and the direction of ports A and B. This register provides the handshake controls to interface between the 8255A-5 and the output connector. The handshake control bits are set/reset by using the control word register which is described in Paragraph 6.3.2. The port C condition for the different modes is described in the following paragraphs that explain the modes.

6.3.1.2 Mode 0 Basic Input/Output – Mode 0 provides simple input and output of either port A or port B or both as described in Table 6-7. The data is read from the port if programmed as an input or written to the port if programmed as an output with no handshaking requirements. The port A and port B bidirectional buffers may be hardwired as described in Chapter 2. They may also be program controlled by port C bits 4 and 6 if dynamic change of the port direction is wanted. In this mode, the outputs are latched but the inputs are not.

Table 6-7 Mode 0 Configuration

PPI Element	To Act as Input	To Act as Output	Direction Control via Port C
Port A	M52 to M50	M52 to M51	M52 to M54 or M58
Port B	M49 to M50	M49 to M51	M49 to M54 or M58
PC7	Never an input	Always an output	
PC6	M54 to M53	Never an output	
PC5	Never an input	Always an output	
PC4	M58 to M65	Never an external output	
PC3	Never an input	Interrupt A (vector 134) Always an output	
PC2	Always an input	Never an output	
PC1	Never an input	Always an output	
PC0	Never an input	Interrupt B (vector 130) Always an output	

6.3.1.3 **Port A and B Registers** – The bit assignments for the port A and B registers are shown in Figure 6-5 and described in Table 6-8. The port A and B registers are used as data buffers for all modes of operation.

6.3.1.4 **Port C Register in Mode 0** – Ports A and B use no handshaking signals, and some port C lines can be used as input/output data lines. The bit assignments are shown in Figure 6-6 and described in Table 6-9. When PC0 and PC3 lines are not used as interrupt requests, they should be cleared by the control word to prevent false interrupts.

6.3.1.5 **Mode 1 (Strobed Input/Output)** – In mode 1, the lines on port C generate or accept signals from the user interface that control the transfer of data through ports A and B. Port C bits 0–3 (lower nibble) are used with port B, and bits 4–7 (upper nibble) are used with port A. These signals are known as handshaking signals. The basic functions of these control signals are defined in Table 6-10 followed by a more detailed description of the handshake protocol.

Table 6-11 describes the four input/output combinations of ports A and B usable in mode 1. The port C bit assignments used in mode 1 are illustrated in Figure 6-7 and described in Table 6-12. Table 6-13 links operation of mode 1 to the jumper configurations discussed in Chapter 2.

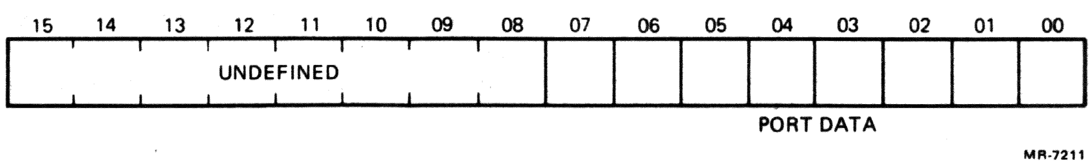


Figure 6-5 Mode 0 Port A or B Bit Assignments

Table 6-8 Mode 0 Port A or B Bit Descriptions

Bits	Name	Direction	Function
08–15	Undefined	–	Not valid if a read is performed on the entire word.
00–07	Port data	Read/write	Data to output or input data to be read, depending on the port direction.

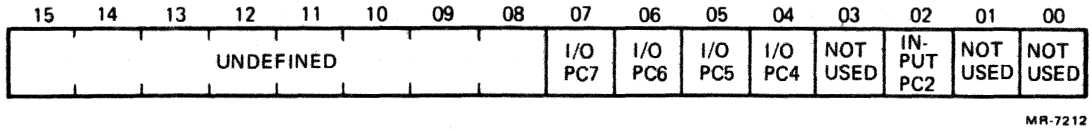


Figure 6-6 Mode 0 Port C Bit Assignments

Table 6-9 Mode 0 Port C Bit Descriptions

Bits	Name	Direction	Function
08–15	Undefined	–	Not valid if a read is performed on the entire word.
07	PC7	Read/write*	Output bit, drives the LED.
06	PC6	Read/write*	If port C upper is defined as input and M54 is connected to M53, it is an input bit. If port C upper is defined as output and M54 is connected to M52 (M49), it is output that controls the buffer direction for port A (port B). A one sets the buffer for input and a zero for output.
05	PC5	Read/write*	Same as PC7, no LED.
04	PC4	Read/write*	If port C upper is defined as input and M58 is connected to M65, it is an input bit. If port C upper is defined as output and M58 is connected to M49 (M52), it is output that controls the buffer direction for port B (port A). A one sets the buffer for input and a zero for output.
03	PC3	Not used	Not valid
02	PC2	Read only	Input bit
00–01	PC0–PC1	Not used	Not valid

* Bit is written by using the control word bit set/reset function explained in Paragraph 6.3.2.

Table 6-10 Port C Control Signals in Mode 1

Signal	Abbreviated/ Port C Bit	Function
Strobe input	STB_A/PC4 STB_B/PC2	A low on this input loads user data into the input latch.
Input buffer full	IBF_A/PC5 IBF_B/PC1	A high on this output acknowledges that the data has been loaded into the input latch. Set by STB and reset by the program reading the input latch.
Interrupt request (Input mode)	$INTR_A$/PC3 $INTR_B$/PC0	A high on this output can interrupt the CPU when an input device strobes its data into the port.
Interrupt enable (Input mode)	$INTE_A$/PC4 $INTE_B$/PC2	Enables setting of $INTR_A$ and $INTR_B$. Program controlled by PC4 or PC2.
Output buffer full	OBF_A/PC7 OBF_B/PC1	This output goes low to tell the user interface that the CPU has written data to the port. Reset by ACK input going low.
Acknowledge input	ACK_A/PC6 ACK_B/PC2	A low on this input tells the processor that the user's device accepted the data from A or B.
Interrupt request (Output mode)	$INTR_A$/PC3 $INTR_B$/PC0	A high on this output can interrupt the CPU when an output device has accepted data transmitted by the CPU. Set by ACK and reset when new data is written to the port.
Interrupt enable (Output mode)	$INTE_A$/PC6 $INTE_B$/PC2	Enables setting of INTR. Program controlled by PC6 or PC2.

Table 6-11 Combinations of Mode 1

Port C Bit Functions	Port A Input with Port B Output	Port A Output with Port B Input	Ports A & B Output	Ports A & B Input
STB_A	PC4	N/A	N/A	PC4
STB_B	N/A	PC2	N/A	PC2
IBF_A	PC5	N/A	N/A	PC5
IBF_B	N/A	PC1	N/A	PC1
$INTR_A$	PC3	PC3	PC3	PC3
$INTR_B$	PC0	PC0	PC0	PC0
OBF_A	N/A	PC7	PC7	N/A
OBF_B	PC1	N/A	PC1	N/A
ACK_A	N/A	PC6	PC6	N/A
ACK_B	PC2	N/A	PC2	N/A
Other port C outputs	PC7 (controls LED)	N/A	PC5	N/A
Other port C inputs	N/A	PC4	N/A	PC6,7
Control Word				
D0 (Direction of PC0–3)	X	X	X	X
D1 (Direction of port B)	0	1	0	1
D2 (Mode of port B)	1	1	1	1
D3 (Direction of PC4–7)	0	1	1	0
D4 (Direction of port A)	1	0	0	1
D5 Port A mode	1	1	1	1
D6 Port A mode	0	0	0	0
D7 Mode set enable	1	1	1	1

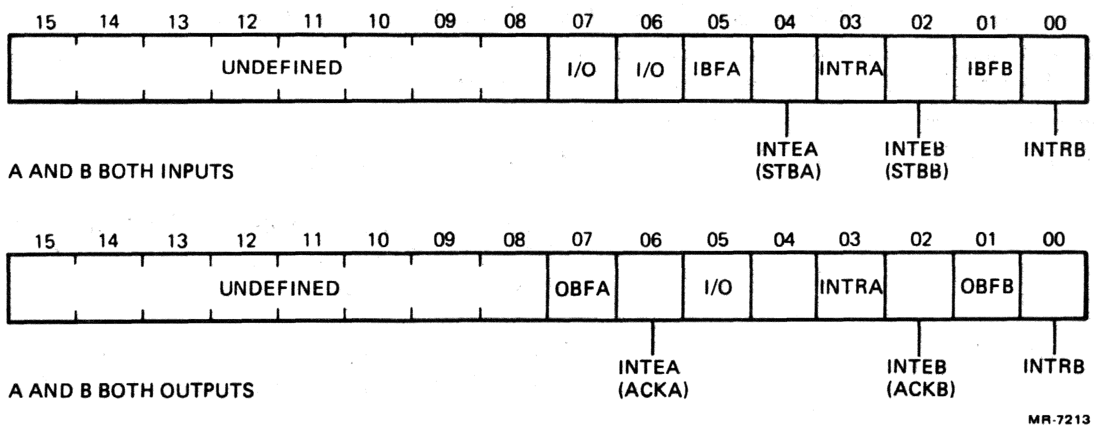


Figure 6-7 Mode 1 Port C Bit Assignments

Table 6-12 Mode 1 Port C Bit Descriptions

Bits	Name	Direction	Function
08–15	Undefined	–	Not valid if a read is performed on the entire word.
07	PC7	Read/write*	If port A mode 1 input:
			If port C bits 04–07, are defined as output, this bit is an output bit and controls the LED. A zero turns the LED on, and a one turns it off.
			Unused if port C bits 04–07 are defined as input.
	OBFA**	Read only	If port A mode 1 output:
			OBFA goes low to indicate that data has been written into the output buffer by the processor. This bit is set when the ACKA (PC6, M54 to M53) input goes low indicating that the external device has accepted the output data. OBFA is present on PC7 to the external device.
06	PC6	Read/write*	If port A mode 1 input:
			If port C bits 04–07 are defined as input and M54 is connected to M53, it is an input bit. If port C bits 04–07 are defined as output and M49 is connected to M54, it is an output that controls the buffer direction for port B. A one sets the buffer for input, and a zero sets the buffer for output.

Table 6-12 Mode 1 Port C Bit Descriptions (Cont)

Bits	Name	Direction	Function
	INTEA	Read/write*	If port A mode 1 output: When set, INTEA enables INTRA to interrupt the SBC-11/21 PLUS when output data has been accepted by the external device.
	ACKA		When M54 is connected to M53, an external signal acknowledging the receipt of data acts as INTEA.
05	IBFA	Read only	If port A mode 1 input: IBFA indicates that the input data has been latched for port A. It is set by the STBA input (PC4, M58 to M65) going low and is reset by the processor reading the port data. This signal is present on PC5 to the external device.
	PC5	Read/write*	If port A mode 1 output: If port C upper is defined as output, it is an output bit. If port C upper is defined as input, it is unused.
04	INTEA	Read/write*	If port A mode 1 input: If set, INTEA will allow INTRA to interrupt the SBC-11/21 PLUS whenever the input buffer is full.
	PC4	Read/write*	If port A mode 1 output: If port C bits 04–07 are defined as output and M49 is connected to M58, this bit is output that controls the direction of the port B buffer. A one sets the buffer for input and a zero sets it for output. If port C bits 04–07 are defined as input and M58 is connected to M65, it is an input bit and is interpreted as STBA (input strobe).
03	INTRA	Read only	If port A mode 1 input: A one indicates that port A has valid input data. It is set by STBA (PC4, M58 to M65) being pulsed low and is reset by the processor reading the port data. INTRA is enabled by INTEA being a one and disabled by INTEA being a zero. If port A mode 1 output: A one indicates that port A is ready to accept new output data. It is set by ACKA (PC6, M54 to M53)

Table 6-12 Mode 1 Port C Bit Descriptions (Cont)

Bits	Name	Direction	Function
			being pulsed low and reset by the processor writing new output data to the port. Enabled and disabled as above. When enabled, INTRA interrupts the processor and has a vector of 134. This signal is also an output to the external device on line PC3.
02	INTEB	Read/write*	When set, INTEB will allow INTRB to interrupt the SBC-11/21 PLUS to request service.
01	IBFB	Read only	If port B mode 1 input: IBFB indicates input data has been latched for port B when a one. It is set by the STBB (PC2) being low and is reset by the processor reading the port data. This signal is present on PC1 to the external device.
	OBFB**	Read only	If port B mode 1 output: OBFB goes low to indicate that the processor has written data to the port. This bit is set by ACKB (PC2) going low, indicating the external device has accepted the output data. This signal is present on PC1 to the external device.
00	INTRB	Read only	If port B mode 1 input: A one indicates port B has valid input data. It is set by STBB (PC2) being pulsed low and is reset by the processor reading the port data. INTRB is enabled when INTEB is one and disabled when it is zero. If port B mode 1 output: A one indicates the port is ready to accept new output data. It is set by ACKB (PC2) being pulsed low and reset by the processor writing new output data to the port. Enabled and disabled as above. This signal is also an output to the external device on PC0.

* Bit is written by using the control word bit set/reset function described in Paragraph 6.3.2.

** If OBF is asserted low and a read or write access is made to the port by the processor before an ACK strobe is sent by the external device, the OBF line for the accessed port will negate during the assertion of the read or write to the port and become reasserted when the read or write operation is complete.

Table 6-13 Mode 1 Configuration

PPI Element	Input Conditions	Output Conditions	Program Control via Port C
Port A	M52 to M50	M52 to M51	N/A
Port B	M49 to M50	M49 to M51	M49 to M54 or M58
PC7	Never an input	Output buffer A full	
PC6	M53 to M54 (Acknowledge A)*	Never an external output	
PC5	Never an input	Input buffer A full	
PC4	M65 to M58 (Strobe A)	Never an external output	
PC3	Never an input	Interrupt A (vector 134)	
PC2	Strobe B in input mode Acknowledge B in output mode	Never an output	
PC1	Never an input	Buffer B full on input or output	
PC0	Never an input	Interrupt B (vector 130)	

*User's hardware acknowledges receipt of data output by port A.

6.3.1.6 Mode 2 (Strobed Bidirectional I/O) – Mode 2 implements communication with a user device over a single 8-bit bus for both transmitting and receiving data. Handshaking and interrupt signals are used as they are in mode 1.

Mode 2 is used with port A only and five control lines on port C. Both inputs and outputs are latched. When port A is operating in this mode, the port B bidirectional buffers cannot be operated under program control because PC4 and PC6 are being used. Port B can operate in either mode 0 or mode 1 but the buffers must be hardwired. PC0–PC2 are defined by port B conditions for mode 1 and are available as I/O lines when port B is in mode 0.

Control signals are defined in Table 6-14. The port C bit assignments as used in mode 2 are illustrated in Figure 6-8 and described in Table 6-15. Table 6-16 links operation of mode 2 to the jumper configurations discussed in Chapter 2.

Table 6-14 Port C Control Signals in Mode 2

Signal	Abbreviated/ Port C Bit	Function
Interrupt request	$INTR_A$/PC3	A high on this output can interrupt the CPU for both input and output operations.
Output buffer full	OBF_A/PC7	This output goes low to indicate that the CPU has written data to port A.
Acknowledge	ACK_A/PC6	A low on this input enables the output tristate buffers of port A to send out the data. Otherwise, that buffer is in the high impedance state.
Interrupt enable	INTEA1/PC6	Enables INTR when OBF is true. Controlled by bit set/reset of PC6.
Strobe input	STB_A/PC4	A low on this input loads data into the input latch.
Input buffer full	IBF_A/PC5	A high on this output indicates that data has been loaded into the input latch.
Interrupt enable	INTEA2/PC4	Enables INTR when IBF is true. Controlled by bit set/reset of PC4.

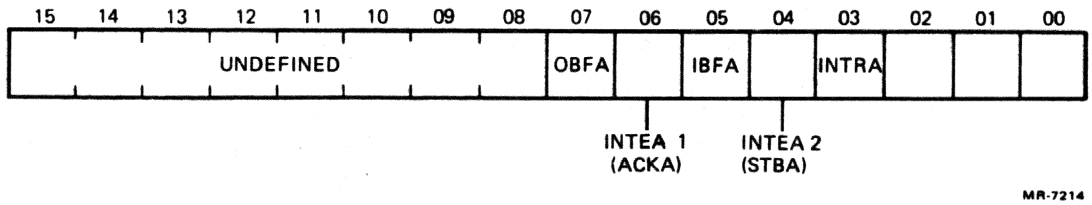


Figure 6-8 Mode 2 Port C Bit Assignments

Table 6-15 Mode 2 Port C Bit Descriptions

Bits	Name	Direction	Function
08–15	Undefined	–	Not valid if a read is done on the entire word.
07	OBFA	Read only	Will go low to indicate that the processor has written output data to the port. It goes high when ACKA (PC6, M54 to M53) goes low indicating the external device has accepted the data. This signal is output on PC7 to the external device.
06	INTEA1	Read/write*	When this bit is set, it allows an interrupt INTRA when the output buffer is ready to accept new data.
05	IBFA**	Read only	IBFA indicates that input data has been latched when it is a one. This bit is reset when the processor reads the input data. This signal is output on PC5 to the external device.
04	INTEA2	Read/write*	When this bit is set, it allows an interrupt INTRA when the input buffer is full.
03	INTRA	Read only	A high on this bit indicates that the port is requesting service of the processor. This signal is output on PC3 to the external device.
02–00	PC0–PC2	–	These bits are defined by port B mode selection.

* Bit is written by using the control word bit set/reset function described in Paragraph 6.3.2.

** When using port A in mode 2 operation, the software must clear the input buffer of port A if the input buffer full flag (IBFA) is set before it performs the read during an intended write to ensure that the handshake lines and port flags are not set out of sequence.

Table 6-16 Mode 2 Configuration

PPI Element	Input Conditions	Output Conditions
Port A	Bidirectional bus	M52 to M54 to M53
Port B	Hardwired only	Hardwired only
PC7	Never an input	Output buffer A full
PC6	Acknowledge A	Never an output
PC5	Never an input	Input buffer A full
PC4	Strobe A (M65 to M58)	Never an output
PC3	Never an input	Interrupt A (vector 134)
PC2	Always an input	Never an output
PC1	Never an input	Always an output
PC0	Never an input	Always an output

6.3.2 Control Word Register

The control word register controls the operation of the parallel interface. If bit 7 is set, the contents of the register determine the mode of operation and the input/output direction of the ports. If bit 7 is cleared, the contents of the register set/reset the port C register bits. The functions of the register bits are described in Table 6-17 and are selected by the state of the bit.

6.3.2.1 Mode Selection – The user determines the mode of operation for the ports and defines them as inputs, outputs, or bidirectional. The user then must ensure that the bidirectional buffers are configured (see Chapter 2) to match the software requirements. Table 6-18 lists all the control words available for the control word register. The user selects the control word that matches the requirements and loads it into the register. The register is defined as write only; reading the register results in erroneous data.

6.3.2.2 Setting Bits in Port C – The control word register is also used to set or reset the port C register bits. The control word bit functions are described in Table 6-19. To set a bit, the register is loaded with bit 7 cleared, bits 1–3 equal to the bit number being set, and bit 0 set. To reset the same bit, bit 0 is cleared. The bit set/reset can be used to enable or disable the port A and port B interrupts for the SBC-11/21 PLUS. The control words used to enable or disable the interrupts are listed in Table 6-20.

Table 6-17 Control Register Mode Selection Bit Functions

Bits	Bit Set	Bit Reset
08–15	Unused	Unused
07	Always set	Always set
06	Port A mode 2	Port A mode 0 or 1
05	Port A mode 1	Port A mode 0
04	Port A input	Port A output
03	Port C bits 04 and 06 inputs	Port C bits 04–07 outputs
02	Port B mode 1	Port B mode 0
01	Port B input	Port B output
00	Port C bit 02 input	Port C bits 00, 01, and 03 outputs

Table 6-18 Control Words for Mode Selection

	Port B Mode 0 IN	Port B Mode 0 OUT	Port B Mode 1 IN	Port B Mode 1 OUT	Port C PC4,PC6	Port C PC5,PC7
Port A Mode 0 IN	233	231	237	235	Input	
	233	221	227	225		Output
Port A Mode 0 OUT	213	211	217	215	Input	
	203	201	207	205		Output
Port A Mode 1 IN	273	271	277	275	Input	
	263	261	267	265		Output
Port A Mode 1 OUT	253	251	257	255	Input	
	243	241	247	245		Output
Port A Mode 2	3X3	3X1	3X7	3X5	*	

*Port C unavailable, used for handshaking.
X = Do not care condition.

Table 6-19 Control Register Bit Set/Reset Functions

Bits	Function
08–15	Not used
07	Always reset
06–04	Not used
03–01	These bits select the port C bit that is to be set or reset as follows.
00	This bit is set to set the selected bit or is cleared to reset the selected bit of port C.

Bit	03	02	01
PC0	0	0	0
PC1	0	0	1
PC2	0	1	0
PC3	0	1	1
PC4	1	0	0
PC5	1	0	1
PC6	1	1	0
PC7	1	1	1

Table 6-20 Interrupt Set/Reset Control Words

		INTRA		INTRB	
Mode	Direction	Enable	Disable	Enable	Disable
1	Input	011	010	005	004
1	Output	015	014	005	004
2	Input	011	010	None*	None*
2	Output	015	014	None*	None*

*Port B does not function in the bidirectional mode 2.

6.3.3 Parallel I/O Initialization

During power-up or the execution of a RESET instruction, the port C data lines are driven high and the LED (driven by bit 7 of port C) is turned off. If the bidirectional buffers of ports A and B are hardwired, the directions are not changed, and the data lines are driven high if the buffer is configured as an output. If the bidirectional buffers of ports A and B are program controlled by port C, the data lines will go to the input state.

6.3.4 Parallel I/O Handshaking

The parallel I/O can operate in either mode 0, 1, or 2 to transfer data into or out of the SBC-11/21 PLUS. The mode 0 data transfers do not require any handshaking control signals. The mode 0 input data is not latched, and data should be available on the I/O connector at the same time as the read strobe enables the 8255A-5. The mode 0 output data is latched, and data is valid at the I/O connector 362 ns after the trailing edge of the write strobe to the 8255A-5.

The handshaking signals that pass across the user interface are detailed as follows. Mode 1 operation requires the handshaking control signals; these are dependent on defining the ports as inputs or outputs. Mode 1 input signals are listed in Table 6-21, and the handshaking function is shown in Figure 6-9. Mode 1 input timing is described in Figure 6-10. Mode 1 output signals are listed in Table 6-22, and the handshaking function is shown in Figure 6-11. Mode 1 output timing is described for port A in Figure 6-12, and mode 1 output timing is described for port B in Figure 6-13. Mode 2 operation allows port A to be bidirectional. The handshaking signals are listed in Table 6-23, and mode 2 timing is described in Figure 6-14. When port A operates in mode 2, port B can operate only in mode 0 or mode 1.

Table 6-21 Mode 1 Input Handshaking Signals

Signal Name	Function
STB (A or B) Port A - PC4 Port B - PC2	Strobe input - This signal is asserted low by the external device and loads data into the SBC-11/21 PLUS input port latch. It must be asserted low for 525 ns minimum.
IBF (A or B) Port A - PC5 Port B - PC1	Input buffer full - This signal is asserted by the SBC-11/21 PLUS in response to an assertion of STB to notify the interface that data was loaded into the input latch.
INTR (A or B) Port A - PC3	Interrupt request - This signal can be used to generate an interrupt to the microprocessor. The bitset/bitreset commands must be used to enable/disable the INTE bit for each port. Interrupts will be generated either when STB is negated with IBF asserted, or when ACK is negated with OBF asserted.

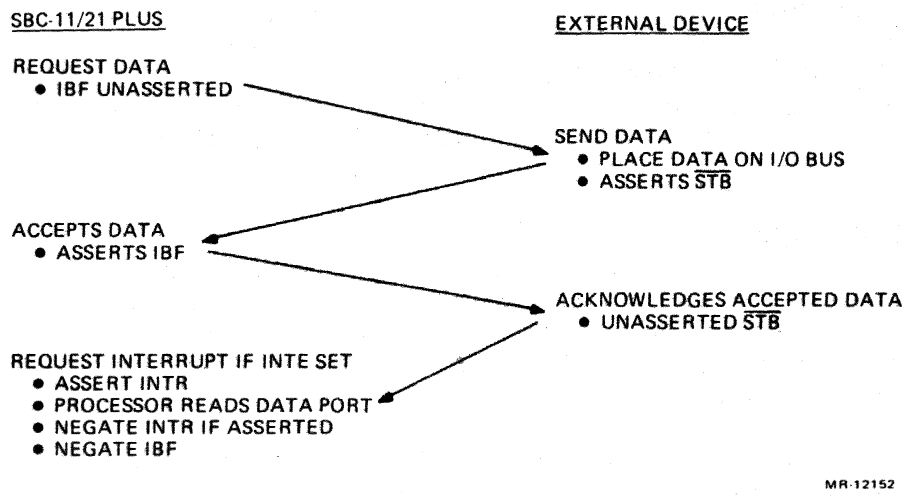


Figure 6-9 Mode 1 Input Data Handshaking Sequence

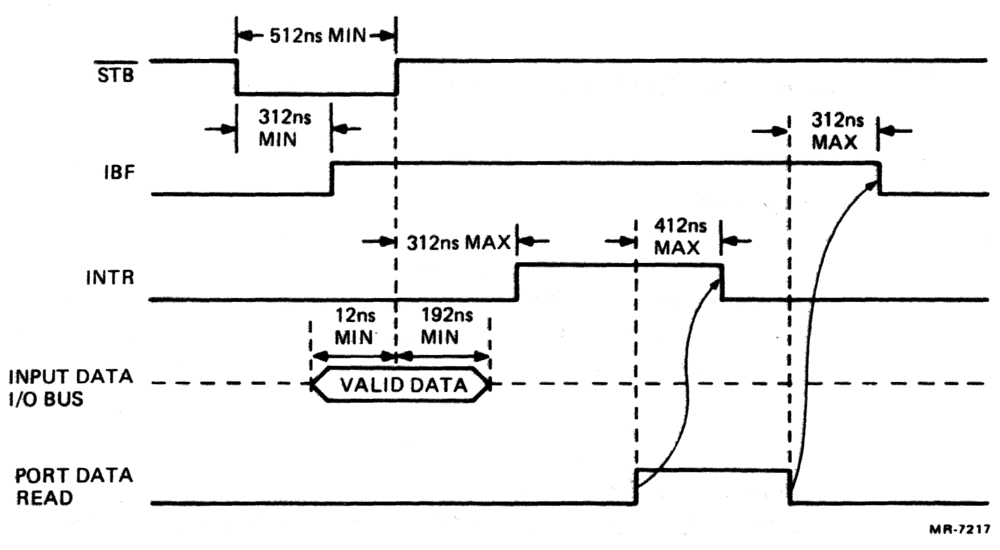


Figure 6-10 Mode 1 Strobed Input Timing

Table 6-22 Mode 1 Output Handshaking Signals

Signal Name	Function
OBF (A or B) Port A – PC7 Port B – PC1	Output buffer full – This output is asserted low to indicate that the microprocessor has written data into the specified port latches.
ACK (A or B) Port A – PC6 Port B – PC2	Acknowledge input – This signal is asserted low by the external device to indicate it has accepted the latched output data from the specified port.
INTR (A or B) Port A – PC3 Port B – PC0	Interrupt request – This signal can be used to generate an interrupt to the microprocessor when the external device has received the data and INTE is set and ACK is negated.

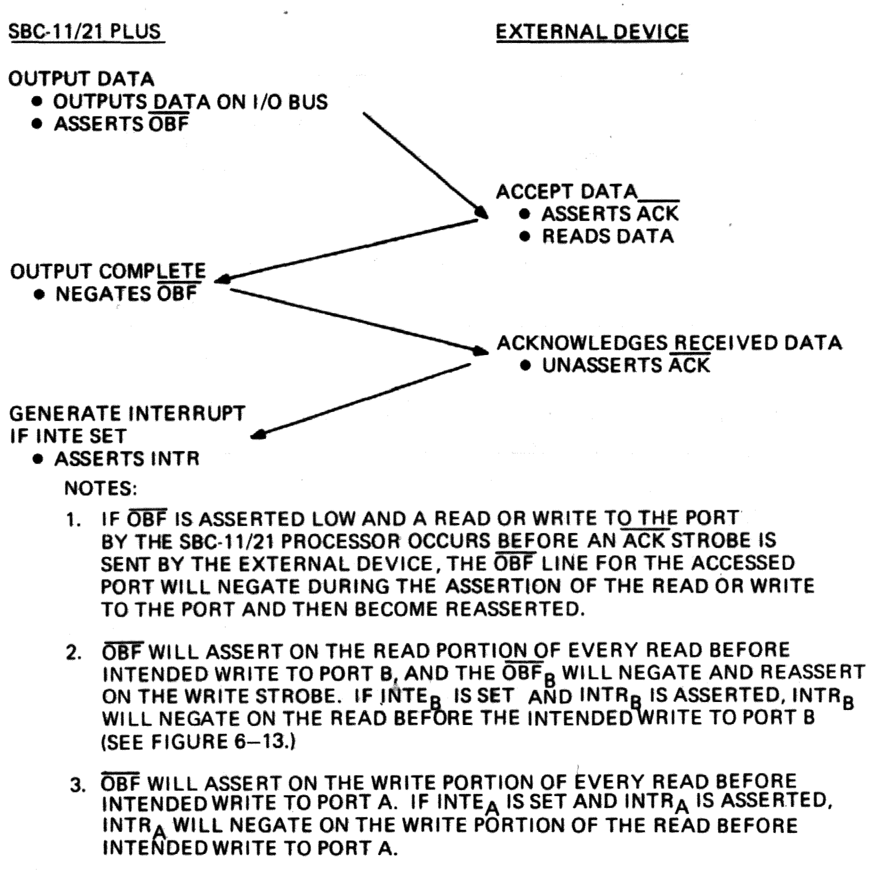


Figure 6-11 Mode 1 Output Data Handshaking Sequence

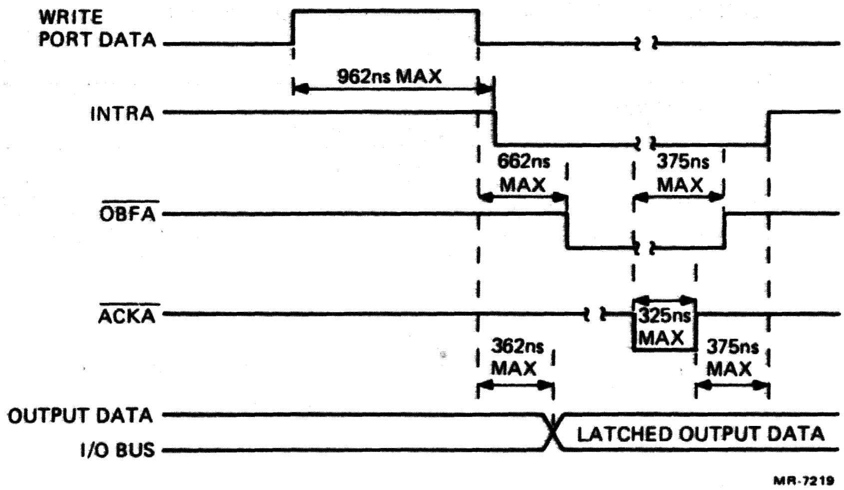
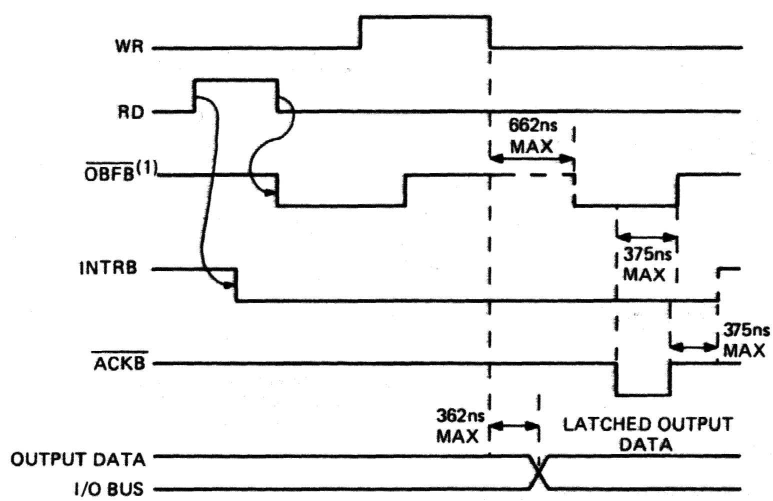


Figure 6-12 Mode 1 Port A Strobed Output Timing



NOTE:
1. OBF WILL ASSERT ON THE READ PORTION OF EVERY READ BEFORE INTENDED WRITE TO PORT B AND THE $\overline{OBF}_B$ WILL NEGATE AND REASSERT ON THE WRITE STROBE. IF $INTE_B$ IS SET AND $INTR_B$ IS ASSERTED, $INTR_B$ WILL NEGATE ON THE READ BEFORE THE INTENDED WRITE TO PORT B.

MR-7220

Figure 6-13 Mode 1 Port B Strobed Output Timing

Table 6-23 Mode 2 Bidirectional Handshaking Signals

Signal Name	Function
STB (PC4)	Strobe input – This signal is asserted low by the external device and strobes data into port A.
IBF (PC5)*	Input buffer full – This signal is asserted when the microprocessor has accepted STB strobe.
INTR (PC3)	Interrupt request – This signal can be used to generate an interrupt to the microprocessor when the external device is demanding service.
OBF (PC7)	Output buffer full – This output is asserted to indicate that the microprocessor has written data into the output port latches.
ACK (PC6)**	Acknowledge input – This signal is asserted low by the external device to indicate it has taken data from the output port latches. It controls the DIR pin of the port A buffer.

* Because every write is preceded by a read, the contents of the input buffer should be saved if IBF_A is asserted prior to writing port A mode 2 data.

** When mode 2 is configured, PC6 (ACK) is jumpered to the port A direction control pin through a rising edge delay circuit. Hence, when PC6 is negated, the rising edge is delayed by 250 ns minimum. This means that the buffer will be driving data out of the connector 250 ns minimum after the user interface negates ACK.

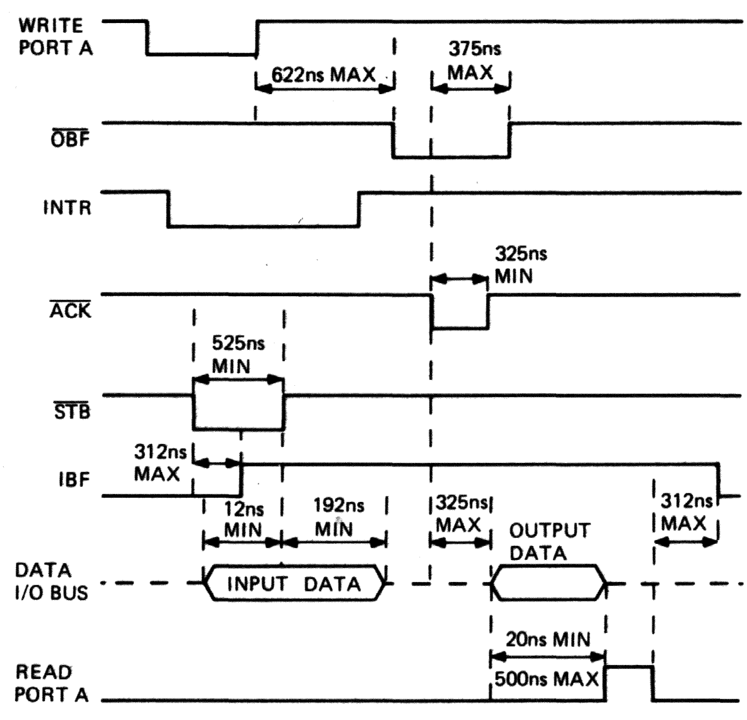


Figure 6-14 Mode 2 Port A Bidirectional Timing

CHAPTER 7
ADDRESSING MODES AND INSTRUCTION SET

7.1 INTRODUCTION
This chapter provides a detailed discussion of addressing modes and descriptions of individual instructions. The discussion of addressing modes is divided into six major topics.

1. Single operand addressing – One part of the instruction word specifies the registers; the remaining part provides information for locating the operand.

2. Double operand addressing – Part of the instruction word specifies the registers; the remaining parts provide information for locating two operands.

3. Direct addressing – The operand is the content of the selected register.

4. Deferred (indirect) addressing – The content of the selected register is the address of the operand.

5. Use of the program counter (PC) as a general-purpose register – The PC is unique from other general-purpose registers. Whenever the processor retrieves an instruction, it automatically advances the PC by two. By combining this automatic advancement of the PC with four of the basic addressing modes, four special PC modes are produced – immediate, absolute, relative, and relative deferred.

6. Use of the stack pointer (SP) as a general-purpose register – The SP can be used for stack operations.

NOTE

Instruction mnemonics and address mode symbols are sufficient for writing assembly language programs. The programmer need not be concerned about conversion to binary digits; this is accomplished automatically by the assembler program.

7.2 ADDRESSING MODES
Data stored in memory must be accessed and manipulated. Data handling is specified by an SBC-11/21 PLUS instruction (MOV, ADD, etc.) that usually specifies the following.

1. The function to be performed (operation code).

2. A general-purpose register to be used when locating the source and/or destination operand.

3. An addressing mode that specifies how the selected register(s) is/are to be used.

Most data handled by a computer is structured (in character strings, arrays, lists, etc.). SBC-11/21 PLUS addressing modes allow efficient and flexible handling of structured data.

The general-purpose registers may be used with an instruction in any of the following four ways.

1. As accumulators. The data to be manipulated resides within the register.

2. As pointers. The content of the register is the address of the operand, rather than the operand itself.

3. As pointers that automatically step through memory locations. Automatically stepping forward through consecutive locations is known as autoincrement addressing; automatically stepping backward is known as autodecrement addressing. These modes are particularly useful for processing tabular or array data.

4. As index registers. The contents of the register and the word following the instruction are summed to produce the address of the operand. This allows easy access to variable entries in a list.

The register arrangement is an important microprocessor feature that should be considered in conjunction with the addressing modes. There are six general-purpose registers (R0–R5), a hardware stack pointer (SP) register (R6), and a program counter (PC) register (R7).

Registers R0–R5 are not dedicated to any specific function; their use is determined by the instruction that is decoded.

1. They can be used for operand storage. For example, the contents of two registers can be added and stored in another register.

2. They can contain the address of an operand or serve as pointers to the address of an operand.

3. They can be used for the autoincrement or autodecrement features.

4. They can be used as index registers for convenient data and program access.

The SBC-11/21 PLUS also has instruction addressing mode combinations that facilitate temporary data storage structures. These combinations can be used for conveniently handling data that must be accessed frequently. This is known as stack manipulation. The register that keeps track of stack manipulation is the stack pointer (SP). Any register can be used as a stack pointer under program control; however, certain instructions associated with subroutine linkage and interrupt service automatically use register R6 as a hardware stack pointer, and therefore, R6 is frequently referred to as the SP.

- The stack pointer keeps track of the latest entry on the stack.

- The stack pointer moves down as items are added to the stack and moves up as items are removed. It always points to the top of the stack.

- The hardware stack is used during trap or interrupt handling to store information and allow the processor to return to the main program.

Register R7 is used by the processor as its program counter (PC) and should not be used as a stack pointer or accumulator. Whenever an instruction is fetched from memory, the program counter is automatically incremented by two to point to the next instruction word.

7.2.1 Single Operand Addressing

The instruction format for all single operand instructions (such as clear, increment, and test) is illustrated in Figure 7-1.

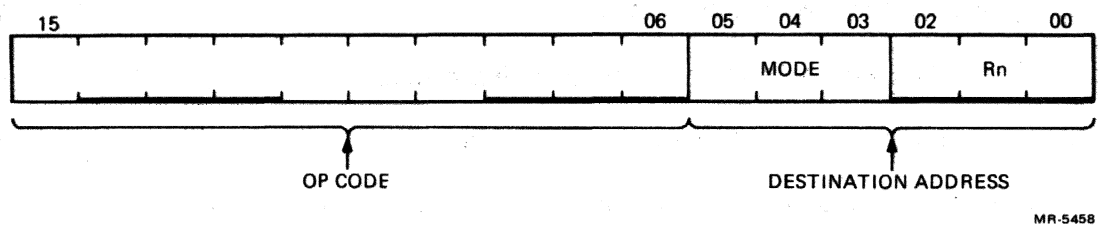


Figure 7-1 Single Operand Addressing

Bits 15–6 specify the operation code that defines the type of instruction to be executed. Bits 5–0 form a 6-bit field called the destination address field that consists of two subfields.

1. Bits 0–2 specify which of the eight general-purpose registers is to be referenced by the instruction word.

2. Bits 3–5 specify how the selected register will be used (address mode). Bit 3 is set to indicate deferred (indirect) addressing.

7.2.2 Double Operand Addressing

Operations that imply two operands (such as add, subtract, move, and compare) are handled by instructions that specify two addresses. The first operand is called the source operand; the second operand is called the destination operand. Bit assignments in the source and destination address fields may specify different modes and different registers. The instruction format for the double operand instruction is illustrated in Figure 7-2.

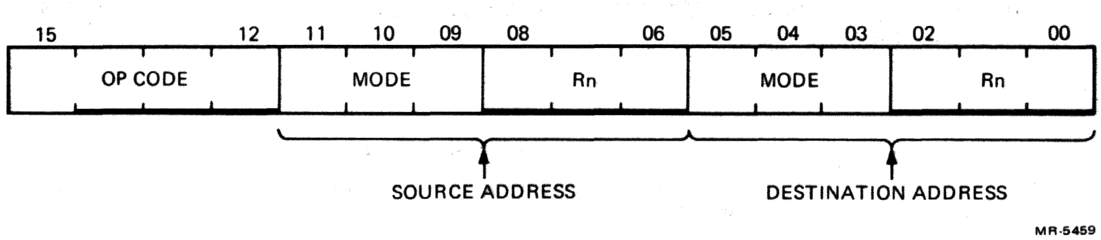


Figure 7-2 Double Operand Addressing

The source address field is used to select the source operand, the first operand. The destination is used similarly and locates the second operand and the result. For example, the instruction ADD A, B adds the contents (source operand) of location A to the contents (destination operand) of location B. After execution, B will contain the result of the addition; the contents of A will be unchanged.

Examples in this chapter use the sample SBC-11/21 PLUS instructions listed in Table 7-1. See Paragraph 7.3 for a complete list of the SBC-11/21 PLUS instructions.

Table 7-1 Sample SBC-11/21 PLUS Instructions

Mnemonic	Description	Octal Code
CLR	Clear (zero the specified destination)	0050DD
CLRB	Clear byte (zero the byte in the specified destination)	1050DD
INC	Increment (add one to the contents of the destination)	0052DD
INCB	Increment byte (add one to the contents of the destination byte)	1052DD
COM	Complement (replace the contents of the destination by its logical complement; each zero bit is set and each one bit is cleared)	0051DD
COMB	Complement byte (replace the contents of the destination byte by its logical complement; each zero bit is set and each one bit is cleared)	1051DD
ADD	Add (add source operand to destination operand and store the result at destination address)	06SSDD

DD = destination field (6 bits)
SS = source field (6 bits)
() = contents of

7.2.3 Direct Addressing

Table 7-2 summarizes the four basic modes used with direct addressing. Figures 7-3, 7-4, 7-5, and 7-6, which follow the table, illustrate these four modes.

Table 7-2 Direct Addressing Modes

Mode	Name	Assembler Syntax	Function
0	Register	Rn	Register contains operand.
2	Autoincrement	(Rn)+	Register is used as a pointer to sequential data, then incremented.
4	Autodecrement	–(Rn)	Register is decremented and then used as a pointer.
6	Index	X(Rn)	Value X is added to (Rn) to produce address of operand. Neither X nor (Rn) is modified.

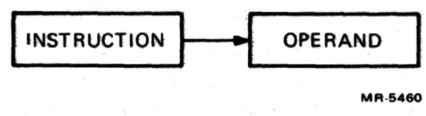


Figure 7-3 Mode 0 Register

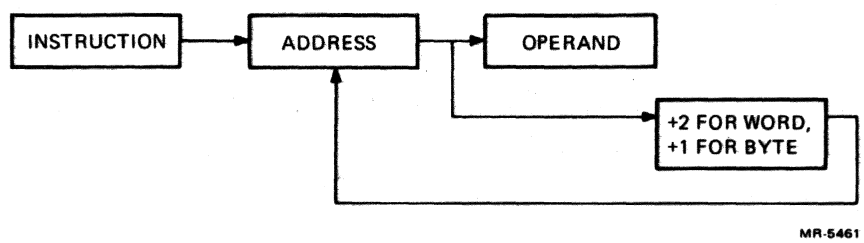


Figure 7-4 Mode 2 Autoincrement

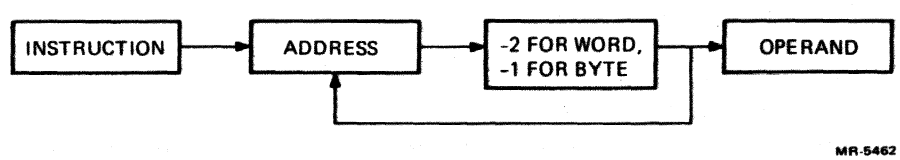


Figure 7-5 Mode 4 Autodecrement

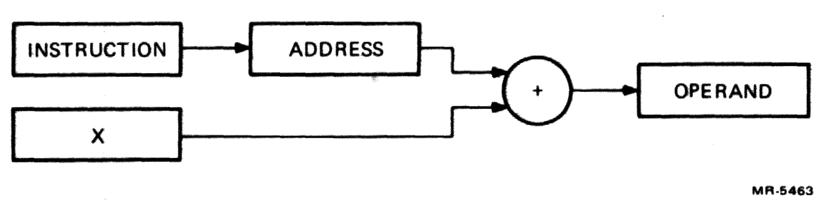


Figure 7-6 Mode 6 Index

7.2.3.1 Register Mode (Mode 0) – With register mode, any of the general-purpose registers may be used as simple accumulators, and the operand is contained in the selected register. Because they are hardware registers, within the processor, the general-purpose registers operate at high speeds and provide speed advantages when used for operating on frequently accessed variables. The assembler interprets and assembles instructions in the following form as register mode operations.

OPR Rn

Rn represents a general-purpose register name or number, and OPR represents a general instruction mnemonic. Assembler syntax requires that a general-purpose register be defined as follows.

R0 = %0 (The '%' sign indicates register definition.)
R1 = %1
R2 = %2, etc.

Registers are typically referred to by name as R0, R1, R2, R3, R4, R5, R6, and R7. However, R6 and R7 are also referred to as SP and PC, respectively.

Register Mode Examples (Figures 7-7, 7-8, and 7-9)
(all numbers in octal)

Symbolic	Octal Code	Instruction Name	Operation
INC R3	005203	Increment	One is added to the contents of the general-purpose register R3.

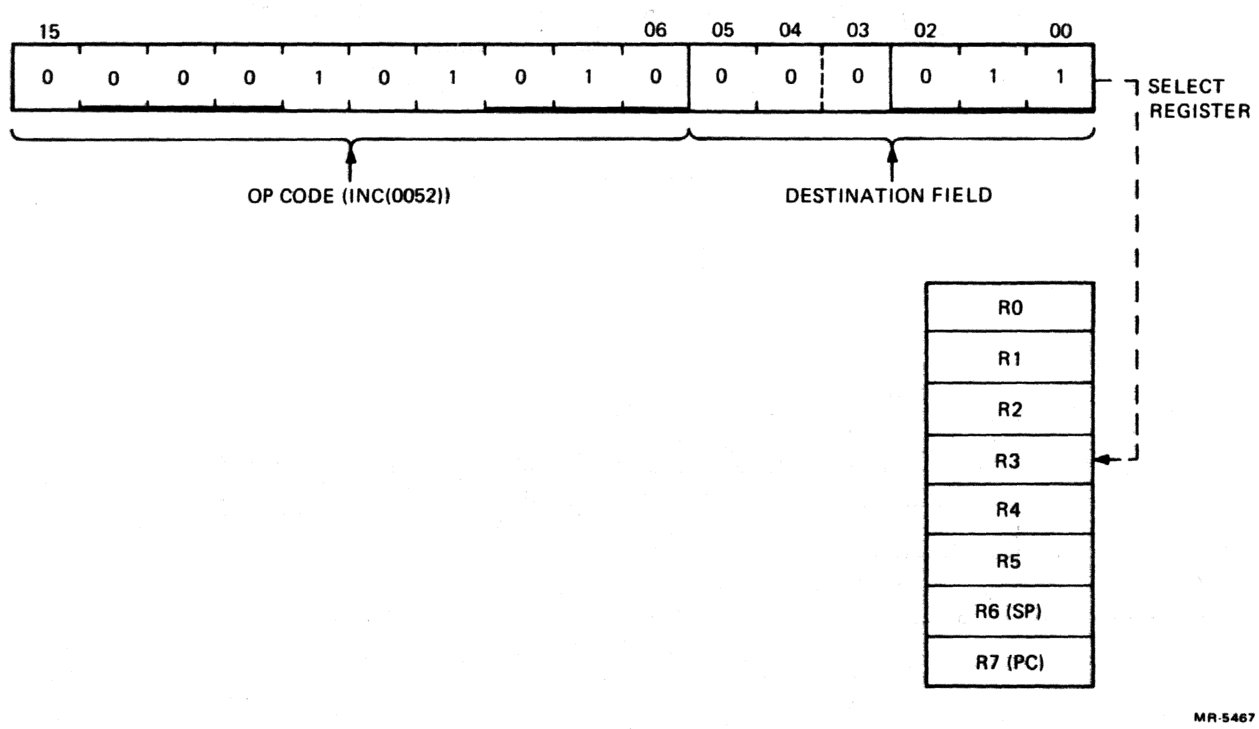


Figure 7-7 INC R3

Symbolic	Octal Code	Instruction Name	Operation
ADD R2, R4	060204	Add	The contents of R2 are added to the contents of R4.

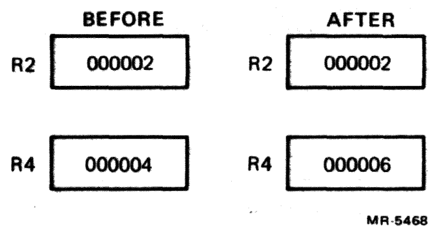


Figure 7-8 ADD R2,R4

Symbolic	Octal Code	Instruction Name	Operation
COMB R4	105104	Complement byte	Complement bits 0–7 (byte) of one in R4. (When general-purpose registers are used, byte instructions only operate on bits 0–7; i.e., byte 0 of the register.)

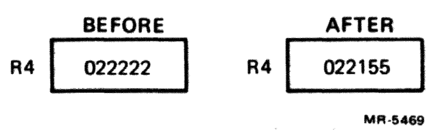


Figure 7-9 COMB R4

7.2.3.2 **Autoincrement Mode (Mode 2)** – Autoincrement mode allows automatic stepping of a pointer through sequential elements of a table of operands. It assumes that the content of the selected general-purpose register is the address of the operand. Contents of registers are stepped (by one for bytes, by two for words, and by two for R6 and R7) to address the next sequential location. The autoincrement mode is especially useful for array processing and stack processing; it accesses an element of a table and then steps the pointer to address the next operand in the table. Although most useful for table handling, this mode is general and may be used for a variety of purposes. The assembler interprets and assembles instructions in the following form as autoincrement mode operations.

OPR (Rn)+

Autoincrement Mode Examples (Figures 7-10, 7-11, and 7-12)

Symbolic	Octal Code	Instruction Name	Operation
CLR (R5)+	005025	Clear	The contents of R5 are used as the address of the operand. The selected operand is cleared, and the contents of R5 are then incremented by two.

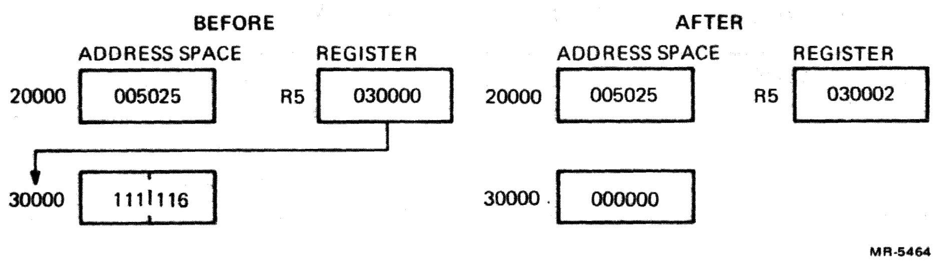


Figure 7-10 CLR (R5)+

Symbolic	Octal Code	Instruction Name	Operation
CLRB (R5)+	105025	Clear byte	The contents of R5 are used as the address of the operand. The selected byte operand is cleared, and the contents of R5 are then incremented by one.

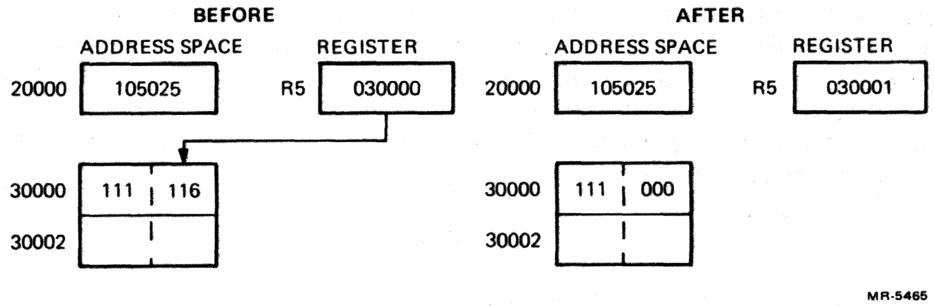


Figure 7-11 CLRB (R5)+

Symbolic	Octal Code	Instruction Name	Operation
ADD (R2)+,R4	062204	Add	The contents of R2 are used as the address of the operand that is added to the contents of R4. R2 is then incremented by two.

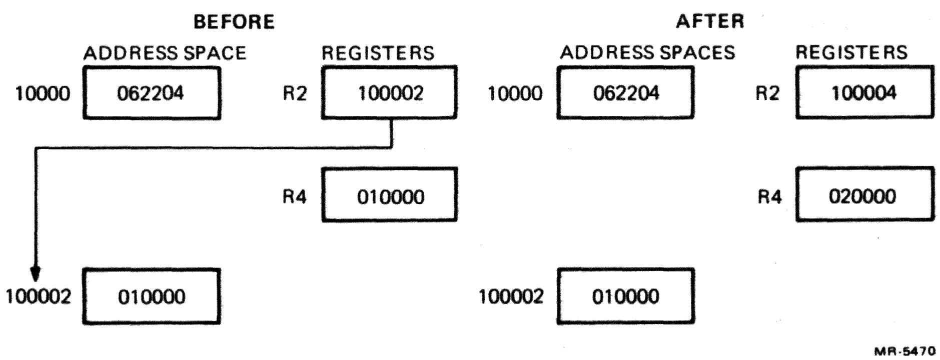


Figure 7-12 ADD (R2)+,R4

7.2.3.3 Autodecrement Mode (Mode 4) – Autodecrement mode is useful for processing data in a list in reverse direction. The contents of the selected general-purpose register are decremented (by two for word instructions, by one for byte instructions) and then used as the address of the operand. The choice of postincrement, predecrement features for the SBC-11/21 PLUS are not arbitrary; they are intended to facilitate hardware/software stack operations. The assembler interprets and assembles instructions in the following form as autodecrement mode operations.

OPR –(Rn)

Autodecrement Mode Examples (Figures 7-13, 7-14, and 7-15)

Symbolic	Octal Code	Instruction Name	Operation
INC –(R0)	005240	Increment	The contents of R0 are decremented by two and used as the address of the operand. The operand is incremented by one.

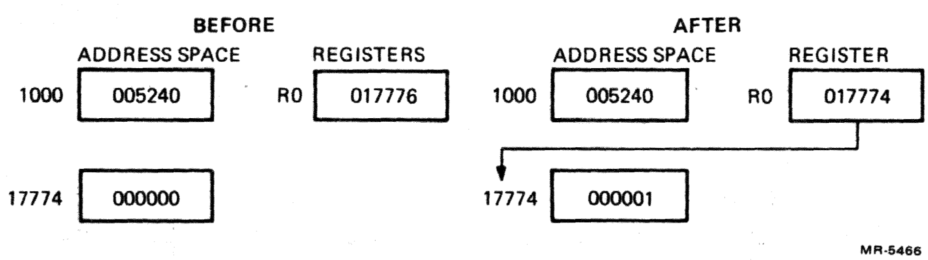


Figure 7-13 INC –(R0)

Symbolic	Octal Code	Instruction Name	Operation
INCB –(R0)	105240	Increment byte	The contents of R0 are decremented by one and used as the address of the operand. The operand byte is increased by one.

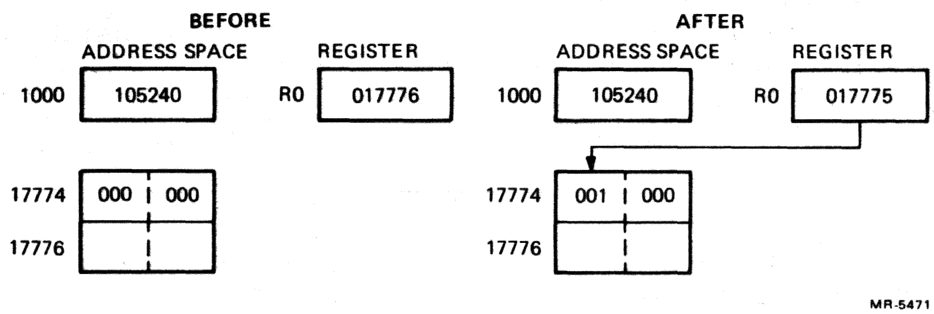


Figure 7-14 INCB –(R0)

Symbolic	Octal Code	Instruction Name	Operation
ADD –(R3),R0	064300	Add	The contents of R3 are decremented by two and then used as a pointer to an operand (source) which is added to the contents of R0 (destination operand).

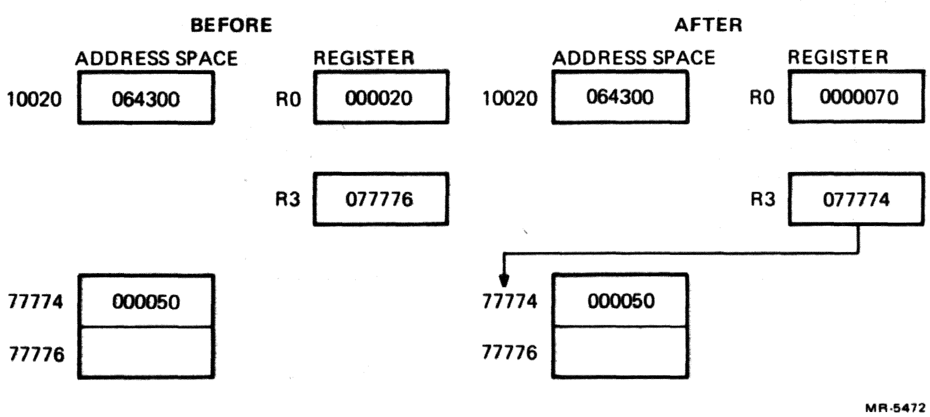


Figure 7-15 ADD –(R3),R0

7.2.3.4 Index Mode (Mode 6) – With index mode, the contents of the selected general-purpose register and an index word following the instruction word are summed to form the address of the operand. The contents of the selected register may be used as a base for calculating a series of addresses, thus allowing random access to elements of data structures. The selected register can then be modified by the program to access data in the table. Index addressing instructions are in the following form:

OPR X(Rn)

where X is the indexed word and is located in the memory location following the instruction word, and Rn is the selected general-purpose register.

Index Mode Examples (Figures 7-16, 7-17, and 7-18)

Symbolic	Octal Code	Instruction Name	Operation
CLR 200(R4)	005064 000200	Clear	The address of the operand is determined by adding 200 to the contents of R4. The operand location is cleared.

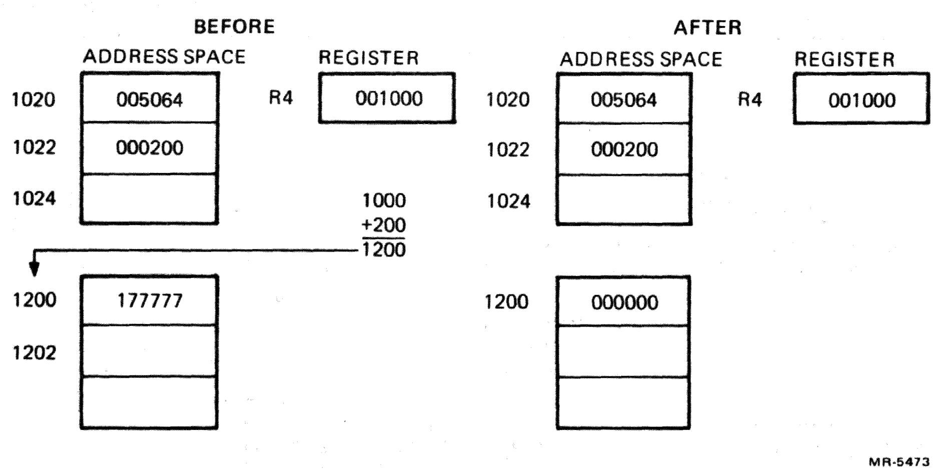


Figure 7-16 CLR 200(R4)

Symbolic	Octal Code	Instruction Name	Operation
COMB 200(R1)	105161 000200	Complement byte	The contents of a location that is determined by adding 200 to the contents of R1 are one's complemented (i.e., logically complemented).

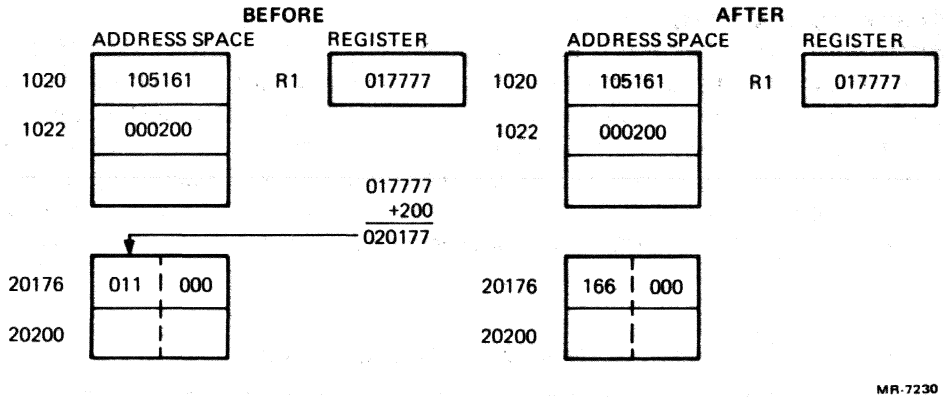


Figure 7-17 COMB 200(R1)

Symbolic	Octal Code	Instruction Name	Operation
ADD 30(R2),20(R5)	066265 000030 000020	Add	The contents of a location that is determined by adding 30 to the contents of R2 are added to the contents of a location that is determined by adding 20 to the contents of R5. The result is stored at the destination address, i.e., 20(R5).

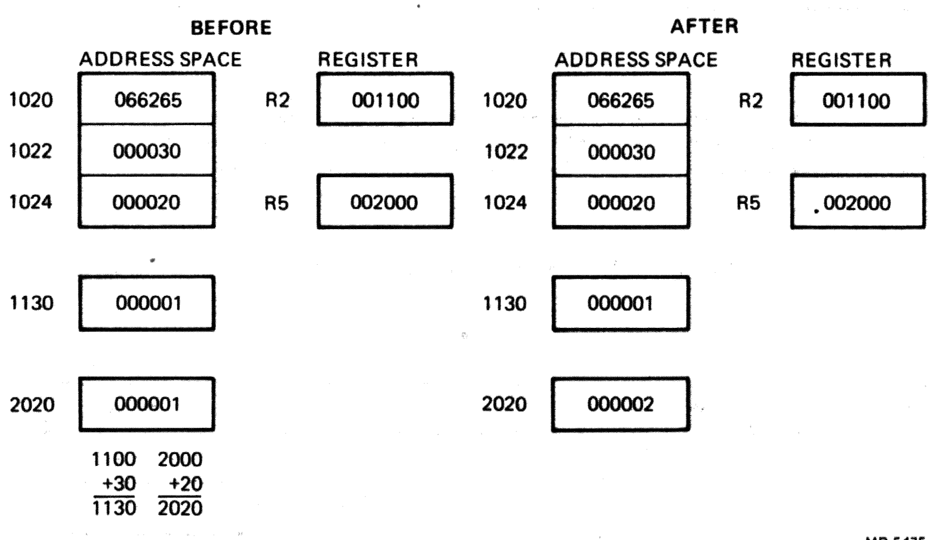


Figure 7-18 ADD 30(R2),20(R5)

7.2.4 Deferred (Indirect) Addressing

The four basic modes may also be used with deferred addressing. In the register mode, the operand is the content of the selected register; in the register deferred mode, the content of the selected register is the address of the operand. In the three other deferred modes, the contents of the register select the address of the operand rather than the operand itself. Therefore, these modes are used when a table consists of addresses rather than operands. Assembler syntax for indicating deferred addressing is '@' (or '()' when this is not ambiguous). Table 7-3 summarizes the deferred versions of the basic modes. Figures 7-19, 7-20, 7-21, and 7-22, which follow the table, illustrate these deferred versions of the basic modes.

Table 7-3 Indirect Addressing Modes

Mode	Name	Assembler Syntax	Function
1	Register deferred	@Rn or (Rn)	Register contains the address of the operand.
3	Autoincrement deferred	@(Rn)+	Register is first used as a pointer to a word containing the address of the operand and then incremented (always by two, even for byte instructions).
5	Autodecrement deferred	@–(Rn)	Register is decremented (always by two, even for byte instructions) and then used as a pointer to a word containing the address of the operand.
7	Index deferred	@X(Rn)	Value X (stored in a word following the instruction) and (Rn) are added, and the sum is used as a pointer to a word containing the address of the operand. Neither X nor (Rn) is modified.

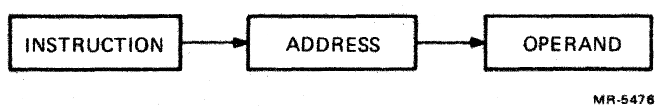


Figure 7-19 Mode 1 Register Deferred

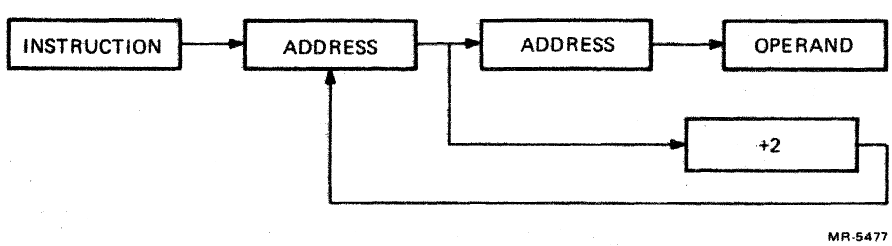


Figure 7-20 Mode 3 Autoincrement Deferred

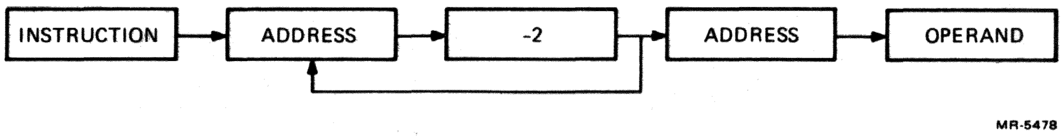


Figure 7-21 Mode 5 Autodecrement Deferred

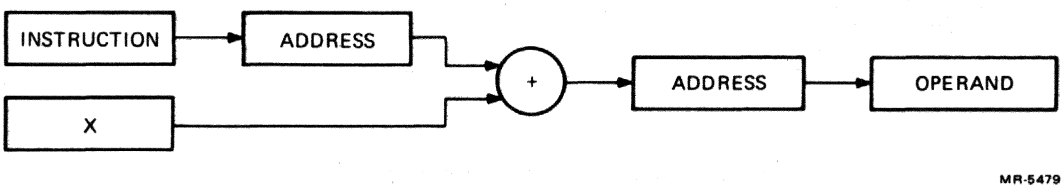


Figure 7-22 Mode 7 Index Deferred

Register Deferred Mode Example – Mode 1 (Figure 7-23)

Symbolic	Octal Code	Instruction Name	Operation
CLR @R5	005015	Clear	The contents of the location specified in R5 are cleared.

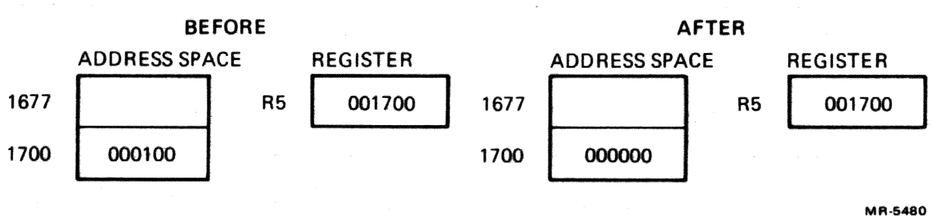


Figure 7-23 CLR @R5

Autoincrement Deferred Mode Example – Mode 3 (Figure 7-24)

Symbolic	Octal Code	Instruction Name	Operation
INC @(R2)+	005232	Increment	The contents of R2 are used as the address of the address of the operand. The operand is increased by one; and the contents of R2 are incremented by two.

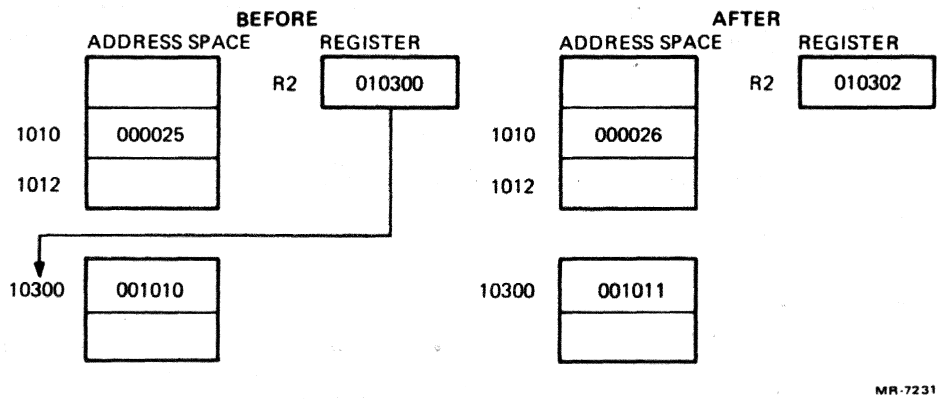


Figure 7-24 INC @(R2)+

Autodecrement Deferred Mode Example – Mode 5 (Figure 7-25)

Symbolic	Octal Code	Instruction Name	Operation
COM @–(R0)	005150	Complement	The contents of R0 are decremented by two and then used as the address of the address of the operand. Operand is one's complemented (i.e., logically complemented).

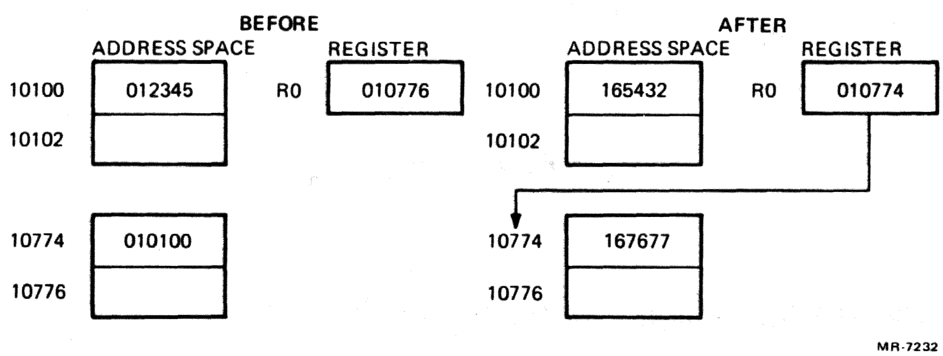


Figure 7-25 COM @ –(R0)

Index Deferred Mode Example – Mode 7 (Figure 7-26)

Symbolic	Octal Code	Instruction Name	Operation
ADD @1000(R2),R1	067201 001000	Add	1000 and the contents of R2 are summed to produce the address of the address of the source operand. The contents of the source operand are added to the contents of R1; the result is stored in R1.

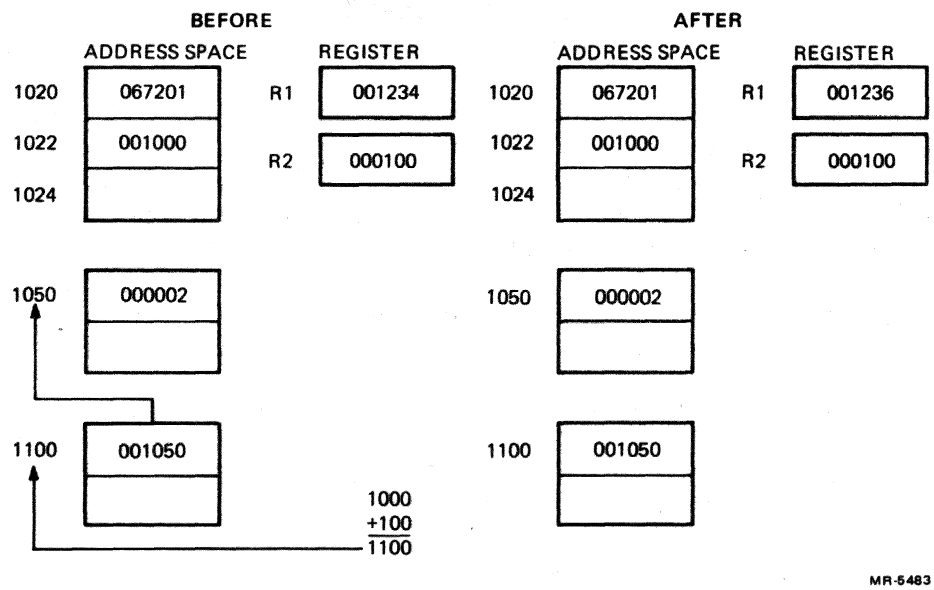


Figure 7-26 ADD @1000(R2),R1

7.2.5 Use of the PC as a General-Purpose Register

Although R7 is a general-purpose register, it doubles as the program counter for the microprocessor. Whenever the processor uses the program counter to acquire a word from memory, the program counter is automatically incremented by two to contain the address of the next word of the instruction being executed or the address of the next instruction to be executed. (When the program uses the PC to locate byte data, the PC is still incremented by two.)

The PC responds to all standard SBC-11/21 PLUS addressing modes. However, the PC provides advantages for handling position independent code and unstructured data with four of these modes. When utilizing the PC, these modes are termed immediate, absolute (or immediate deferred), relative, and relative deferred. Table 7-4 provides a summary of these modes.

Table 7-4 PC Register Addressing Modes

Mode	Name	Assembler Syntax	Function
2	Immediate	#n	Operand follows the instruction.
3	Absolute	@#A	Absolute address of operand follows the instruction.
6	Relative	A	Relative address (index value) follows the instruction.
7	Relative deferred	@A	Index value (stored in the word following the instruction) is the relative address for the address of the operand.

When a standard program is available to different users, the ability to load it into different areas of memory and run it there is useful. The SBC-11/21 PLUS can relocate a program efficiently using position independent code (PIC) that is written using the PC addressing modes. If an instruction and its operands are moved so that the relative distance between them is not altered, the same offset relative to the PC can be used in all positions in memory. Thus, PIC usually references locations relative to the current location.

The PC also facilitates the handling of unstructured data. This is particularly true of the immediate and relative modes.

7.2.5.1 Immediate Mode – Using the immediate mode is equivalent to using the autoincrement mode with the PC. It provides time improvements for accessing constant operands by including the constant in the memory location immediately following the instruction word. The assembler interprets and assembles instructions in the following form as immediate mode operations.

OPR #n,DD

Immediate Mode Example (Figure 7-27)

Symbolic	Octal Code	Instruction Name	Operation
ADD #10,R0	062700 000010	Add	The value 10 is located in the second word of the instruction and is added to the contents of R0. Just before this instruction is fetched and executed, the PC points to the first word of the instruction. The processor fetches the first word and increments the PC by two. The source operand mode is 27 (autoincrement the PC). Thus, the PC is used as a pointer to fetch the operand (the second word of the instruction) before being incremented by two to point to the next instruction.

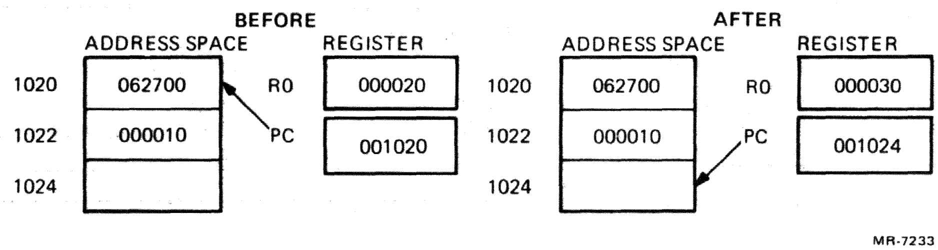


Figure 7-27 ADD #10,R0

7.2.5.2 Absolute Addressing – Using the absolute addressing mode is the equivalent of using the immediate deferred or autoincrement deferred modes with the PC. The contents of the location following the instruction are taken as the address of the operand. Immediate data is interpreted as an absolute address (i.e., an address that remains constant no matter where in memory the assembled instruction is executed). The assembler interprets and assembles instructions in the following form as absolute addressing mode operations.

OPR @#A

Absolute Mode Examples (Figures 7-28 and 7-29)

Symbolic	**Octal Code**	**Instruction Name**	**Operation**
CLR @#1100	005037 001100	Clear	The contents of location 1100 are cleared.

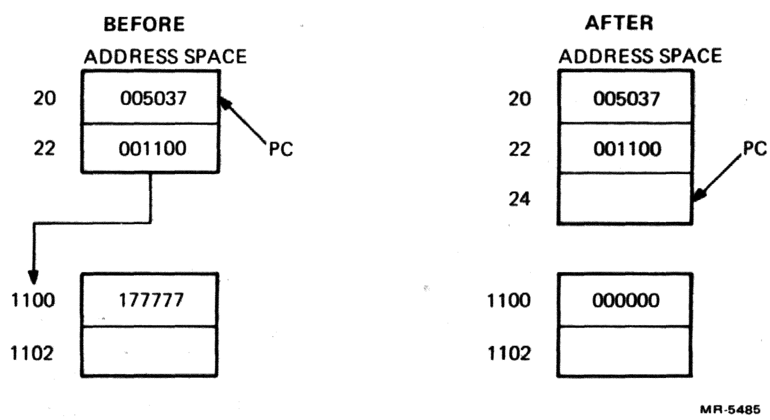


Figure 7-28 CLR @#1100

Symbolic	Octal Code	Instruction Name	Operation
ADD @#2000,R3	063703 002000	Add	The contents of location 2000 are added to R3.

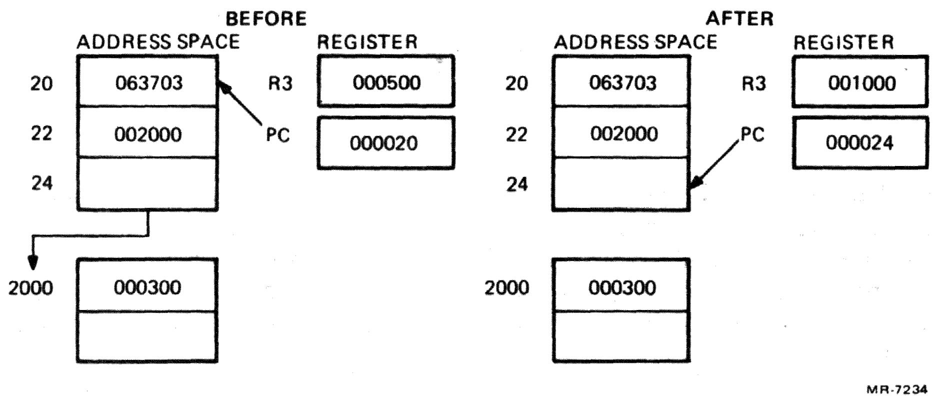


Figure 7-29 ADD @#2000,R3

7.2.5.3 Relative Addressing – The relative addressing mode is assembled as index mode using R7. The base of the address calculation, which is stored in the second or third word of the instruction, is not the address of the operand, but the number that, when added to the PC, becomes the address of the operand. This mode is useful for writing position independent code because the location referenced is always fixed relative to the PC. When instructions are to be relocated, the operand is moved by the same amount. The assembler interprets and assembles instructions in the following forms as relative addressing mode operations.

OPR A or **OPR X(PC)**

where X is the location of A relative to the instruction.

Relative Addressing Example (Figure 7-30)

Symbolic	Octal Code	Instruction Name	Operation
INC A	005267 000054	Increment	To increment location A, contents of memory location immediately following instruction word are added to (PC) to produce address A. Contents of A are increased by one.

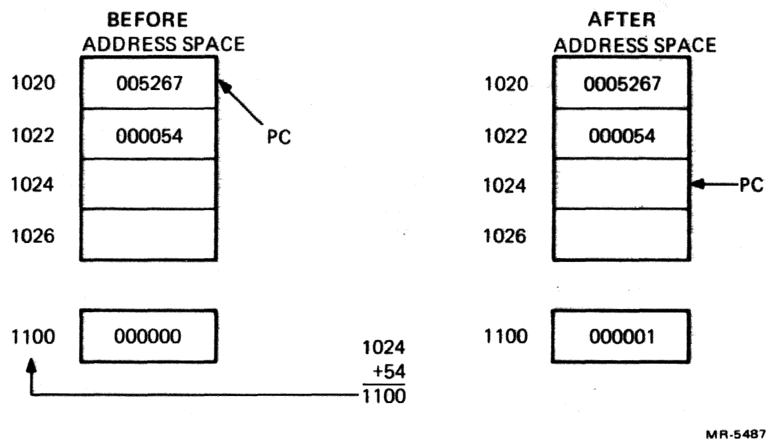


Figure 7-30 INC A

7.2.5.4 Relative Deferred Addressing – The relative deferred addressing mode is similar to the relative mode. However, the second word of the instruction, when added to the PC, contains the address of the address of the operand rather than the address of the operand. The assembler interprets and assembles instructions in the following forms as relative deferred addressing mode operations.

OPR @A or **OPR @X(PC)**

where X is the location containing the address of A relative to the instruction.

Relative Deferred Mode Example (Figure 7-31)

Symbolic	Octal Code	Instruction Name	Operation
CLR @A	005077 000020	Clear	The second word of instruction is added to updated PC to produce address of address of operand. The operand is cleared.

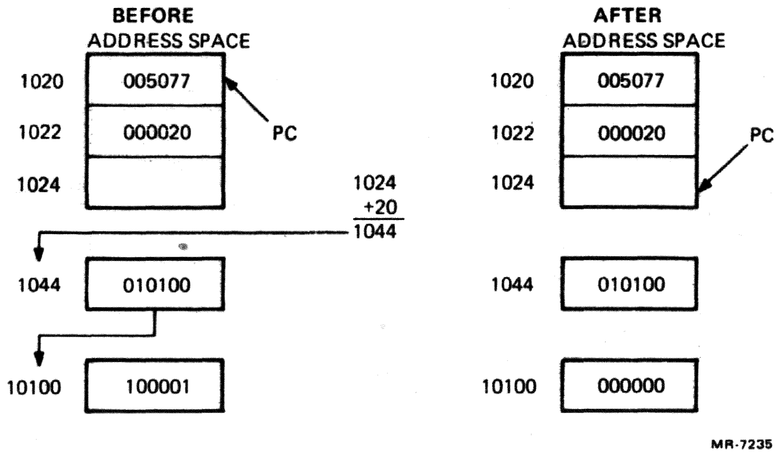


Figure 7-31 CLR @A

7.2.6 Use of the Stack Pointer as a General-Purpose Register

The processor stack pointer (SP, register R6) is the general-purpose register most often used for stack operations related to program nesting. Autodecrement with register R6 pushes data onto the stack, and autoincrement with register R6 pops data off the stack. Since the SP is used by the processor for interrupt handling, it has a special attribute: autoincrements and autodecrements are always done in steps of two. Byte operations using the SP in this way leave odd addresses unmodified.

7.3 INSTRUCTION SET

Specifications for each instruction in the SBC-11/21 PLUS instruction set follow and include each instruction's mnemonic, octal code, binary code, a diagram showing the format of the instruction, a symbolic notation describing its execution and effect on the condition codes, a description, special comments, and examples.

MNEMONIC: A mnemonic is indicated before each description. When the word instruction has a byte equivalent, the byte mnemonic is also shown.

INSTRUCTION FORMAT: A diagram accompanying each instruction shows the octal op code, binary op code, and bit assignments. In byte instructions, the most significant bit (bit 15) is always a one.

SYMBOLS: The following symbols are used in the instruction specifications.

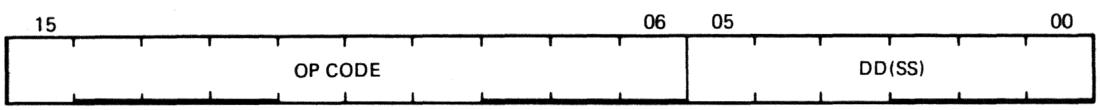
()	= contents of
SS or src	= source address
DD or dst	= destination address
loc	= location
←	= becomes
↑	= "is popped from stack"
↓	= "is pushed onto stack"

$\wedge$	= boolean AND
$\vee$	= boolean OR
$\forall$	= exclusive OR
~	= boolean not
Reg or R	= register
B	= Byte
■	= 0 for word 1 for byte
,	= concatenated

7.3.1 Instruction Formats

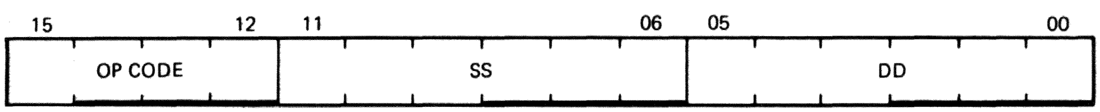
The following formats include all instructions used in the SBC-11/21 PLUS. Refer to individual instructions for more detailed information.

1. Single operand group: CLR, CLRB, COM, COMB, INC, INCB, DEC, DECB, NEG, NEGB, ADC, ADCB, SBC, SBCB, TST, TSTB, ROR, RORB, ROL, ROLB, ASR, ASRB, ASL, ASLB, JMP, SWAB, MFPS, MTPS, SXT, XOR



MR-5191

2. Double operand group: BIT, BITB, BIC, BICB, BIS, BISB, ADD, SUB, MOV, MOVB, CMP, CMPB



MR-5192

3. Program control group:

a. Branch (all branch instructions)

15 — 08	07 — 00
OP CODE	OFFSET

MR-5193

b. Jump to subroutine (JSR)

15 — 09	08 — 06	05 — 00
0 0 4	R	DD

MR-5194

c. Subroutine return (RTS)

15 — 03	02 — 00
0 0 0 2 0	R

MR-5195

d. Traps (breakpoint, IOT, EMT, TRAP, BPT)

15 — 00
OP CODE

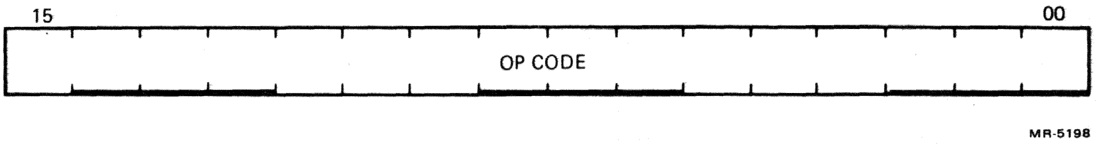
MR-5196

e. Subtract 1 and branch if = 0 (SOB)

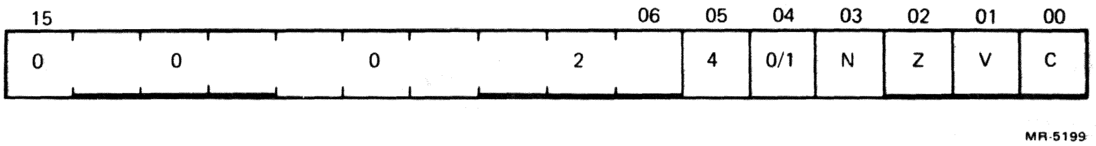
15 — 09	08 — 06	05 — 00
0 0 7	R	NN

MR-5197

4. Operate group: HALT, WAIT, RTI, RESET, RTT, NOP, MFPT



5. Condition code operators: (all condition code instructions)



Byte Instructions – The SBC-11/21 PLUS includes a full complement of instructions that manipulate byte operands. Because all microprocessor addressing is byte-oriented, byte manipulation addressing is straightforward. Byte instructions with autoincrement or autodecrement direct addressing cause the specified register to be modified by one to point to the next byte of data. Byte operations in register mode access the low-order byte of the specified register. These provisions enable the SBC-11/21 PLUS to perform as either a word or byte microprocessor. The numbering scheme for word and byte addresses in memory is illustrated in Figure 7-32.

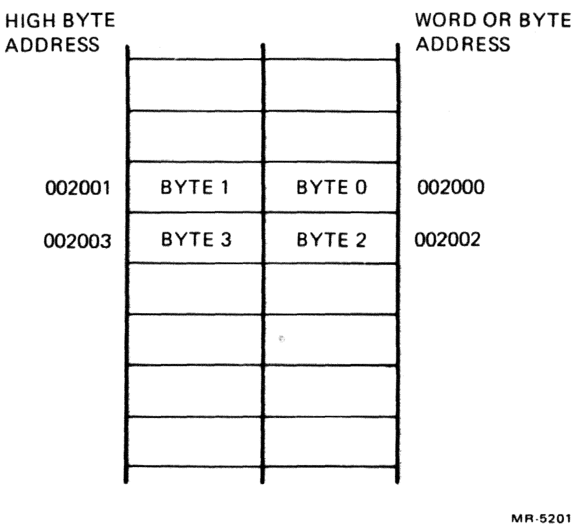


Figure 7-32 Byte Instructions

The most significant bit (bit 15) of the instruction word is set to indicate a byte instruction.

Byte Instruction Example

Symbolic	Octal Code	Instruction Name
CLR	0050DD	Clear word
CLRB	1050DD	Clear byte

7.3.2 List of Instructions

The SBC-11/21 PLUS instruction set is shown in Table 7-5.

Table 7-5 SBC-11/21 PLUS Instruction Set

Mnemonic	Instruction	Op Code
SINGLE OPERAND		
General		
CLR(B)	Clear dst	■050DD
COM(B)	Complement dst	■051DD
INC(B)	Increment dst	■052DD
DEC(B)	Decrement dst	■053DD
NEG(B)	Negate dst	■054DD
TST(B)	Test dst	■057DD
Shift & Rotate		
ASR(B)	Arithmetic shift right	■062DD
ASL(B)	Arithmetic shift left	■063DD
ROR(B)	Rotate right	■060DD
ROL(B)	Rotate left	■061DD
SWAB	Swap bytes	0003DD
Multiple Precision		
ADC(B)	Add carry	■055DD
SBC(B)	Subtract carry	■056DD
SXT	Sign extend	0067DD
PS Word Operators		
MFPS	Move byte from PS	1067DD
MTPS	Move byte to PS	1064SS
DOUBLE OPERAND		
General		
MOV(B)	Move source to destination	■1SSDD
CMP(B)	Compare src to dst	■2SSDD
ADD	Add src to dst	06SSDD
SUB	Subtract src from dst	16SSDD

Table 7-5 SBC-11/21 PLUS Instruction Set (Cont)

Mnemonic	Instruction	Op Code
Logical		
BIT(B)	Bit test	■3SSDD
BIC(B)	Bit clear	■4SSDD
BIS(B)	Bit set	■5SSDD
XOR	Exclusive OR	074RDD
PROGRAM CONTROL		
Branch		
BR	Branch (unconditional)	000400
BNE	Branch if not equal (to zero)	001000
BEQ	Branch if equal (to zero)	001400
BPL	Branch if plus	100000
BMI	Branch if minus	100400
BVC	Branch if overflow is clear	102000
BVS	Branch if overflow is set	102400
BCC	Branch if carry is clear	103000
BCS	Branch if carry is set	103400
Signed Conditional Branch		
BGE	Branch if greater than or equal (to zero)	002000
BLT	Branch if less than (zero)	002400
BGT	Branch if greater than (zero)	003000
BLE	Branch if less than or equal (to zero)	003400
Unsigned Conditional Branch		
BHI	Branch if higher	101000
BLOS	Branch if lower or same	101400
BHIS	Branch if higher or same	103000
BLO	Branch if lower	103400
Jump & Subroutine		
JMP	Jump	0001DD
JSR	Jump to subroutine	004RDD
RTS	Return from subroutine	00020R
SOB	Subtract one and branch (if $\neq$ 0)	077R00
Trap & Interrupt		
EMT	Emulator trap	104000–104377
TRAP	Trap	104400–104777
BPT	Breakpoint trap	000003
IOT	Input/output trap	000004
RTI	Return from interrupt	000002
RTT	Return from interrupt	000006

Table 7-5 SBC-11/21 PLUS Instruction Set (Cont)

Mnemonic	Instruction	Op Code
MISCELLANEOUS		
HALT	Halt	000000
WAIT	Wait for interrupt	000001
RESET	Reset external bus	000005
MFPT	Move processor type	000007
RESERVED INSTRUCTIONS		
		00021R
		00022R
CONDITION CODE OPERATORS		
CLC	Clear C	000241
CLV	Clear V	000242
CLZ	Clear Z	000244
CLN	Clear N	000250
CCC	Clear all CC bits	000257
SEC	Set C	000261
SEV	Set V	000262
SEZ	Set Z	000264
SEN	Set N	000270
SCC	Set all CC bits	000277
NOP	No operation	000240

7.3.3 Single Operand Instructions

NOTE

In most SBC-11/21 PLUS instructions, a write operation to a memory location or register is always preceded by a read operation from the same location except when writing PC and processor status (PS) to the stack in the following two cases.

1. **The execution of the microcode preceding an interrupt or trap service routine.**

2. **Interrupt and trap instructions:**

   HLT
   TRAP
   BPT
   IOT

7.3.3.1 General –

CLR
CLRB

Clear Destination ■050DD

15									06	05					00
0/1	0	0	0	1	0	1	0	0	0	d	d	d	d	d	d

MR-5202

Operation: (dst) ← 0

Condition Codes:

N: cleared
Z: set
V: cleared
C: cleared

Description:

Word: Contents of specified destination are replaced with zeros.
Byte: Same

Example:

CLR R1

Before	After
(R1) = 177777	(R1) = 000000
N Z V C	N Z V C
1 1 1 1	0 1 0 0

COM
COMB

Complement Destination ■051DD

15									06	05					00
0/1	0	0	0	1	0	1	0	0	1	d	d	d	d	d	d

MR-5203

Operation: (dst) ← ~ (dst)

Condition Codes:

N: set if most significant bit of result is set; cleared otherwise
Z: set if result = 0; cleared otherwise
V: cleared
C: set

Description: Word: The contents of the destination address are replaced by their logical complement (each bit equal to zero is set, and each bit equal to one is cleared).
Byte: Same

Example:

COM R0

Before	After
(R0) = 013333	(R0) = 164444
N Z V C	N Z V C
0 1 1 0	1 0 0 1

INC
INCB

Increment Destination ■052DD

15									06	05					00
0/1	0	0	0	1	0	1	0	1	0	d	d	d	d	d	d

MR-5204

Operation: (dst) ← (dst) + 1

Condition Codes:
N: set if result < 0; cleared otherwise
Z: set if result = 0; cleared otherwise
V: set if (dst) held 077777; cleared otherwise
C: not affected

Description: Word: One is added to contents of destination.
Byte: Same

Example:

INC R2

Before	After
(R2) = 000333	(R2) = 000334
N Z V C	N Z V C
0 0 0 0	0 0 0 0

DEC
DECB

Decrement Destination **■053DD**

15									06	05					00
0/1	0	0	0	1	0	1	0	1	1	d	d	d	d	d	d

MR-5205

Operation: (dst) ← (dst) −1

Condition Codes:

N: set if result < 0, cleared otherwise
Z: set is result = 0; cleared otherwise
V: set if (dst) was 100000; cleared otherwise
C: not affected

Description:

Word: One is subtracted from the contents of the destination.
Byte: Same

Example:

DEC R5

Before	After
(R5) = 000001	(R5) = 000000
N Z V C	N Z V C
1 0 0 0	0 1 0 0

NEG
NEGB

Negate Destination **■054DD**

15									06	05					00
0/1	0	0	0	1	0	1	1	0	0	d	d	d	d	d	d

MR-5206

Operation: (dst) ← − (dst)

Condition Codes:

N: set if the result < 0; cleared otherwise
Z: set if the result = 0; cleared otherwise
V: set if the result is 100000; cleared otherwise
C: cleared if the result is 0; set otherwise

Description:

Word: The contents of the destination address are replaced by its two's complement. 100000 is replaced by itself (in two's complement notation, the most negative number has no positive counterpart).
Byte: Same

Example:

NEG R0

Before	After
(R0) = 000010	(R0) = 177770
N Z V C	N Z V C
0 0 0 0	1 0 0 1

TST
TSTB

Test Destination ■057DD

15									06	05					00
0/1	0	0	0	1	0	1	1	1	1	d	d	d	d	d	d

MR-5207

Operation: (dst) ← (dst)

Condition Codes:

N: set if the result < 0; cleared otherwise
Z: set if result = 0; cleared otherwise
V: cleared
C: cleared

Description:

Word: The condition codes N and Z are set according to the contents of the destination address, and the contents of the destination remain unmodified.
Byte: Same

Example:

TST R1

Before	After
(R1) = 012340	(R1) = 012340
N Z V C	N Z V C
0 0 1 1	0 0 0 0

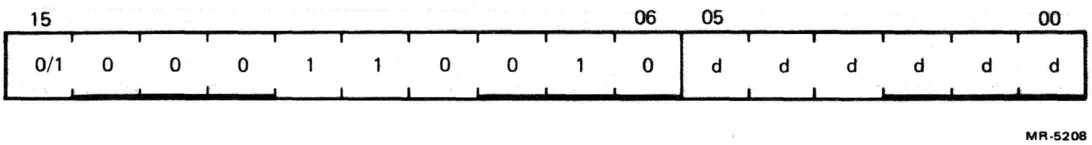
7.3.3.2 Shifts and Rotates – Scaling data by factors of two is accomplished with two shift instructions:

1. ASR – Arithmetic shift right
2. ASL – Arithmetic shift left

The sign bit (bit 15) of the operand is reproduced in shifts to the right. The low-order bit is filled with zero in shifts to the left. Bits shifted out of the C-bit, as shown in the following examples, are lost.

The rotate instructions operate on the destination word and the C-bit as though they formed a 17-bit circular buffer. These instructions facilitate sequential bit testing and detailed bit manipulation.

Arithmetic Shift Right ■062DD

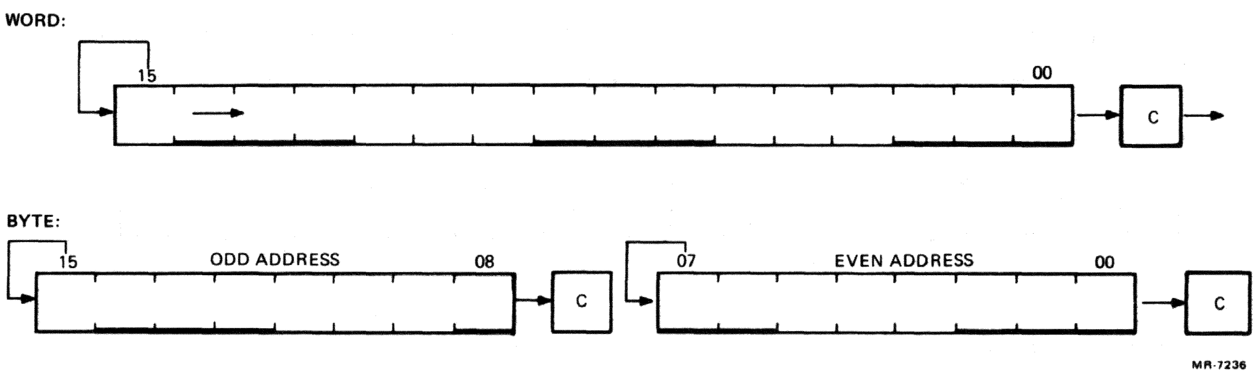


Operation: (dst) ← (dst) shifted one place to the right

Condition Codes:
N: set if the high-order bit of the result is set (result < 0); cleared otherwise
Z: set if the result = 0; cleared otherwise
V: loaded from the exclusive OR of the N-bit and C-bit (as set by the completion of the shift operation)
C: loaded from the low-order bit of the destination

Description:
Word: All bits of the destination are shifted right one place. Bit 15 is reproduced. The C-bit is loaded from bit 0 of the destination. ASR performs signed division of the destination by two.
Byte: Same

Example:



ASL
ASLB

Arithmetic Shift Left ■063DD

15									06	05					00
0/1	0	0	0	1	1	0	0	1	1	d	d	d	d	d	d

MR-5210

Operation: (dst) ← (dst) shifted one place to the left

Condition Codes:

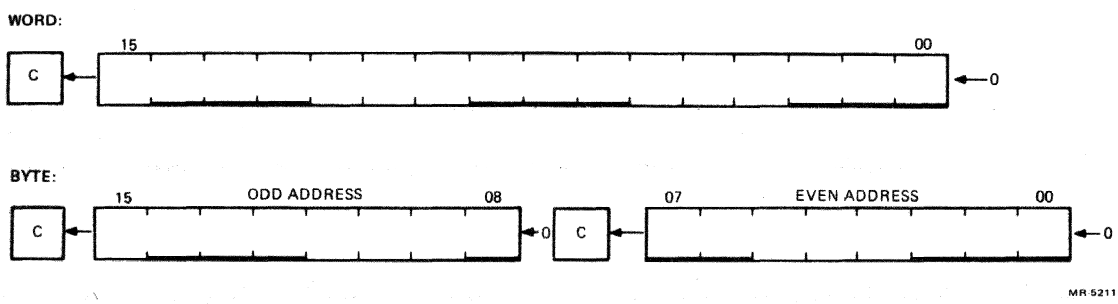
N: set if the high-order bit of the result is set (result < 0); cleared otherwise
Z: set if the result = 0; cleared otherwise
V: loaded with the exclusive OR of the N-bit and C-bit (as set by the completion of the shift operation)
C: loaded with the high-order bit of the destination

Description:

Word: All bits of the destination are shifted left one place. Bit 0 is loaded with a zero. The C-bit of the status word is loaded from the most significant bit of the destination. ASL performs signed multiplication of the destination by two with overflow indication.

Byte: Same

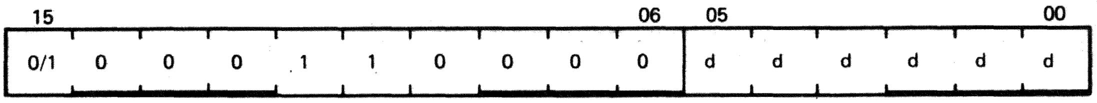
Example:



ROR
RORB

Rotate Right

■060DD



MR-5212

Operation: (dst) ← (dst) rotate right one place

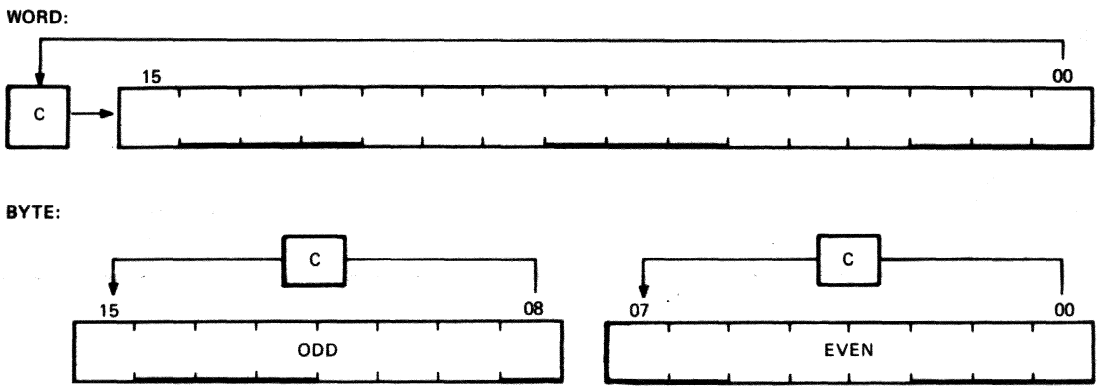
Condition Codes:

N: set if the high-order bit of the result is set (result < 0); cleared otherwise
Z: set if all bits of result = 0; cleared otherwise
V: loaded with the exclusive OR of the N-bit and C-bit (as set by the completion of the rotate operation)
C: loaded with the low-order bit of the destination

Description:

Word: All bits of the destination are rotated right one place. Bit 0 is loaded into the C-bit, and the previous contents of the C-bit are loaded into bit 15 of the destination.
Byte: Same

Example:



MR-5213

ROL
ROLB

Rotate Left ■061DD

15									06	05					00
0/1	0	0	0	1	1	0	0	0	1	d	d	d	d	d	d

MR-5214

Operation: (dst) ← (dst) rotate left one place

Condition Codes:

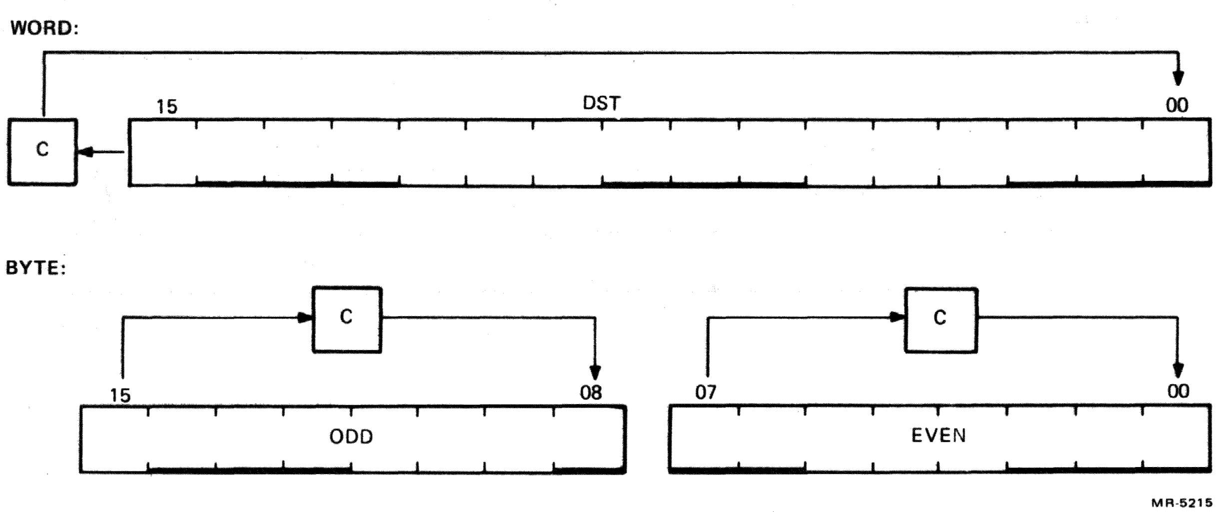
- N: set if the high-order bit of the result word is set (result < 0); cleared otherwise
- Z: set if all bits of the result word = 0; cleared otherwise
- V: loaded with the exclusive OR of the N-bit and C-bit (as set by the completion of the rotate operation)
- C: loaded with the high-order bit of the destination

Description:

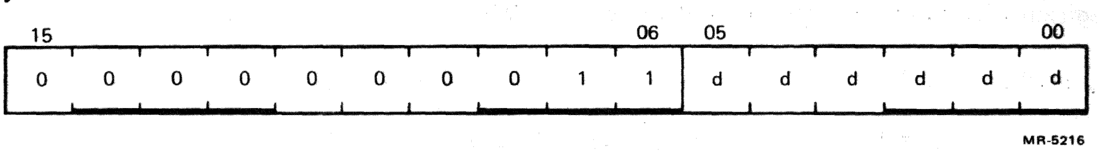
Word: All bits of the destination are rotated left one place. Bit 15 is loaded into the C-bit of the status word, and the previous contents of the C-bit are loaded into bit 0 of the destination.

Byte: Same

Example:



Swap Bytes 0003DD



Operation: Byte 1/Byte 0 ← Byte 0/Byte 1

Condition Codes:

N: set if the high-order bit of the low-order byte (bit 7) of the result is set; cleared otherwise
Z: set if low-order byte of result = 0; cleared otherwise
V: cleared
C: cleared

Description: High-order byte and low-order byte of the destination word are exchanged (destination must be a word address).

Example:

SWAB R1

Before	After
(R1) = 077777	(R1) = 177577
N Z V C	N Z V C
1 1 1 1	0 0 0 0

7.3.3.3 Multiple Precision – It is sometimes necessary to do arithmetic on operands considered as multiple words or bytes. The SBC-11/21 PLUS makes special provisions for such operations with the instructions ADC (add carry) and SBC (subtract carry) and their byte equivalents. For example, two 16-bit words may be combined into a 32-bit double precision word and added or subtracted as shown in Figure 7-33.

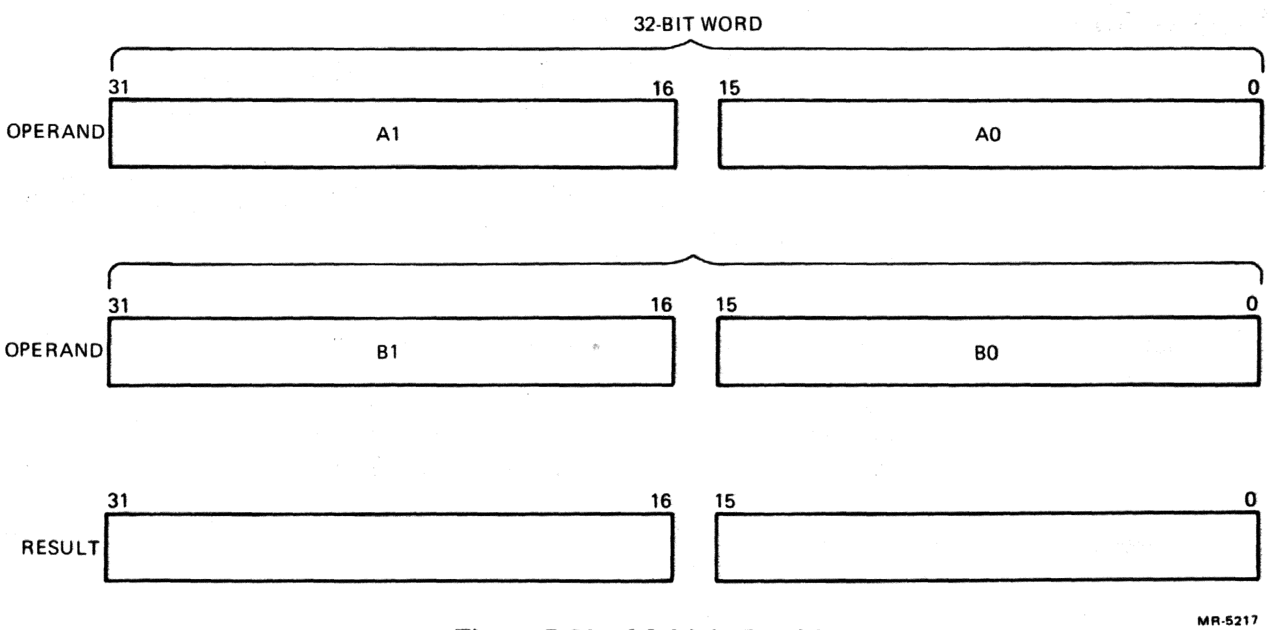


Figure 7-33 Multiple Precision

Multiple Precision Example

The addition of −1 and −1 could be performed as follows:

−1 = 37777777777

(R1) = 177777 (R2) = 177777 (R3) = 177777 (R4) = 177777

```
ADD R1,R2
ADC R3
ADD R4,R3
```

1. After (R1) and (R2) are added, 1 is loaded into the C-bit
2. ADC instruction adds C-bit to (R3); (R3) = 0
3. (R3) and (R4) are added
4. Result is 37777777776 or −2

ADC
ADCB

Add Carry ■055DD

15									06	05					00
0/1	0	0	0	1	0	1	1	0	1	d	d	d	d	d	d

MR-5218

Operation: (dst) ← (dst) + (C-bit)

Condition Codes:
N: set if result < 0; cleared otherwise
Z: set if result = 0; cleared otherwise
V: set if (dst) was 077777 and (C) was 1; cleared otherwise
C: set if (dst) was 177777 and (C) was 1; cleared otherwise

Description:
Word: The contents of the C-bit are added into the destination. This permits the carry from the addition of the low-order words to be carried into the high-order result.
Byte: Same

Example: Double precision addition is done with the following instruction sequence:

```
ADD A0,B0        add low-order parts
ADC B1           add carry into high order
ADD A1,B1        add high-order parts
```

SBC
SBCB

Subtract Carry ■056DD

15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
0/1	0	0	0	1	0	1	1	1	0	d	d	d	d	d	d

MR-5219

Operation: (dst) ← (dst) − (C)

Condition Codes:

N: set if result < 0; cleared otherwise
Z: set if result = 0; cleared otherwise
V: set if (dst) was 100000; cleared otherwise
C: set if (dst) was 0 and C was 1; cleared otherwise

Description:

Word: The contents of the C-bit are subtracted from the destination. This permits the carry from the subtraction of two low-order words to be subtracted from the high-order part of the result.

Byte: Same

Example: Double precision subtraction is done with the following instruction sequence:

```
SUB A0,B0
SBC B1
SUB A1,B1
```

SXT

Sign Extend 0067DD

15									06	05					00
0	0	0	0	1	1	0	1	1	1	d	d	d	d	d	d

MR-5220

Operation:

(dst) ← 0 if N-bit is clear
(dst) ← 1 if N-bit is set

Condition Codes:

N: unaffected
Z: set if N-bit is clear
V: cleared
C: unaffected

Description: If the condition code bit N is set, a −1 is placed in the destination operand; if the N-bit is clear, then a zero is placed in the destination operand. This instruction is particularly useful in multiple precision arithmetic because it permits the sign to be extended through multiple words.

Example: SXT A

Before	After
(A) = 012345	(A) = 177777
N Z V C	N Z V C
1 0 0 0	1 0 0 0

7.3.3.4 PS Word Operators –

MFPS

Move Byte from Processor Status (PS) 1067DD

15							08	07							00
1	0	0	0	1	1	0	1	1	1	d	d	d	d	d	d

MR-5221

Operation: (dst) ← PS
dst lower 8 bits

Condition Codes:
N: set if PS bit 7 = 1; cleared otherwise
Z: set if PS <0:7> = 0; cleared otherwise
V: cleared
C: not affected

Description: The 8-bit contents of the PS are moved to the effective destination. If the destination is mode 0, PS bit 7 is sign extended through the upper byte of the register. The destination operand address is treated as a byte address.

Example: MFPS R0

Before	After
R0 [0]	R0 [000014]
PS [000014]	PS [000000]

MTPS

Move Byte to Processor Status 1064SS

15							08	07							00
1	0	0	0	1	1	0	1	0	0	s	s	s	s	s	s

MR-5222

Operation: PS ← (src)

Condition Codes: Set according to effective source operand bits 0–3

Description: The 8 bits of the effective operand replace the current contents of the PS. The source operand address is treated as a byte address. The T-bit (PS bit 4) cannot be set with this instruction. The source operand remains unchanged. This instruction can be used to change the priority bits (PS bits 7–5) in the PS.

7.3.4 Double Operand Instructions

Double operand instructions save instructions and time because they eliminate the need for load and save sequences such as those used in accumulator-oriented machines.

7.3.4.1 General –

MOV
MOVB

Move Source to Destination ■1SSDD

15			12	11					06	05					00
0/1	0	0	1	s	s	s	s	s	s	d	d	d	d	d	d

MR-5223

Operation: (dst) ← (src)

Condition Codes:

N: set if (src) < 0; cleared otherwise
Z: set if (src) = 0; cleared otherwise
V: cleared
C: not affected

Description:

Word: The source operand is moved to the destination location. The previous contents of the destination are lost. The contents of the source address are not affected.

Byte: Same as MOV. The MOVB to a register (unique among byte instructions) extends the most significant bit of the low-order byte (sign extension). Otherwise, MOVB operates on bytes exactly as MOV operates on words.

Example:

MOV XXX,R1	loads register 1 with the contents of memory location; XXX represents a programmer-defined mnemonic used to represent a memory location
MOV #20,R0	loads the number 20 into register 0; '#' indicates that the value 20 is the operand
MOV @#20,–(R6)	pushes the operand contained in location 20 onto the stack
MOV (R6)+,@#177566	pops the operand off a stack and moves it into memory location 177566 (terminal print buffer)
MOV R1,R3	performs an inter-register transfer
MOVB @#177562,@#177566	moves a character from terminal keyboard buffer to terminal printer buffer

CMP
CMPB

Compare Source to Destination ■2SSDD

15			12	11					06	05					00
0/1	0	1	0	s	s	s	s	s	s	d	d	d	d	d	d

MR-5224

Operation: (src) – (dst)

Condition Codes:

N: set if result < 0; cleared otherwise
Z: set if result = 0; cleared otherwise
V: set if there was arithmetic overflow; that is, operands were of opposite signs and the sign of the destination was the same as the sign of the result; cleared otherwise
C: cleared if there was a carry from the most significant bit of the result; set otherwise

Description:

Word: The source and destination operands are compared, and the condition codes are set. The condition codes may then be used for arithmetic and logical conditional branches. Both operands are unaffected. The only action is to set the condition codes. The compare is customarily followed by a conditional branch instruction. Unlike the subtract instruction, the order of operation is (src) – (dst), not (dst) – (src).

Byte: Same

ADD

Add Source to Destination 06SSDD

15			12	11					06	05					00
0	1	1	0	s	s	s	s	s	s	d	d	d	d	d	d

MR-5225

Operation: (dst) ← (src) + (dst)

Condition Codes:

N: set if result < 0; cleared otherwise
Z: set if result = 0; cleared otherwise
V: set if there was arithmetic overflow as a result of the operation; that is, both operands were of the same sign and the result was of the opposite sign; cleared otherwise
C: set if there was a carry from the most significant bit of the result; cleared otherwise

Description:

Word: The source operand is added to the destination operand and the result is stored at the destination address. The original contents of the destination are lost. The contents of the source are not affected. Two's complement addition is performed.

Byte: There is no equivalent byte mode.

Example:

ADD 20,R0	add to register
ADD R1,XXX	add to memory
ADD R1,R2	add register to register
ADD @#17750,XXX	add memory to memory

XXX is a programmer-defined mnemonic for a memory location.

SUB

Subtract Source from Destination — 16SSDD

15			12	11					06	05					00
1	1	1	0	s	s	s	s	s	s	d	d	d	d	d	d

MR-5226

Operation: (dst) ← (dst) − (src)

Condition Codes:

N: set if result < 0; cleared otherwise
Z: set if result = 0; cleared otherwise
V: set if there was arithmetic overflow as a result of the operation, that is if operands were of opposite signs and the sign of the source was the same as the sign of the result; cleared otherwise
C: cleared if there was a carry from the most significant bit of the result; set otherwise

Description:

Word: The source operand is subtracted from the destination operand, and the result is left at the destination address. The original contents of the destination are lost. The contents of the source are not affected. In double-precision arithmetic, the C-bit, when set, indicates a borrow.

Byte: There is no equivalent byte mode.

Example:

SUB R1,R2

Before	After
(R1) = 011111	(R1) = 011111
(R2) = 012345	(R2) = 001234
N Z V C	N Z V C
1 1 1 1	0 0 0 0

7.3.4.2 Logical – Logical group instructions have the same format as the double operand arithmetic group. They permit operations on data at the bit level.

BIT
BITB

Bit Test ■3SSDD

15			12	11					06	05					00
0/1	0	1	1	s	s	s	s	s	s	d	d	d	d	d	d

MR-5227

Operation: (src) ∧ (dst)

Condition Codes:
N: set if high-order bit of result is set; cleared otherwise
Z: set if result = 0; cleared otherwise
V: cleared
C: not affected

Description:
Word: Logical "and" comparison of the source and destination operands is performed, and condition codes are modified accordingly. Neither the source nor destination is affected. The BIT instruction may be used either to test whether any of the corresponding bits that are set in the destination are also set in the source or whether all corresponding bits set in the destination are clear in the source.
Byte: Same

Example:
BIT #30,R3 test bits three and four of R3 to see if both are off

R3 = 0 000 000 000 011 000

Before	After
N Z V C	N Z V C
1 1 1 1	0 0 0 1

BIC
BICB

Bit Clear ■4SSDD

15			12	11					06	05					00
0/1	1	0	0	s	s	s	s	s	s	d	d	d	d	d	d

MR-5228

Operation: (dst) ← (dst) ~ (src)

Condition Codes:
N: set if high-order bit of result is set; cleared otherwise
Z: set if result = 0; cleared otherwise
V: cleared
C: not affected

Description: Word: Each bit in the destination that corresponds to a set bit in the source is cleared. The original contents of the destination are lost. The contents of the source are not affected.

Byte: Same

Example:

BIC R3,R4

Before	After
(R3) = 001234	(R3) = 001234
(R4) = 001111	(R4) = 000101
N Z V C	N Z V C
1 1 1 1	0 0 0 1

Before: (R3) = 0 000 001 010 011 100
(R4) = 0 000 001 001 001 001

After: (R4) = 0 000 000 001 000 001

BIS
BISB

Bit Set ■5SSDD

15			12	11					06	05					00
0/1	1	0	1	s	s	s	s	s	s	d	d	d	d	d	d

MR-5229

Operation: (dst) ← (dst) $\vee$ (src)

Condition Codes:
N: set if high-order bit of result is set, cleared otherwise
Z: set if result = 0; cleared otherwise
V: cleared
C: not affected

Description: Word: Inclusive OR operation is performed between the source and destination operands, and the result is left at the destination address (i.e., corresponding bits set in the source are set in the destination). The contents of the destination are lost.

Byte: Same

Example:

BIS R0,R1

Before	After
(R0) = 001234	(R0) = 001234
(R1) = 001111	(R1) = 001335
N Z V C	N Z V C
0 0 0 0	0 0 0 0

Before: (R0) = 0 000 001 010 011 100
(R1) = 0 000 001 001 001 001

After: (R1) = 0 000 001 011 011 101

XOR

Exclusive OR **074RDD**

15						09	08		06	05					00
0	1	1	1	1	0	0	r	r	r	d	d	d	d	d	d

MR-5230

Operation:	(dst) ← (dst) $\forall$ (Reg)
Condition Codes:	N: set if the result < 0; cleared otherwise Z: set if result = 0; cleared otherwise V: cleared C: unaffected
Description:	The exclusive OR of the register and destination operand is stored in the destination address. Contents of register are unaffected. Assembler format is: XOR R,D.

Example:

XOR R0,R2

Before	After
(R0) = 001234	(R0) = 001234
(R2) = 001111	(R2) = 000325
N Z V C	N Z V C
1 1 1 1	0 0 0 1

Before: (R0) = 0 000 001 010 011 100
(R2) = 0 000 001 001 001 001

After: (R2) = 0 000 000 011 010 101

7.3.5 Program Control Instructions

7.3.5.1 Branches – Program control instructions cause a branch to a location defined by the sum of the offset (multiplied by two) and the current contents of the program counter if:

1. The branch instruction is unconditional.

2. The branch instruction is conditional, and the conditions are met after testing the condition codes (NZVC).

The offset is the number of words from the current contents of the PC forward or backward. The current contents of the PC point to the word following the branch instruction.

Although the offset expresses a byte address, the PC is expressed in words. Before it is added to the PC, the offset is automatically multiplied by two and sign extended to express words. Bit 7 is the sign of the offset. If it is set, the offset is negative and the branch is done in the backward direction. Similarly, if bit 7 is not set, the offset is positive and the branch is done in the forward direction.

The 8-bit offset allows branching in the backward direction by 200_8 words (400 bytes) from the current PC, and in the forward direction by 177_8 words (376 bytes) from the current PC.

The microprocessor assembler handles address arithmetic for the user and computes and assembles the proper offset field for branch instructions in the following form.

Bxx loc

where Bxx is the branch instruction and loc is the address to which the branch is to be made. The assembler gives an error indication in the instruction if the permissible branch range is exceeded. Branch instructions have no effect on condition codes. Conditional branch instructions, where the branch condition is not met, are treated as NO OPs.

BR

Branch (Unconditional) 000400 Plus Offset

15							08	07 – 00
0	0	0	0	0	0	0	1	OFFSET

MR-5231

Operation: PC ← PC + (2 × offset)

Condition Codes: Unaffected

Description: A way of transferring program control within a range of -128_{10} to $+127_{10}$ words with a one-word instruction is provided.

New PC address = updated PC + (2 × offset)

Updated PC = address of branch instruction +2

Example: With the branch instruction at location 500, the following offsets apply.

New PC Address	Offset Code	Offset (decimal)
474	375	−3
476	376	−2
500	377	−1
502	000	0
504	001	+1
506	002	+2

BNE

Branch If Not Equal (to Zero) — 001000 Plus Offset

15							08	07							00
0	0	0	0	0	0	1	0	OFFSET							

MR-5232

Operation: PC ← PC + (2 × offset) if Z = 0

Condition Codes: Unaffected

Description: The state of the Z-bit is tested, and a branch is caused if the Z-bit is clear. BNE is the complementary operation to BEQ. BNE is used to test inequality following a CMP, to test that some bits set in the destination were also in the source following a BIT operation, and generally, to test that the result of the previous operation was not zero.

Example:

```
CMP A,B        compare A and B
BNE C          branch if they are not equal
```

will branch to C if A $\neq$ B

and the sequence

```
ADD A,B        add A to B
BNE C          branch if the result is not equal to 0
```

will branch to C if A + B = 0

BEQ

Branch If Equal (to Zero) — 001400 Plus Offset

15							08	07							00
0	0	0	0	0	0	1	1	OFFSET							

MR-5233

Operation: PC ← PC + (2 × offset) if Z = 1

Condition Codes: Unaffected

Description: The state of the Z-bit is tested and a branch is caused if Z is set. BEQ is used to test equality following a CMP operation, to test that no bits set in the destination were also set in the source following a BIT operation, and generally, to test that the result of the previous operation was zero.

Example:

CMP A,B	compare A and B
BEQ C	branch if they are equal

will branch to C if A = B $(A - B = 0)$

and the sequence

ADD A,B	add A to B
BEQ C	branch if the result = 0

will branch to C if $A + B = 0$

BPL

Branch If Plus — 100000 Plus Offset

15							08	07–00
1	0	0	0	0	0	0	0	OFFSET

MR-5234

Operation: $PC \leftarrow PC + (2 \times \text{offset})$ if $N = 0$

Condition Codes: Unaffected

Description: The state of the N-bit is tested, and a branch is caused if N is clear (positive result). BPL is the complementary operation of BMI.

BMI

Branch If Minus — 100400 Plus Offset

15							08	07–00
1	0	0	0	0	0	0	1	OFFSET

MR-5235

Operation: $PC \leftarrow PC + (2 \times \text{offset})$ if $N = 1$

Condition Codes: Unaffected

Description: The state of the N-bit is tested, and a branch is caused if N is set. BMI is used to test the sign (most significant bit) of the result of the previous operation, branching if negative. BMI is the complementary function of BPL.

BVC

Branch If Overflow Is Clear — 102000 Plus Offset

15							08	07	00
1	0	0	0	0	1	0	0	OFFSET	

MR-5236

Operation: PC ← PC + (2 × offset) if V = 0

Condition Codes: Unaffected

Description: The state of the V-bit is tested, and a branch is caused if the V-bit is clear. BVC is the complementary operation to BVS.

BVS

Branch If Overflow Is Set — 102400 Plus Offset

15							08	07	00
1	0	0	0	0	1	0	1	OFFSET	

MR-5237

Operation: PC ← PC + (2 × offset) if V = 1

Condition Codes: Unaffected

Description: The state of the V-bit (overflow) is tested, and a branch is caused if the V-bit is set. BVS is used to detect arithmetic overflow in the previous operation.

BCC

Branch If Carry Is Clear — 103000 Plus Offset

15							08	07	00
1	0	0	0	0	1	1	0	OFFSET	

MR-5238

Operation: PC ← PC + (2 × offset) if C = 0

Condition Codes: Unaffected

Description: The state of the C-bit is tested, and a branch is caused if C is clear. BCC is the complementary operation to BCS.

Branch If Carry Is Set — 103400 Plus Offset

15							08	07 … 00
1	0	0	0	0	1	1	1	OFFSET

MR-5239

Operation: $PC \leftarrow PC + (2 \times \text{offset})$ if $C = 1$

Condition Codes: Unaffected

Description: The state of the C-bit is tested, and a branch is caused if C is set. BCS is used to test for a carry in the result of a previous operation.

7.3.5.2 Signed Conditional Branches – Particular combinations of the condition code bits are tested with the signed conditional branches. These instructions are used to test the results of instructions in which the operands were considered as signed (two's complement) values.

The sense of signed comparisons differs from unsigned comparisons. In signed 16-bit (two's complement) arithmetic, the sequence of values is as follows.

largest	077777
	077776
positive	.
	.
	.
	000001
zero	000000
	177777
	177776
	.
negative	.
	.
	100001
smallest	100000

In unsigned 16-bit arithmetic, the sequence is as follows.

highest	177777
	.
	.
	.
	.
	.
	.
	000002
	000001
lowest	000000

BGE

Branch If Greater Than or Equal (to Zero) 002000 Plus Offset

15							08	07 … 00
0	0	0	0	0	1	0	0	OFFSET

MR-5240

Operation: $PC \leftarrow PC + (2 \times \text{offset})$ if $N \forall V = 0$

Condition Codes: Unaffected

Description: A branch is caused if N and V are either both clear or both set. BGE is the complementary operation to BLT. Thus, BGE will always cause a branch when it follows an operation that caused addition of two positive numbers. BGE will also cause a branch on a zero result.

BLT

Branch If Less Than (Zero) 002400 Plus Offset

15							08	07 … 00
0	0	0	0	0	1	0	1	OFFSET

MR-5241

Operation: $PC \leftarrow PC + (2 \times \text{offset})$ if $N \forall V = 1$

Condition Codes: Unaffected

Description: A branch is caused if the exclusive OR of the N- and V-bits is one. Thus, BLT will always branch following an operation that added two negative numbers, even if overflow occurred. In particular, BLT will always cause a branch if it follows a CMP instruction operating on a negative source and a positive destination (even if overflow occurred). Further, BLT will never cause a branch when it follows a CMP instruction operating on a positive source and negative destination. BLT will not cause a branch if the result of the previous operation was zero (without overflow).

BGT

Branch If Greater Than (Zero) — 003000 Plus Offset

15							08	07–00
0	0	0	0	0	1	1	0	OFFSET

MR-5242

Operation: $PC \leftarrow PC + (2 \times \text{offset})$ if $Z \vee (N \veebar V) = 0$

Condition Codes: Unaffected

Description: Operation of BGT is similar to BGE, however, BGT will not cause a branch on a zero result.

BLE

Branch If Less Than or Equal (to Zero) — 003400 Plus Offset

15							08	07–00
0	0	0	0	0	1	1	1	OFFSET

MR-5243

Operation: $PC \leftarrow PC + (2 \times \text{offset})$ if $Z \vee (N \veebar V) = 1$

Condition Codes: Unaffected

Description: Operation is similar to BLT, however, BLE also will cause a branch if the result of the previous operation was zero.

7.3.5.3 Unsigned Conditional Branches – The unsigned conditional branches provide a means to test the results of comparison operations in which the operands are considered unsigned values.

BHI

Branch If Higher — 101000 Plus Offset

15							08	07–00
1	0	0	0	0	0	1	0	OFFSET

MR-5244

Operation: $PC \leftarrow PC + (2 \times \text{offset})$ if $C = 0$ and $Z = 0$

Condition Codes: Unaffected

Description: A branch occurs if the previous operation did not cause a carry or a zero result. This will happen in comparison (CMP) operations as long as the source has a higher unsigned value than the destination.

BLOS

Branch If Lower or Same — 101400 Plus Offset

15							08	07 … 00
1	0	0	0	0	0	1	1	OFFSET

MR-5245

Operation: PC ← PC + (2 × offset) if C ∨ Z = 1

Condition Codes: Unaffected

Description: A branch occurs if the previous operation caused either a carry or a zero result. BLOS is the complementary operation to BHI. The branch will occur in comparison operations as long as the source is equal to, or has a lower unsigned value than the destination.

BHIS

Branch If Higher or Same — 103000 Plus Offset

15							08	07 … 00
1	0	0	0	0	1	1	0	OFFSET

MR-5246

Operation: PC ← PC + (2 × offset) if C = 0

Condition Codes: Unaffected

Description: BHIS is the same instruction as BCC. This mnemonic is included for convenience only.

BLO

Branch If Lower — 103400 Plus Offset

15							08	07 … 00
1	0	0	0	0	1	1	1	OFFSET

MR-5247

Operation: PC ← PC + (2 × offset) if C = 1

Condition Codes: Unaffected

Description: BLO is the same instruction as BCS. This mnemonic is included for convenience only.

7.3.5.4 Jump and Subroutine Instructions – The subroutine call in the microprocessor provides for automatic nesting of subroutines, re-entrance, and multiple entry points. Subroutines may call other subroutines (or themselves) to any level of nesting without making special provisions for storage of return addresses at each level of subroutine call. The subroutine calling mechanism does not modify any fixed location in memory, and thus, provides for re-entrance. This allows one copy of a subroutine to be shared among several interrupting processes.

JMP

Jump **0001DD**

15									06	05					00
0	0	0	0	0	0	0	0	0	1	d	d	d	d	d	d

MR-5248

Operation: PC ← (dst)

Condition Codes: Unaffected

Description: More flexible program branching than that available with the branch instructions is provided. Control may be transferred to any location in memory (no range limitation) and can be accomplished with the full flexibility of the addressing modes, with the exception of register mode 0. Execution of a jump with mode 0 will cause an illegal instruction condition, and will cause the CPU to trap to vector address 4. (Program control cannot be transferred to a register.) Register deferred mode is legal and will cause program control to be transferred to the address held in the specified register. Instructions are word data and therefore, must be fetched from an even-numbered address.

Deferred index mode JMP instructions permit transfer of control to the address contained in a selectable element of a table of dispatch vectors.

Example:

	JMP FIRST	transfers to FIRST
First:		
	JMP @LIST	transfers to location pointed to at LIST
List:	FIRST	pointer to FIRST
	JMP @(SP)+	transfers to location pointed to by the top of the stack and removes the pointer from the stack

---

JSR

Jump to Subroutine 004RDD

15						09	08		06	05					00
0	0	0	0	1	0	0	r	r	r	d	d	d	d	d	d

MR-5249

Operation: (tmp) $\leftarrow$ (dst) (tmp is an internal processor register)

$\downarrow$ (SP) $\leftarrow$ reg (push reg contents onto processor stack)

reg $\leftarrow$ PC (PC holds location following JSR; this address is now put in reg)

PC $\leftarrow$ (dst) (PC now points to subroutine destination)

Condition Codes: Unaffected

Description: The old contents of the specified register (the linkage pointer) are automatically pushed onto the processor stack, and new linkage information is placed in the register. Thus, subroutines nested within subroutines to any depth may all be called with the same linkage register. There is no need either to plan the maximum depth at which any particular subroutine will be called or to include instructions in each routine to save and restore the linkage pointer. Further, since all linkages are saved in a re-entrant manner on the processor stack, execution of a subroutine may be interrupted, and the same subroutine re-entered and executed by an interrupt service routine. Execution of the initial subroutine can then be resumed when other requests are satisfied. This process (called nesting) can proceed to any level.

A subroutine called with a JSR reg,(dst) instruction can access the arguments following the call with either autoincrement addressing, (reg)+, (if arguments are accessed sequentially) or by indexed addressing, X(reg), (if arguments are accessed in random order). These addressing modes may also be deferred, @(reg)+ and @X(reg), if the parameters are operand addresses rather than the operands themselves.

JSR PC,(dst) is a special case of the microprocessor subroutine call and is used for subroutine calls that transmit parameters through the general-purpose registers. The SP and the PC are the only registers that may be modified by this call.

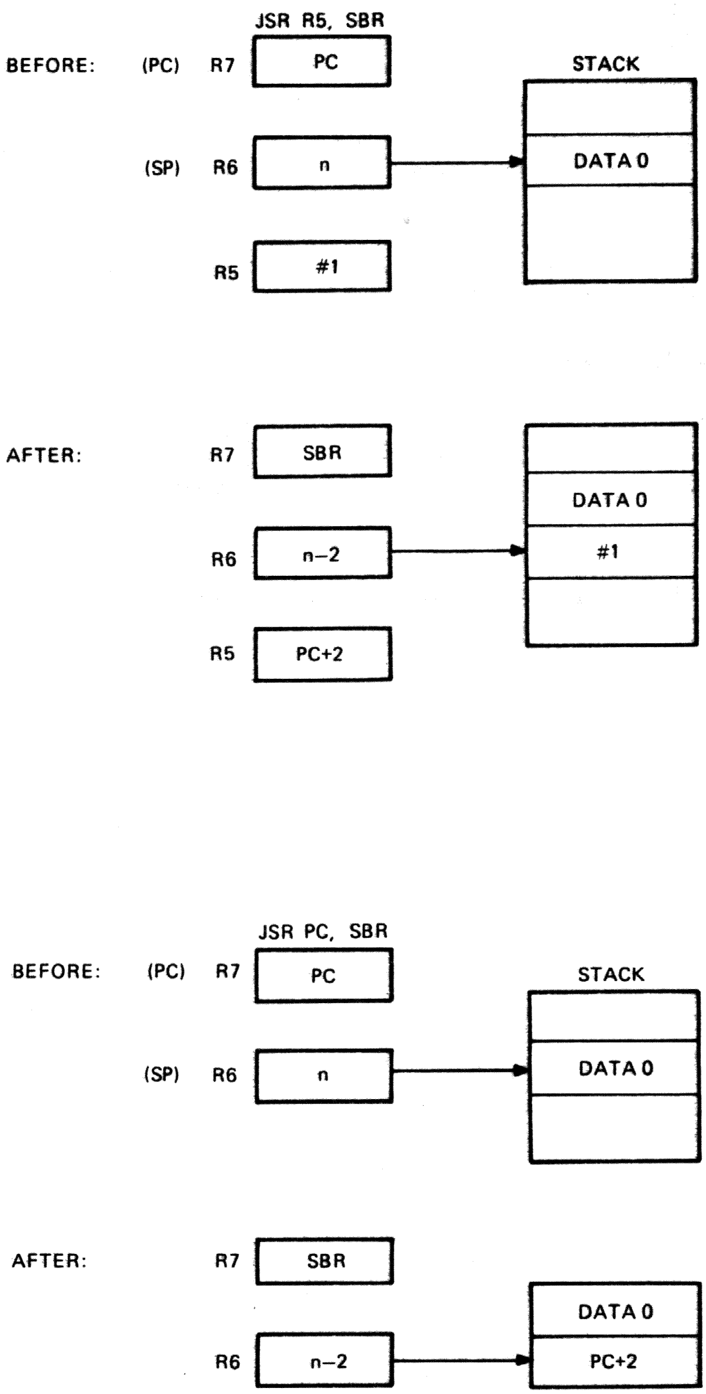
JSR PC,@(SP)+ is another special case of the JSR instruction. It exchanges the top element of the processor stack and the contents of the program counter. This instruction is used to allow two routines to swap program control and resume operation when recalled where they left off. Such routines are called co-routines.

Return from a subroutine is done with the RTS instruction. RTS reg loads the contents of reg into the PC and pops the top element of the processor stack into the specified register.

Example:

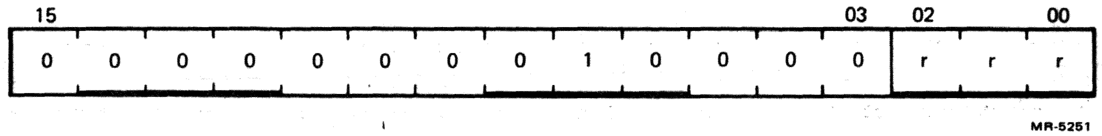
```
                                                  R5              R6    R7
SBCALL        : JSR R5, SBR                       #1              n     SBCALL
SBCALL+4      : ARG 1
                ARG 2
                  .
                  .
SBCALL+2+2M   : ARG M
CONT          : NEXT INSTRUCTION                  #1              n     CONT
                  .
                  .
SBR           : MOV(R5)+,dst1                     SBCALL+4        n-2   SBR
                MOV(R5)+,dst2
                  .
                MOV(R5)+,dst2
                  .
                MOV(R5)+,dstM                     SBCALL+2+2M
                OTHER INSTRUCTIONS                CONT
EXIT          : RTS R5                            CONT            n-2   EXIT
```

This example is illustrated in the following figure.


JSR R5, SBR
BEFORE:
(PC) R7
PC
STACK
(SP) R6
n
DATA 0
R5
#1
AFTER:
R7
SBR
DATA 0
R6
n–2
#1
R5
PC+2
JSR PC, SBR
BEFORE:
(PC) R7
PC
STACK
(SP) R6
n
DATA 0
AFTER:
R7
SBR
DATA 0
R6
n–2
PC+2

MR-5250

Return from Subroutine **00020R**

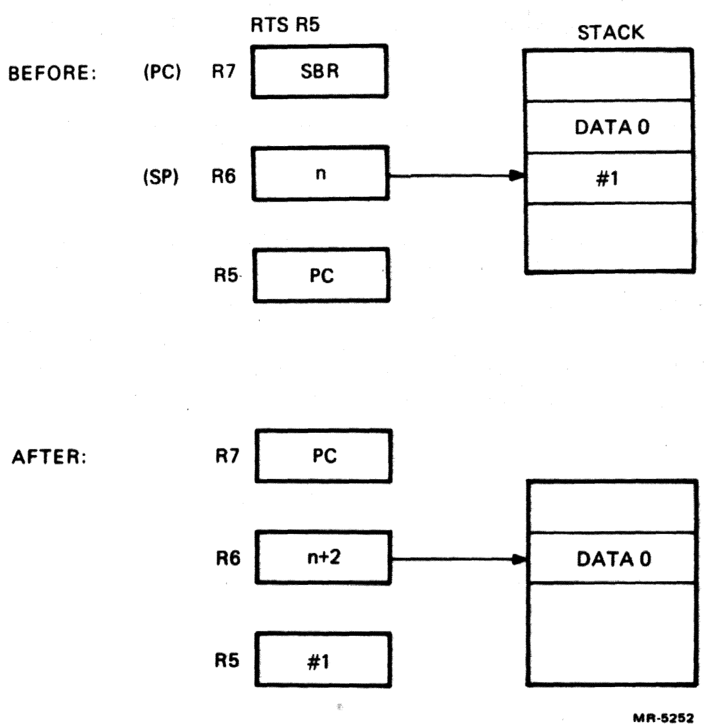


Operation: PC ← (reg)
(reg) ← (SP) ↑

Condition Codes: Unaffected

Description: Contents of register are loaded into PC, and the top element of the processor stack is popped into the specified register. Return from a nonre-entrant subroutine is typically made through the same register that was used in its call. Thus, a subroutine called with a JSR PC,(dst) exits with an RTS PC. A subroutine called with a JSR R5,(dst) may pick up parameters with addressing modes (R5)+, X(R5), or @X(R5) and finally exit with an RTS R5.

Example: RTS R5



SOB

Subtract One and Branch (If $\neq 0$) 077RNN

15						09	08		06	05 … 00
0	1	1	1	1	1	1	r	r	r	OFFSET

MR-5253

Operation: (R) ← (R) − 1; if this result $\neq$ 0 then PC ← PC − (2 × offset); if (R) = 0 then PC ← PC

Condition Codes: Unaffected

Description: The register is decremented. If it is not equal to zero, twice the offset is subtracted from the PC (now pointing to the following word). The offset is interpreted as a 6-bit positive number. SOB provides a fast, efficient method of loop control. The assembler syntax is:

SOB R,A

where A is the address to which transfer is to be made if the decremented R is not equal to zero. The SOB instruction cannot be used to transfer control in the forward direction.

7.3.5.5 Traps – Trap instructions provide for calls to emulators, I/O monitors, debugging packages, and user-defined interpreters. A trap is effectively an interrupt generated by software. When a trap occurs the contents of the current program counter (PC) and processor status (PS) are pushed onto the processor stack and replaced by the contents of a two-word trap vector containing a new PC and PS. The return sequence from a trap involves executing an RTI or RTT instruction that restores the old PC and PS by popping them from the stack. Trap instruction vectors are located at permanently assigned fixed addresses.

EMT

Emulator Trap 104000–104377

15							08	07 … 00
1	0	0	0	1	0	0	0	

MR-5254

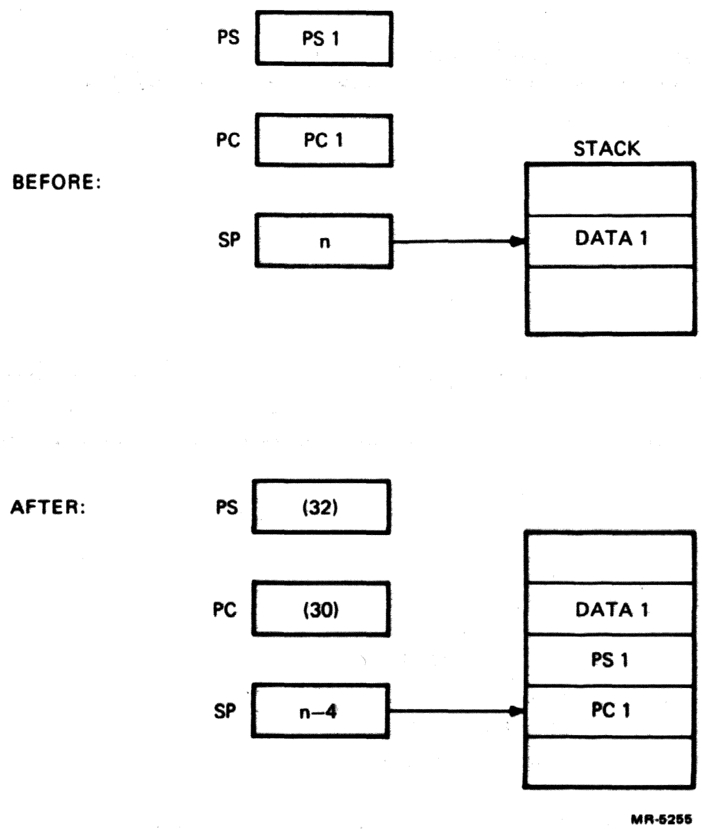
Operation: ↓(SP) ← PS
↓(SP) ← PC
PC ← (30)
PS ← (32)

Condition Codes:
N: loaded from trap vector
Z: loaded from trap vector
V: loaded from trap vector
C: loaded from trap vector

Description: All operation codes from 104000 to 104377 are EMT instructions and may be used to transmit information to the emulating routine (e.g., function to be performed). The trap vector for EMT is at address 30. The new PC is taken from the word at address 30, and the new processor status (PS) is taken from the word at address 32.

CAUTION
EMT is used frequently by Digital system software and is not recommended for general use.

Example:



TRAP

Trap 104400–104777

15							08	07							00
1	0	0	0	1	0	0	1								

MR-5256

Operation:

$\downarrow$ (SP) ← PS
$\downarrow$ (SP) ← PC
PC ← (34)
PS ← (36)

Condition Codes:

N: loaded from trap vector
Z: loaded from trap vector
V: loaded from trap vector
C: loaded from trap vector

Description:

Operation codes from 104400 to 104777 are TRAP instructions. TRAP and EMT instructions are identical in operation, however, the trap vector for TRAP is at address 34.

NOTE

Because Digital software makes frequent use of EMT, the TRAP instruction is recommended for general use.

BPT

Breakpoint Trap 000003

15															00
0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1

MR-5257

Operation:

$\downarrow$ (SP) ← PS
$\downarrow$ (SP) ← PC
PC ← (14)
PS ← (16)

Condition Codes:

N: loaded from trap vector
Z: loaded from trap vector
V: loaded from trap vector
C: loaded from trap vector

Description:

A trap sequence with a trap vector address of 14 is performed. BPT is used to call debugging aids. The user is cautioned against employing code 000003 in programs run under these debugging aids.

(No information is transmitted in the low byte.)

IOT

Input/Output Trap 000004

15															00
0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0

MR-5258

Operation:

$\downarrow$ (SP) $\leftarrow$ PS
$\downarrow$ (SP) $\leftarrow$ PC
PC $\leftarrow$ (20)
PS $\leftarrow$ (22)

Condition Codes:

N: loaded from trap vector
Z: loaded from trap vector
V: loaded from trap vector
C: loaded from trap vector

Description:

A trap sequence with a trap vector address of 20 is performed.

(No information is transmitted in the low byte.)

RTI

Return from Interrupt 000002

15															00
0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	0

MR-5260

Operation:

PC $\leftarrow$ (SP) $\uparrow$
PS $\leftarrow$ (SP) $\uparrow$

Condition Codes:

N: loaded from processor stack
Z: loaded from processor stack
V: loaded from processor stack
C: loaded from processor stack

Description:

Used to exit from an interrupt or TRAP service routine. The PC and PS are restored (popped) from the processor stack. If a trace trap is pending, the first instruction after RTI will not be executed prior to the next T trap.

RTT

Return from Trap 000006

15															00
0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0

MR-5259

Operation: $PC \leftarrow (SP)\uparrow$
$PS \leftarrow (SP)\uparrow$

Condition Codes:
N: loaded from processor stack
Z: loaded from processor stack
V: loaded from processor stack
C: loaded from processor stack

Description: Operation is the same as RTI, however, RTT inhibits a trace trap while RTI permits a trace trap. If new PS has T-bit set, trap will occur after execution of first instruction after RTT.

7.3.5.6 Reserved Instruction Traps – Reserved instruction traps are caused by attempts to execute instruction codes reserved for future processor expansion (reserved instructions) or instructions with illegal addressing modes (illegal instructions). Order codes not corresponding to any of the instructions described are reserved instructions. JMP and JSR with register mode destinations are illegal instructions and trap to vector address 4. Reserved instructions trap to vector address 10.

7.3.5.7 HALT Interrupt – The HALT interrupt is caused by the –HALT line. The –HALT interrupt saves the PC and PS and goes to the restart address with PS = 340.

7.3.5.8 Trace Trap – The trace trap is enabled by bit 4 of the PS and causes processor traps at the end of instruction execution. The instruction that is executed after the instruction that set the T-bit will proceed to completion and then trap through the trap vector at address 14. The trace trap is a system debugging aid and is transparent to the general programmer.

7.3.5.9 Power Failure Interrupt – The power failure interrupt occurs when –PF line is asserted. Vectors for power failure are locations 24 and 26. Trap will occur if an RTI instruction is executed in a power fail service routine.

7.3.5.10 Interrupts – See Table 5-3.

NOTE

Bit 4 of the processor status can only be set indirectly by executing an RTI or RTT instruction with the desired PS on the stack.

7.3.5.11 Special Cases (T-bit) – The following are special cases of the T-bit.

NOTE
The traced instruction follows the instruction that sets the T-bit.

1. An instruction that cleared the T-bit – Upon fetching the traced instruction, an internal flag, the trace flag, was set. The trap will still occur at the end of execution of this instruction. The status word on the stack, however, will have a clear T-bit.

2. An instruction that set the T-bit – Because the T-bit was already set, setting it again has no effect. The trap will occur.

3. An instruction that caused an instruction trap – The instruction trap is performed, and the entire routine for the service trap is executed. If the service routine exits with an RTI or in any other way restores the stacked status word, the T-bit is set again, the instruction following the traced instruction is executed, and, unless it is one of the special cases noted previously, a trace trap occurs.

4. Interrupt trap priorities – When multiple trap and interrupt conditions occur simultaneously, the following order of priorities is observed (from high to low).

   1. Halt line
   2. Power fail trap
   3. Trace trap
   4. Internal interrupt request
   5. External interrupt request
   6. Instruction traps

7.3.6 Miscellaneous Instructions

HALT

Halt **000000**

15															00
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

MR-5261

Operation:

$\downarrow$ (SP) ← PS
$\downarrow$ (SP) ← PC
PC ← restart address
PS ← 340

Condition Codes: Unaffected

Description: The processor goes to the restart address after placing the current PC and PS on the stack. PS is initialized to 340.

WAIT

Wait for Interrupt 000001

15															00
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1

MR-5262

Condition Codes: Unaffected

Description: In WAIT, as in all instructions, the PC points to the next instruction following the WAIT instruction. Thus, when an interrupt causes the PC and PS to be pushed onto the processor stack, the address of the next instruction following the WAIT is saved. The exit from the interrupt routine (i.e., execution of an RTI instruction) will cause resumption of the interrupted process at the instruction following the WAIT.

RESET

Reset External Bus 000005

15															00
0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	1

MR-5263

Condition Codes: Unaffected

Description: The −BCLR line is asserted and the mode register is loaded. −BCLR is negated, and an ASPI transaction takes place. PC, PS, and R0–R5 are not affected.

MFPT

Move from Processor Type Word 000007

15															00
0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1

MR-7198

Operation: $R0 \leftarrow 4$

Condition Codes: Unaffected

Description: The number four is placed in R0 telling the system software that the processor type is Micro/T-11.

7.3.7 Condition Code Operators

CLN SEN
CLZ SEZ
CLV SEV
CLC SEC
CCC SCC

Condition Code Operators **0002XX**

15										05	04	03	02	01	00
0	0	0	0	0	0	0	0	1	0	1	0/1	N	Z	V	C

MR-5266

Description: Condition code bits are set and cleared. Selectable combinations of these bits may be cleared or set together. Condition code bits corresponding to bits in the condition code operator (bits 0–3) are modified according to the sense of bit 4, the set/clear bit of the operator (i.e., set the bit specified by bit 0, 1, 2, or 3, if bit 4 is a one). Corresponding bits are cleared if bit 4 = 0.

Mnemonic	Operation	OP Code
CLC	Clear C	000241
CLV	Clear V	000242
CLZ	Clear Z	000244
CLN	Clear N	000250
SEC	Set C	000261
SEV	Set V	000262
SEZ	Set Z	000264
SEN	Set N	000270
SCC	Set all CCs	000277
CCC	Clear all CCs	000257
	Clear V and C*	000243
NOP	No operation	000240

*Combinations of the above set or clear operations may be ORed together to form combined instructions. Clear V and C represents CLC (241) ORed with CLV (code 242).

CHAPTER 8
THEORY OF OPERATION

8.1 INTRODUCTION
This chapter provides an explanation of SBC-11/21 PLUS hardware operation from the perspective of the logic designer. It is useful for troubleshooting the device to the chip level.

NOTE

The negated or inverse signal is designated by a minus sign (–). For example, RAS is normally low and asserted high when activated; –RAS is normally high and asserted low when activated. This convention is used throughout this chapter. The LSI-11 bus signals are consistent with the standard bus conventions. All gates prefaced with a G designation exist within the gate array (E2O).

The SBC-11/21 PLUS functional block diagram is shown in Figure 8-1 (sheets 1 and 2) and provides an overview of the module functions and how they are related. The main components of the single-board computer are shown on sheet 1 of Figure 8-1. The single-board computer has a microprocessor interconnected to the serial line units, RAM memory, ROM memory, and the parallel I/O interface via the on-board TDAL bus. The TDAL bus can access the LSI-11 bus (BDAL bus) by the bus control function, shown by broken lines, and is for reference only. The address bus, the memory address decode function, and the interrupt control function are also shown on sheet 1 of Figure 8-1.

The microprocessor support functions and the LSI-11 interface functions are described on sheet 2 of Figure 8-1. The microprocessor is shown by broken lines for reference only. The power-up, clock, clock control, ready, DMA, and halt functions are used by the microprocessor. The IAK data in, sync, read/write, reply time-out, and bus control functions are used to interface the LSI-11 bus to the microprocessor.

The functional descriptions used in this chapter define the microprocessor and the input/output signals associated with its operation. The support functions, the LSI-11 bus interface functions, and the remaining single-board computer devices are also described in detail.

8.2 MICROPROCESSOR
The microprocessor is contained within a 40-pin LSI chip and is shown in Figure 8-2. There are eight 16-bit general-purpose registers (R0–R7). R6 operates as the stack pointer (SP); R7 operates as the microprocessor program counter (PC). A special purpose status register contains the current processor status word (PSW). The operating characteristics of the microprocessor are affected by the mode register which is discussed in detail in Paragraph 8.3.

8.2.1 Microprocessor Initialization
The microprocessor initializes the SBC-11/21 PLUS module during the power-up sequence or when the RESET instruction is executed.

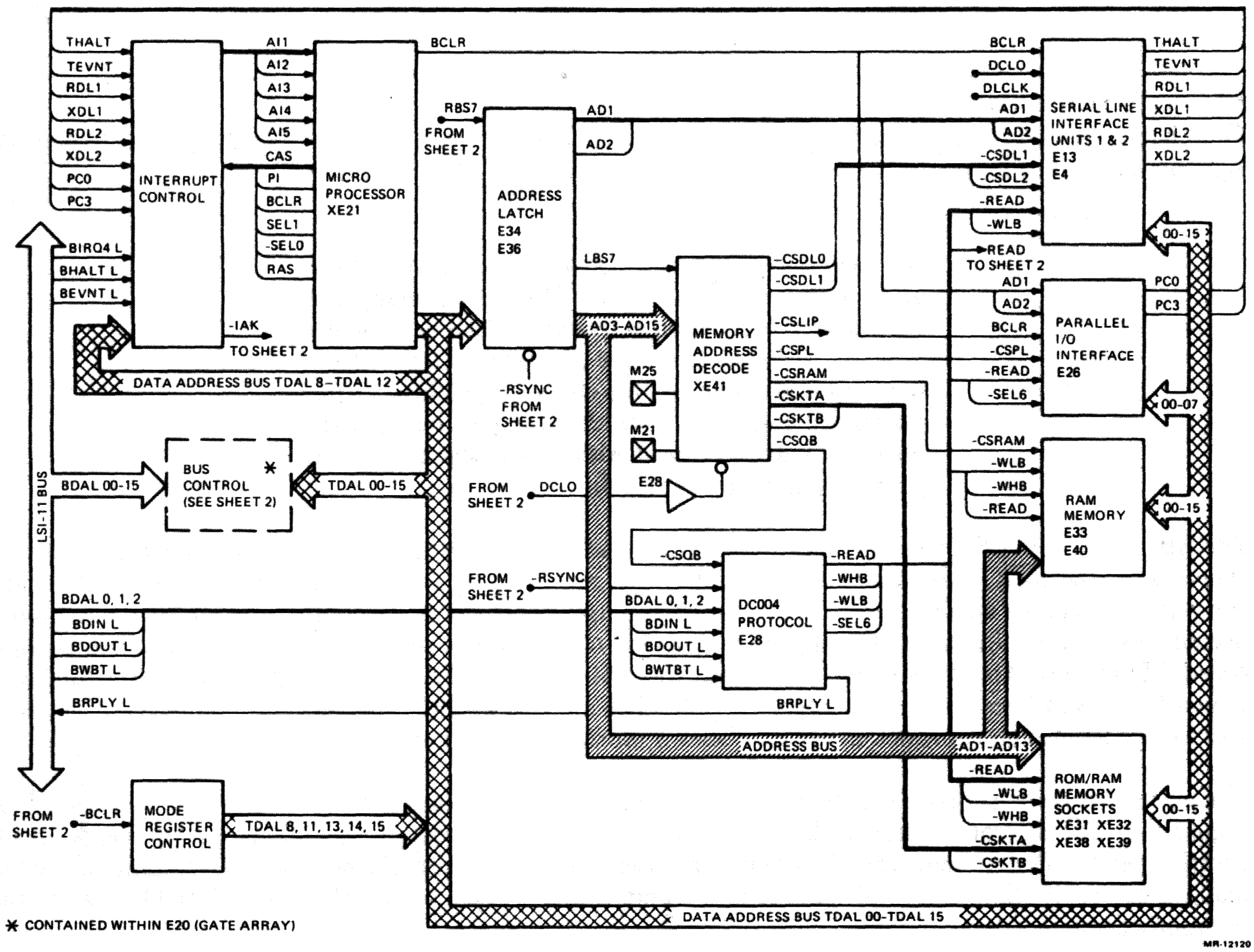
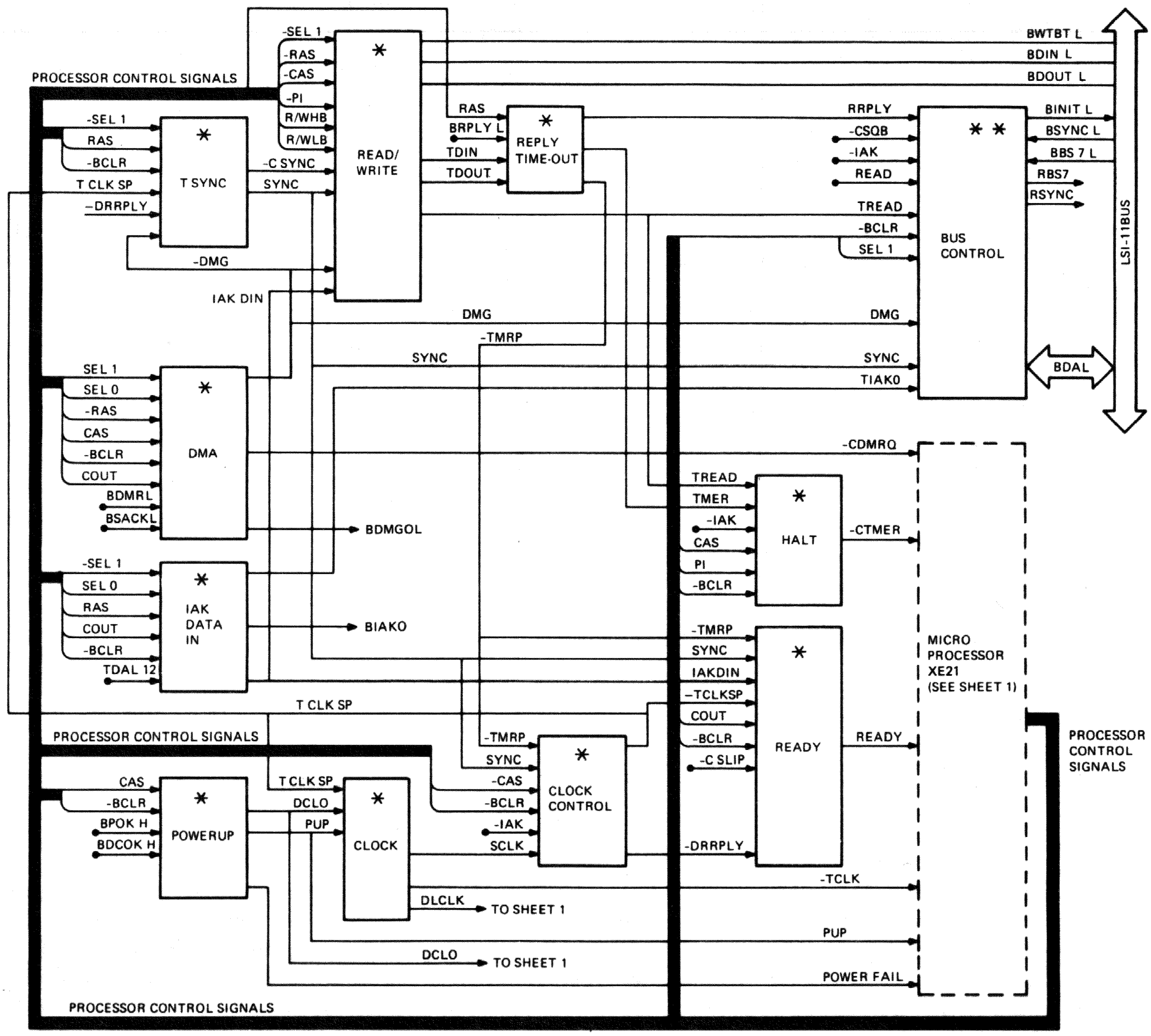


Figure 8-1 SBC-11/21 PLUS Functional Block Diagram (Sheet 1 of 2)



* CONTAINED WITHIN E20 (GATE ARRAY)

* * PARTLY CONTAINED WITHIN E20 (GATE ARRAY)

MR-12141

Figure 8-1 SBC-11/21 PLUS Functional Block Diagram (Sheet 2 of 2)

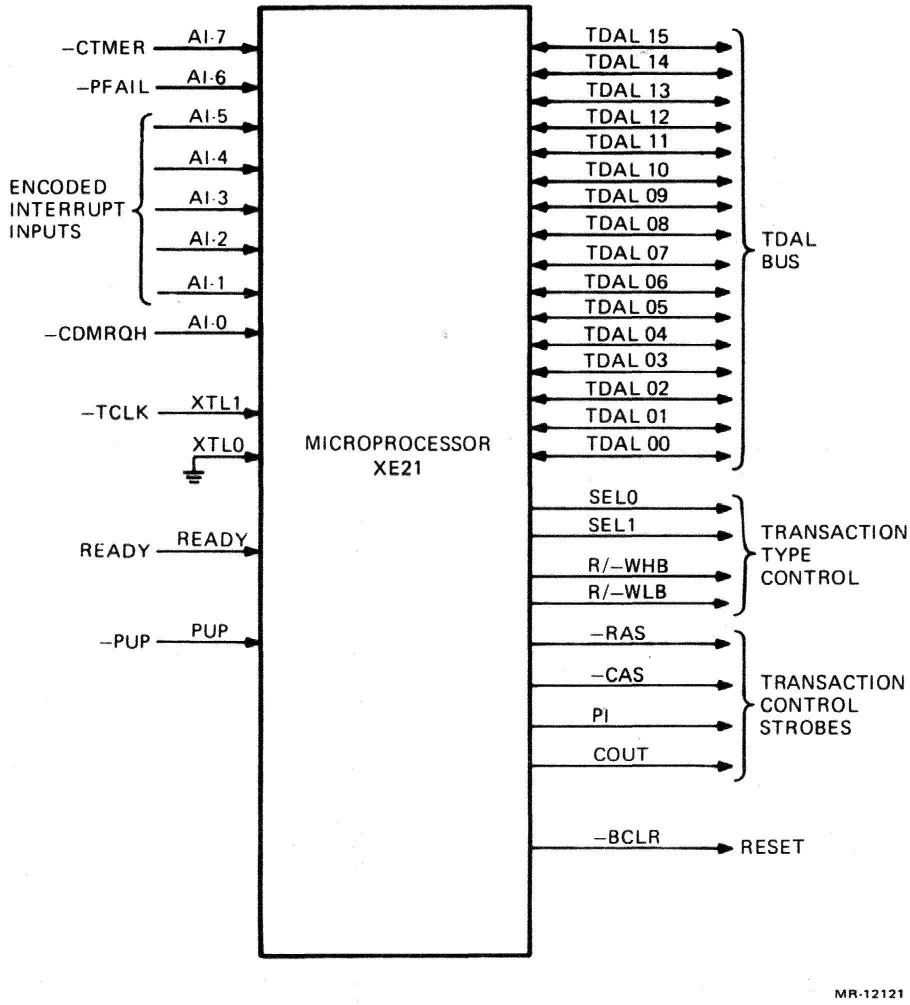


Figure 8-2 SBC-11/21 PLUS Microprocessor

8.2.1.1 RESET Instruction – The RESET instruction asserts the –BCLR output. This clears or resets the control logic of the module to an initial state. The microprocessor loads the mode register from the TDAL bus with the mode register control data. The LSI-11 bus transceivers are disabled when –BCLR is asserted. The RCVIE bit of the RCSRs and the XMITIE, MAINT, and XMITBRK bits of the XCSRs are reset in the serial line units (SLUs). The port C buffer output lines of the parallel I/O are set high. If port A and port B buffers are output to the connectors, they are also set high. The LED is turned off during reset. The –BCLR output is then negated and an assert priority in (ASPI) transaction is performed to service any interrupts or DMA requests. The RESET instruction does not change the PSW or any internal registers.

8.2.1.2 Power-up Input (PUP) – The power-up (PUP) input goes from low to high when the 5.0 V power is first applied. This initiates the power-up sequence. The –BCLR output is asserted. The module is cleared and reset as it is for the RESET instruction, however, the serial line units' (SLUs) registers are completely reset. After a delay, BDCOK and BPOK are asserted, PUP is negated, and the –BCLR output goes high. The microprocessor then performs ten bus NOP transactions. The processor loads the starting address into the program counter (R7), location 376 into the stack pointer (R6), and the processor status word is set to 340. An assert priority in (ASPI) transaction is performed to service any interrupts or DMA requests before the first instruction is fetched.

The PUP input normally stays low for all operations. If PUP is asserted high, the present transaction is terminated and the internal registers go to an undetermined state. The TDAL bus, the interrupt inputs, and the microprocessor control signals will all go to an initial reset state.

8.2.2 Clock Input (-TCLK)

The –TCLK input is a 4.9152 MHz clock that comes from the 19.6608 MHz crystal oscillator. This clock input is used for the internal time base of the microprocessor and the source of the clock output (COUT). COUT is pulsed once for every microcycle. A microcycle can represent either three or four –TCLK input pulses depending on the type of transaction. The microprocessor will halt or stop when the –TCLK input is disabled.

8.2.3 Ready Input (READY)

READY input is normally high and will not interfere with microprocessor transactions. However, when the input is held low, a single microcycle slip occurs during every transaction. When READY is clocked with COUT, while RAS is asserted, the microprocessor slips a microcycle every time the input is pulsed. This allows the microprocessor to be placed in an idle or wait state until a peripheral device has either received or asserted data on the bus.

8.2.4 Microprocessor Control Signals

The microprocessor controls the functions of the SBC-11/21 PLUS through the use of nine microprocessor control signals. A description of these signals and their functions follows. The RAS, CAS, PI, COUT, and BCLR are transaction control strobes used for logic transitions. The R/–WLB, R/–WHB, SEL0, and SEL1 are steady state logic signals used as transaction type control signals.

8.2.4.1 Row Address Strobe (RAS) – The leading edge of the RAS signal is used to acknowledge that the address is stable on the TDAL bus during read/write and fetch transactions. During interrupt transactions, the leading edge of the RAS signal strobes the interrupt acknowledge data onto the TDAL 12–8 bus lines.

8.2.4.2 Column Address Strobe (CAS) – The trailing edge of the CAS signal is used to acknowledge that data on the TDAL bus lines during read and fetch transactions was read by the microprocessor. For write transactions, the signal is used to acknowledge that microprocessor data will be removed after a specified time.

The leading edge of the signal is used to request that read data be placed on the TDAL bus and to strobe interrupt requests into latches that are read during the assertion of PI.

8.2.4.3 Priority In (PI) – The leading edge of the PI signal is used to acknowledge that data on the TDAL bus lines during write transactions is stable. The leading edge is also used to enable the microprocessor to read the interrupt inputs AI-0 to AI-7 and to initiate IAK, restart, power fail, or DMA transactions.

8.2.4.4 Read/Write (R/–WHB and R/–WLB) – The R/–WHB and R/–WLB signals control the read/write and fetch transactions by enabling the TDIN, TDOUT, and TWTBT control signals. For read and fetch transactions, both signals are asserted high and enable the TDIN control circuits. During write transactions, the TDOUT and TWTBT control circuits are enabled when either or both signals are asserted low. If only one signal is asserted low, the TWTBT control circuits are enabled by the leading edge of CAS, and a write byte transaction occurs for either high byte or low byte.

8.2.4.5 Select Output Flags (SEL0 and SEL1) – The SEL0 and SEL1 signals indicate the transaction being performed. When both signals are low, a read, write, ASPI, or NOP transaction is selected. When both signals are high, a DMA transaction is selected. When SEL1 is low and SEL0 is high, the fetch transaction is selected. When SEL1 is high and SEL0 is low, an IAK transaction is being performed.

8.2.4.6 Bus Clear (BCLR) – The BCLR signal is used to reset the control logic and generate BINIT. The signal is asserted during the power-up sequence and the execution of a RESET instruction only.

8.2.4.7 Clock Out (COUT) – The COUT signal is asserted once for every microcycle and is used to time the microprocessor transactions.

8.2.5 Microprocessor Transactions

The microprocessor performs six types of transactions to support the instruction set, direct memory access, and the interrupt structure.

1. Fetch/read
2. Write
3. DMA
4. IAK
5. ASPI
6. Bus NOP.

A normal fetch/read or IAK transaction requires either one or two microcycles; extended transactions can take as many microcycles as required before a time-out occurs. The COUT signal is asserted once for every microcycle. The transactions are used to transfer information and data via the TDAL bus which interconnects all local devices and connects them to the LSI-11 bus interface. A description of each transaction operation follows.

8.2.5.1 Fetch/Read – The fetch/read transaction is used either to fetch an instruction or read data for the microprocessor. The data may originate from the on-board memory, I/O device, or the LSI-11 bus. The microprocessor control signals for the transaction are illustrated in Figure 8-3. The R/–WLB and R/–WHB control signals are asserted. The SEL0 output is high, and the SEL1 output is low for the fetch transaction; both of these outputs are low for the read transaction.

The following sequence of events takes place during a fetch/read transaction.

1. The microprocessor places the address onto the TDAL bus when the transaction is initiated and is latched into the memory address circuits by the assertion of SYNC.

2. The data is received on the TDAL bus after RRPLY is received. The microprocessor accepts the data and negates TDIN.

3. Interrupt and DMA requests are latched by CAS, set up while PI is asserted, and latched into the microprocessor when PI is negated.

NOTE

A write transaction is always preceded by a read transaction except when the microprocessor pushes onto the stack. Therefore, each write has at least four microcycles, assert address, read data, assert address, and write data.

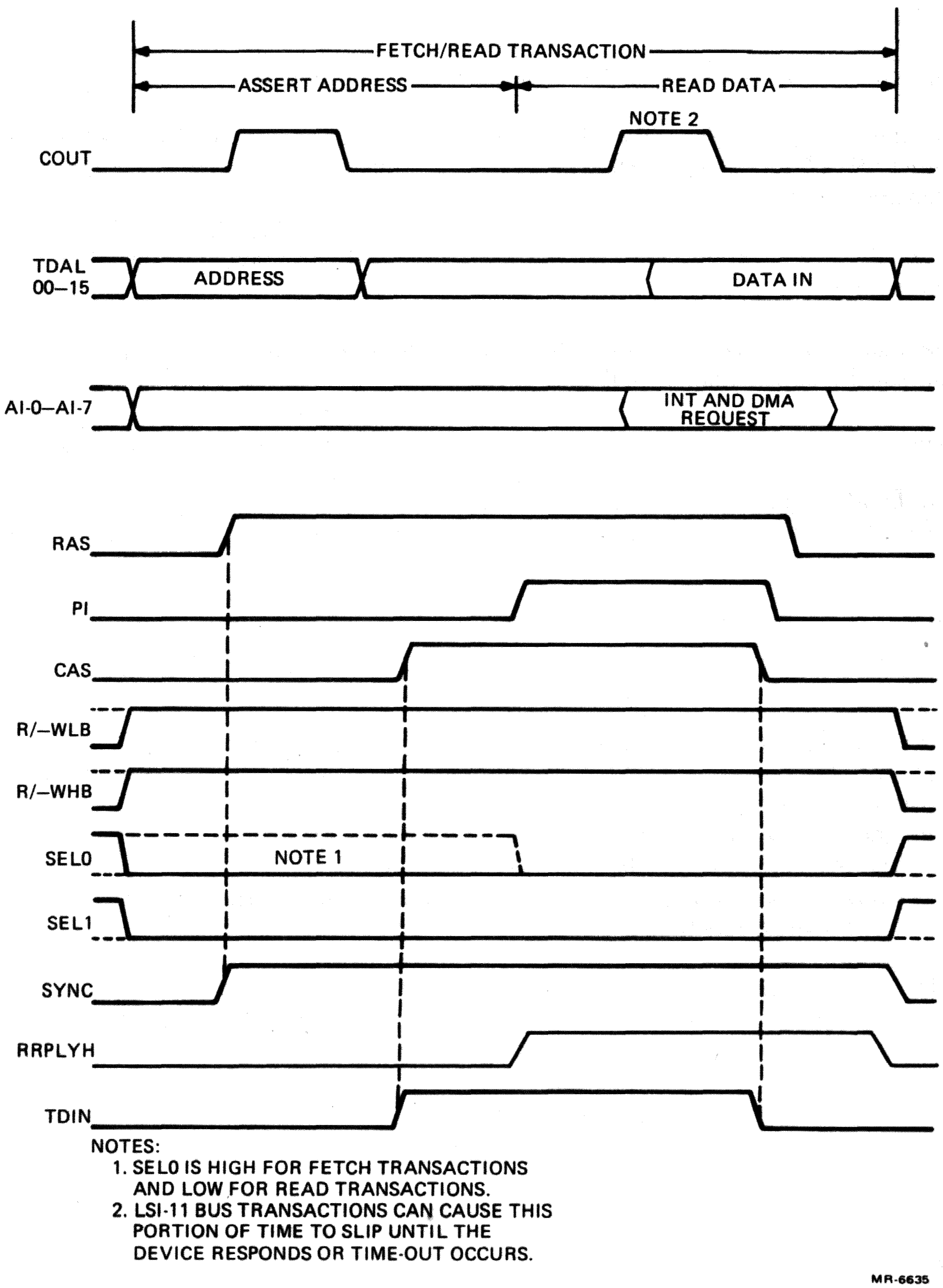


Figure 8-3 Fetch/Read Transaction

8.2.5.2 Write – The write transaction is used to write data from the microprocessor to memory, a local I/O device, or an LSI-11 bus peripheral device. The microprocessor control signals for the transaction are illustrated in Figure 8-4. The R/–WLB and R/–WHB control signals are asserted low when writing a word; for writing a byte, either the high or low byte signal is asserted. Both SEL0 and SEL1 control signals are negated.

The following sequence of events takes place during a write transaction.

1. The microprocessor places the address onto the TDAL bus, and the state of the read/write lines causes TWTBT to be asserted. The address is latched into the memory address circuits by the assertion of SYNC.

2. When CAS is asserted, TWTBT is negated for word transactions and left asserted for byte transactions.

3. The data is placed on the TDAL bus before TDOUT is asserted. The data is written into the addressed location when TDOUT is negated.

4. When the addressed device negates BRPLY, the SYNC and TWTBT signals are cleared.

5. The DMA requests are detected while PI is asserted; they are latched into the microprocessor when PI is negated. No other interrupts are read by the microprocessor during write transactions.

8.2.5.3 IAK – If an interrupt request was detected during a previous read transaction, the microprocessor initiates an IAK transaction as illustrated in Figure 8-5. The R/–WHB and R/–WLB control signals are asserted high, and CAS, PI, and SEL0 are asserted low for the transaction. The TDAL bits 12–8 represent the acknowledged input and are used to reset the interrupt request. For local interrupts, TDAL bits 7–0 are ignored because the vector address is in the microprocessor. For LSI-11 bus interrupts, the vector address is read from the bus using TDAL bits 7–2. TDAL bus bit 12 is set low for this IAK transaction and commands the control logic to initiate an LSI-11 bus IAK transaction. The TDIN signal is asserted for the transaction, and the TIAKO output acknowledges the interrupt. The requesting device then places the vector address on the low byte of the bus and asserts BRPLY. The microprocessor stops slipping microcycles, negates TDIN, and accepts the vector. It then negates TIAKO on the trailing edge of RAS and continues to the next transaction.

8.2.5.4 DMA – The DMA request is read during a previous transaction. The microprocessor will acknowledge the request by tri-stating the TDAL bus as shown in Figure 8-6. The SEL0 and SEL1 outputs are asserted to indicate that the bus mastership has been relinquished. The transaction will continue with no interruptions until the DMA transfer is completed. The microprocessor will then negate the SEL1 control output to indicate that it is resuming bus mastership. The negation of SEL0 will follow if the next transaction is not a fetch.

8.2.5.5 ASPI – The assert priority in (ASPI) transaction is used by the RESET and WAIT instructions or the power-up sequence as shown in Figure 8-7. The CAS and PI outputs are asserted to allow the microprocessor to recognize and latch any interrupts or DMA requests. The R/–WHB and R/–WLB outputs are asserted and the SEL0, SEL1, and RAS outputs are negated for the transaction.

8.2.5.6 NOP – The bus NOP transaction performs no operation and is used during the power-up sequence or if the programmer intentionally introduces a delay into the program. The AI-0 through AI-7 inputs are tri-stated to prevent interrupts. The R/–WHB and R/–WLB outputs are asserted, and the SEL0 and SEL1 outputs are taken low. The RAS, CAS, and PI control strobes are inhibited during the transaction as shown in Figure 8-8.

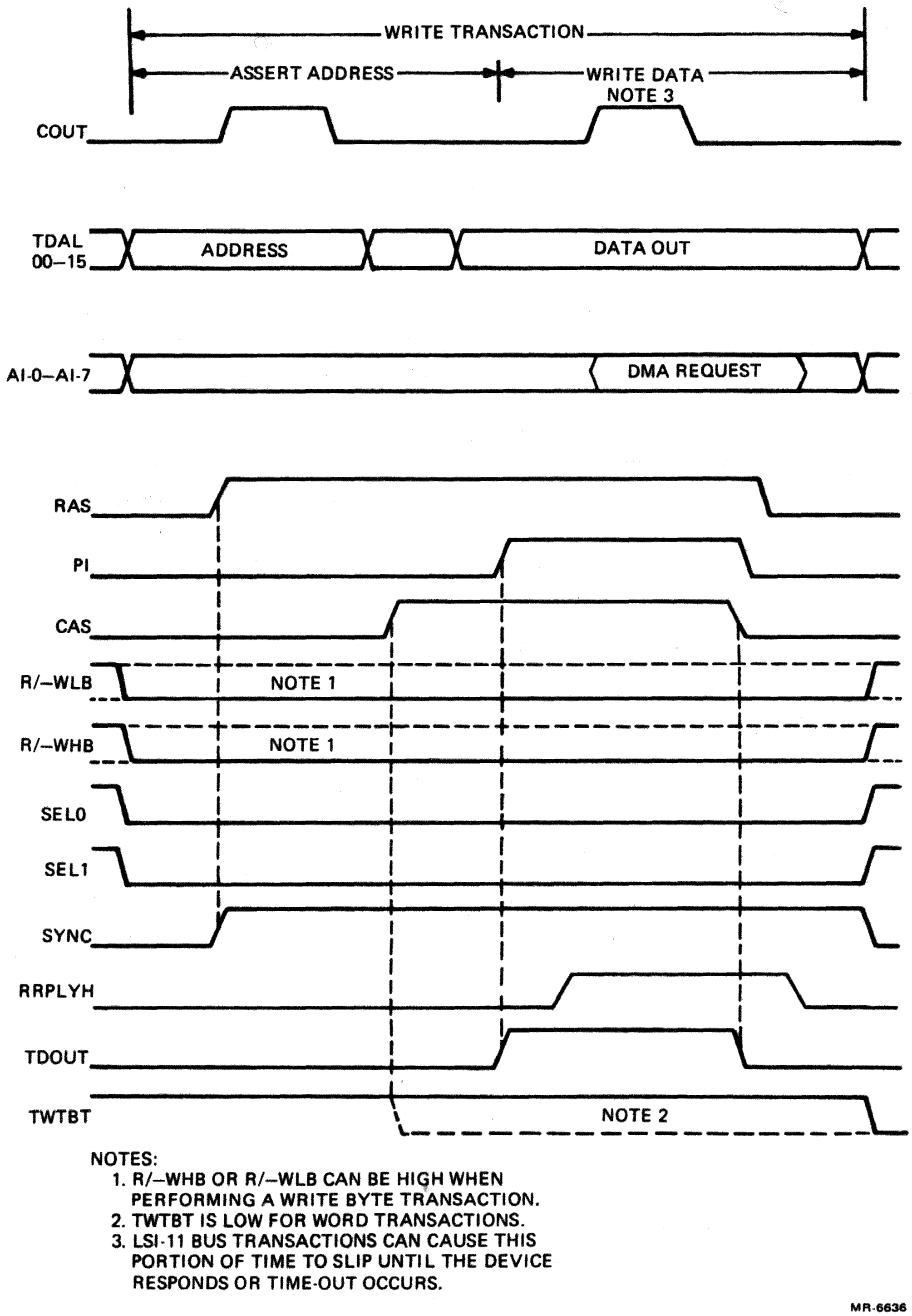


Figure 8-4 Write Transaction

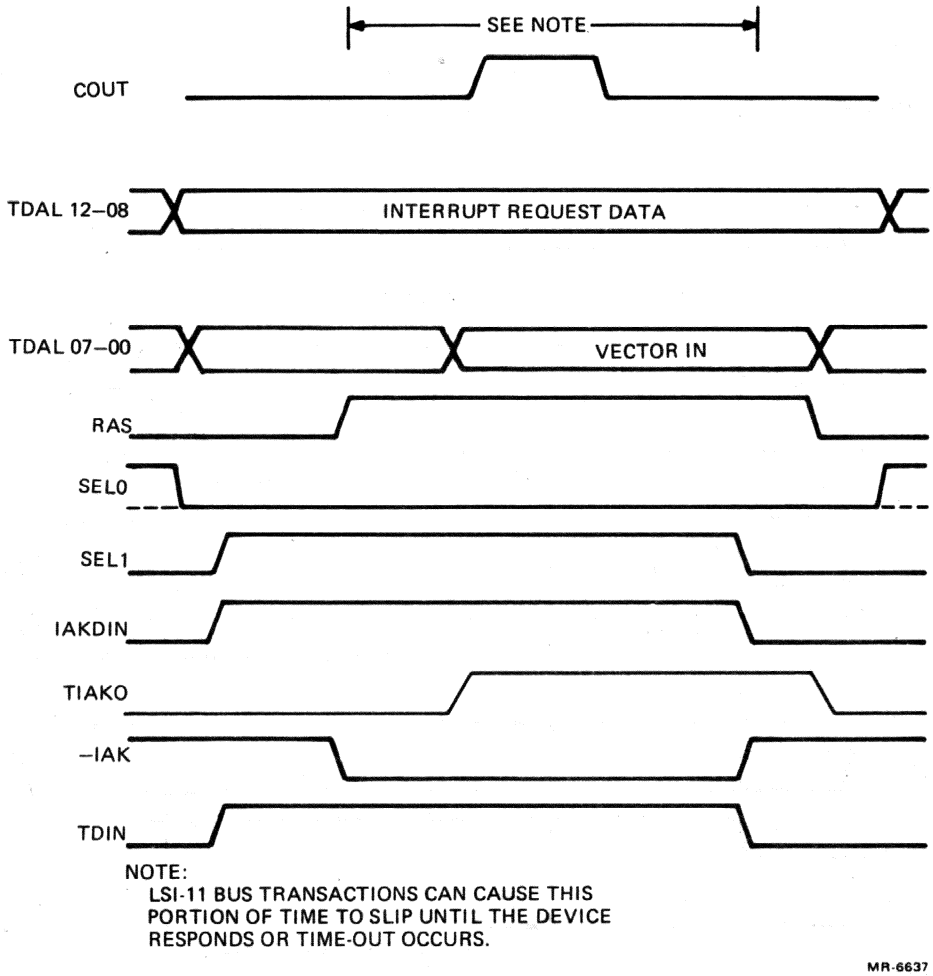


Figure 8-5 IAK Transaction

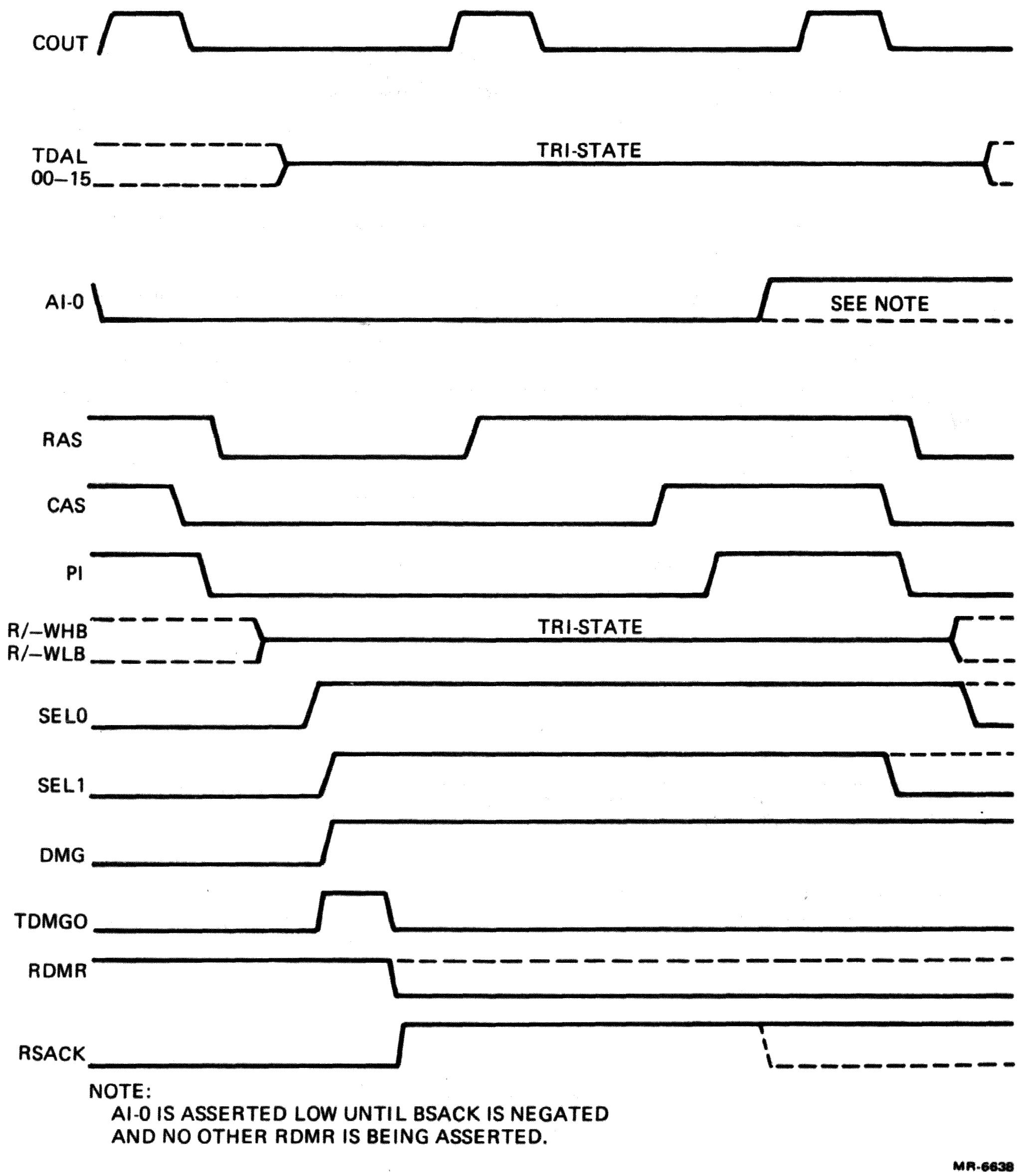


Figure 8-6 DMA Transaction

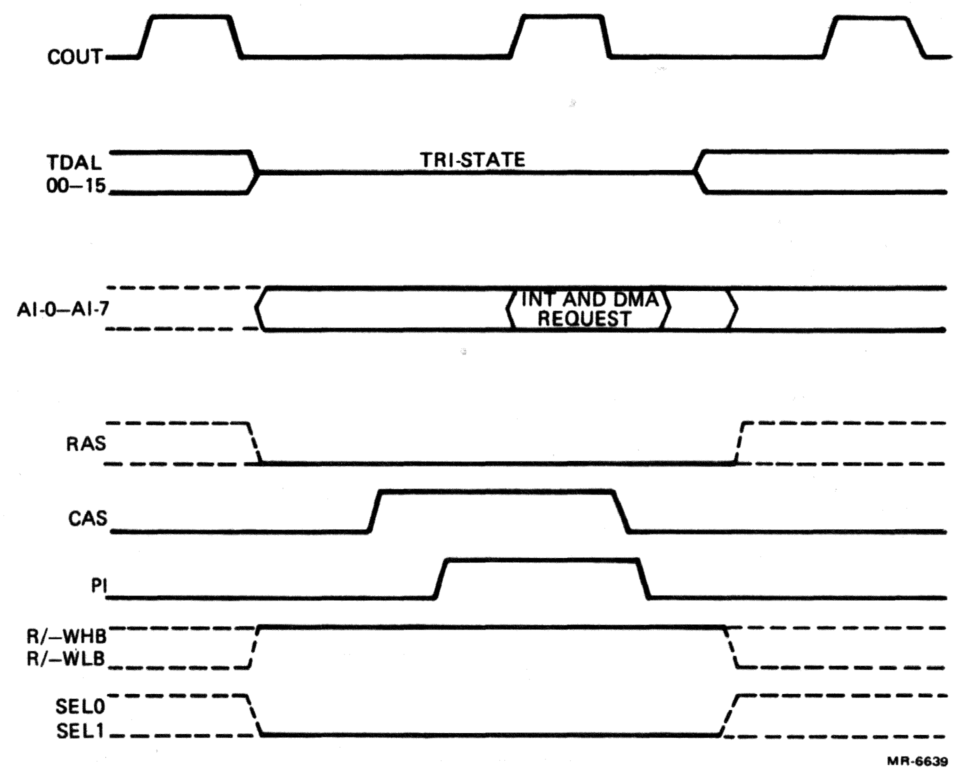


Figure 8-7 ASPI Transaction

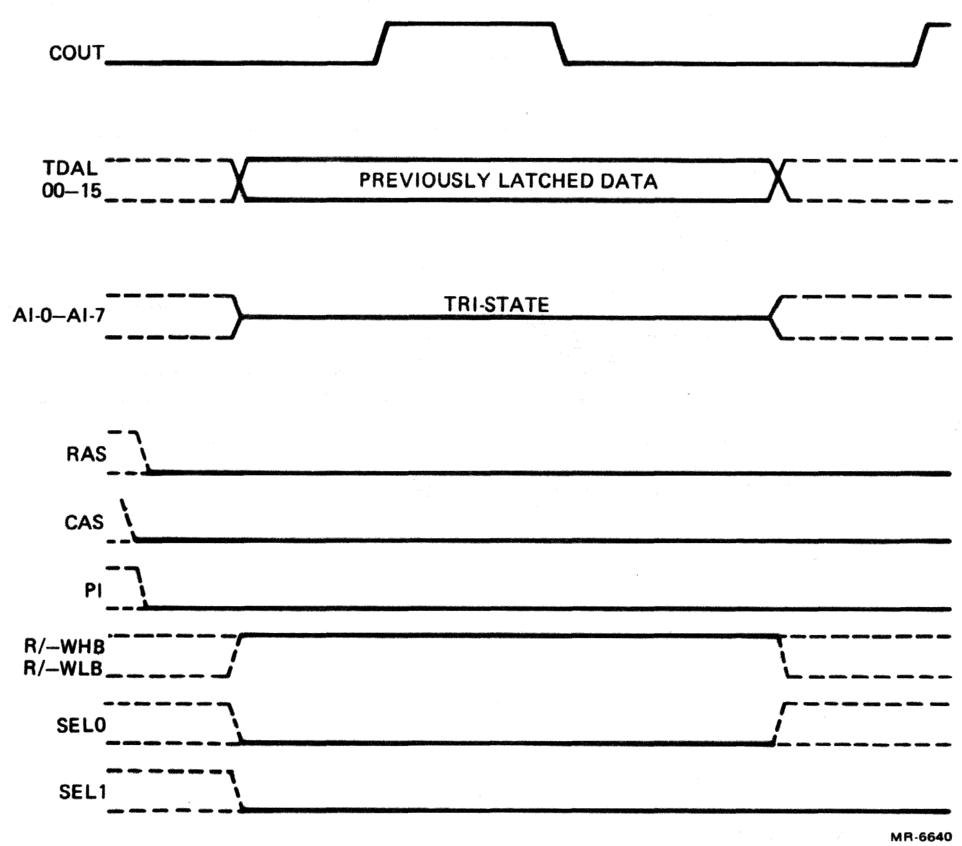


Figure 8-8 BUS NOP Transaction

8.3 MODE REGISTER CONTROL

The mode register is an internal microprocessor register used to define the operating mode of the microprocessor. The 16-bit mode register is written into from the TDAL 0–15 data lines during a power-up sequence or when a RESET instruction is executed. During this time, the –BCLR output is low and the mode register is loaded. The mode register logic (Figure 8-9) has five tri-state drivers that are enabled when the –BCLR input goes low. TDAL bits 11 and 8 are factory set to force the microprocessor to operate in the following mode.

1. The microprocessor clock mode is selected. The microprocessor pulses the COUT output once for every four XTL1 input pulses during DMA and interrupt transactions. For all other transactions, it pulses the COUT output once for every three XTL1 input pulses.

2. The standard microcycle mode is selected. It uses four XTL1 input periods for DMA and interrupt transactions and three XTL1 input periods for all other transactions.

3. The normal read/write mode is selected. The normal read/write mode sets the read/write control lines (R/–WLB and R/–WHB) prior to the assertion of –RAS and remains valid after the negation of –CAS.

4. The static memory mode is selected, and therefore, no dynamic memory chips may be installed on the module. The refresh function is disabled.

5. The memory addressing is limited to 64Kb.

6. The bus has 16 bits.

7. The user mode is selected. This mode performs transactions with no automatic test of the processor status word.

The status of TDAL bits 13–15 are selected by the user. These bits determine the start and restart addresses for the microprocessor. The start address is the location of the first fetch after power-up, and the restart address is the location of a fetch after a HALT instruction is executed or the assertion of the HALT interrupt. The wirewrap pins M22, M26, and M30 control the status of TDAL bits 13–15 during the power-up sequence. Wirewrap pin M18 is pulled up to +3 Vdc and represents a one; wirewrap pin M29 is connected to ground and represents a zero. Pins M22, M26, and M30 are jumpered to either M18 or M29, according to the list in Table 8-1, to select both the start address and restart address for the microprocessor.

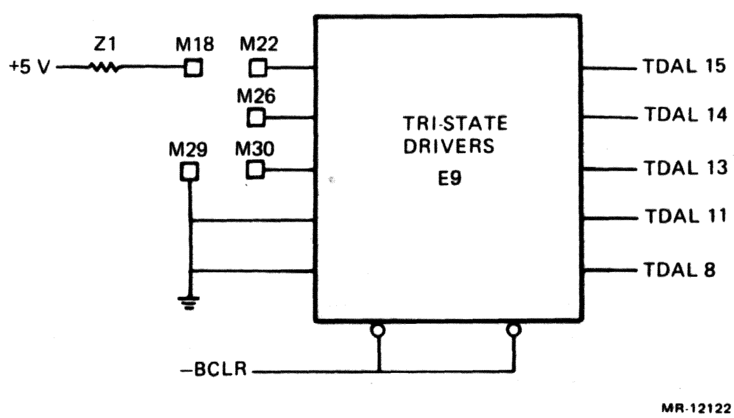


Figure 8-9 Mode Register Control

Table 8-1 Start Address Configurations

Wirewrap Pins			Start Address	Restart Address
Bit 15 M22	**Bit 14 M26**	**Bit 13 M30**		
1	1	1	172000	172004
1	1	0	173000	173004
1	0	1	000000	000004
1	0	0	010000	010004
0	1	1	020000	020004
0	1	0	040000	040004
0	0	1	100000	100004
0	0	0	140000	140004

Connection to M18 = 1
Connection to M29 = 0

8.4 INTERRUPT CONTROL

The interrupt control, as a block diagram, is illustrated in Figure 8-10. (Studying this diagram will make the explanation presented in Paragraph 8.4.1 easier to follow.)

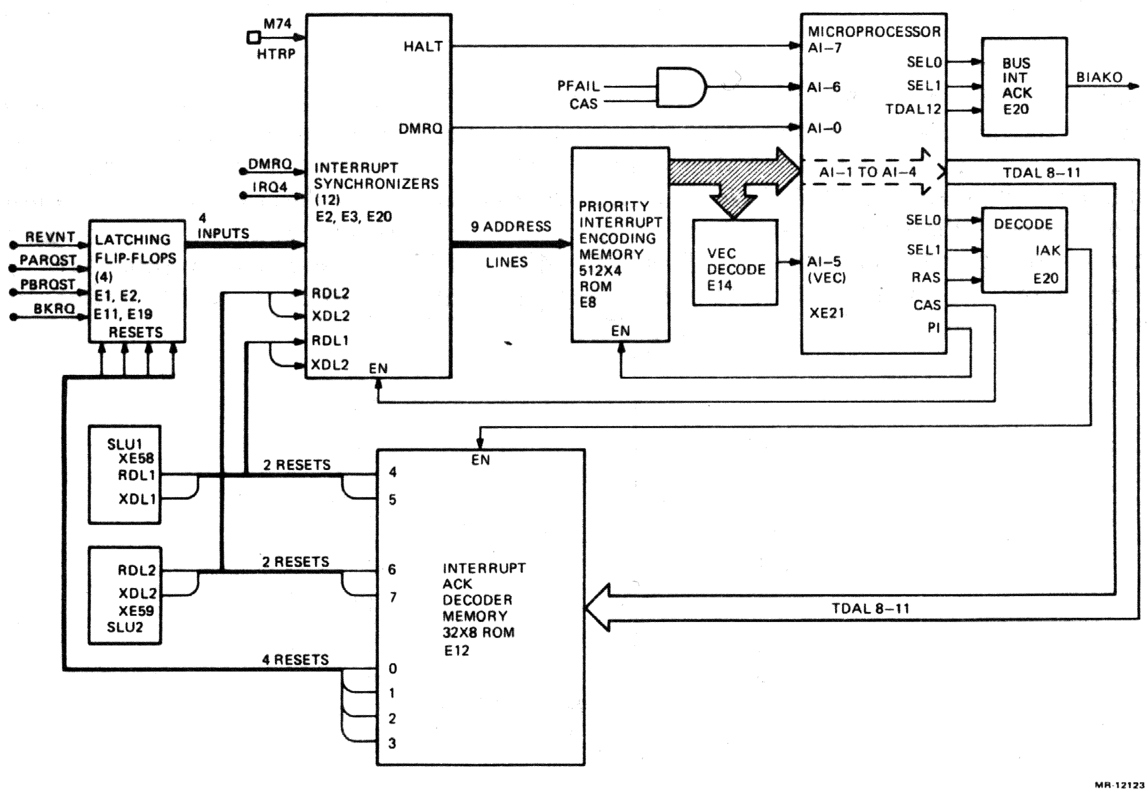


Figure 8-10 SBC-11/21 PLUS Interrupt Control

The SBC-11/21 PLUS interrupt control design includes the following elements:

1. Five D flip-flops that latch five of the interrupt lines.

a.	REVNT	Wire OR-ed TEVNT or BEVNT
b.	PARQST	Parallel I/O port A interrupt request
c.	PBRQST	Parallel I/O port B interrupt request
d.	BKRQ	Level 7, maskable interrupt, configurable
e.	HTRP	Nonmaskable interrupt, configurable

2. Eleven interrupt synchronizing latches, that latch the following signals.

   a. Outputs of the five latches described previously

   b. Two interrupt signals:

(1)	IRQ4	Level 4 LSI-11 bus interrupt
(2)	DMRQ	DMA request

   c. Four signals from the interrupt acknowledge decoder, wire OR-ed with interrupt requests from SLUs.

(1)	RDL1	SLU1 receiver interrupt request
(2)	XDL1	SLU1 transmitter interrupt request
(3)	RDL2	SLU2 receiver interrupt request
(4)	XDL2	SLU2 transmitter interrupt request

3. PFAIL is gated by CAS (through E7)

The operation of the SBC-11/21 PLUS interrupt control centers around eight microprocessor input lines, AI-0 to AI-7, driven by interrupt signals, either directly or indirectly, through the interrupt encoding PROM.

- AI-0 is the DMA request line connected directly to DMRQ.
- AI-1 to AI-4 are driven by the output of the interrupt encoder to request maskable interrupts.
- AI-5 is driven by the VEC gate which detects the presence of the LSI-11 bus interrupt on the outputs of the interrupt encoder. It calls for a vector read transaction from the bus.
- AI-6 is driven directly by the power fail input line to force a power fail trap.
- AI-7 is driven directly by the HALT interrupt line to force a restart trap.

The microprocessor reads the AI-0 to AI-7 input lines and arbitrates the interrupt priority according to Table 8-2. In addition, the state of AI-1 to AI-5 is reproduced on TDAL 12–8 lines during the acknowledge cycle. TDAL 11–8 lines are used as an address in the interrupt acknowledge decoder, which is a 32-byte PROM. Output bits 7–4 of that PROM are the previously stated SLU receive and transmitter interrupt requests (RDL1, RDL2, XDL1, XDL2) which are wire-ORed to reset the latched requests in the SLUs. TDAL 12 reflects the state of the –VEC signal and is used in the LSI-11 bus protocol.

Bits 0–3 are used as reset signals for the four interrupt latches previously described.

8.4.1 Interrupt Control Logic

The interrupt logic (Figure 8-11) receives the interrupt requests from the interface devices and applies them to the microprocessor. The microprocessor will acknowledge the highest priority interrupt if its priority is higher than the current microprocessor status word priority. There are nine interrupts available, and either one or all can be inputs to the interrupt synchronizers E2, E3 and E20. Any interrupt is active when the signal goes high. Five of these inputs are latched and stay high until reset. Four interrupts are clocked through flip-flops E1, E2 and E19 to maintain a high output. The enabled interrupts are clocked through the interrupt flip-flops by CAS asserting during the present transaction. These outputs address interrupt encode memory locations enabled by the –PI input of the present transaction. The interrupt encode memory outputs an interrupt code equivalent to the highest input priority.

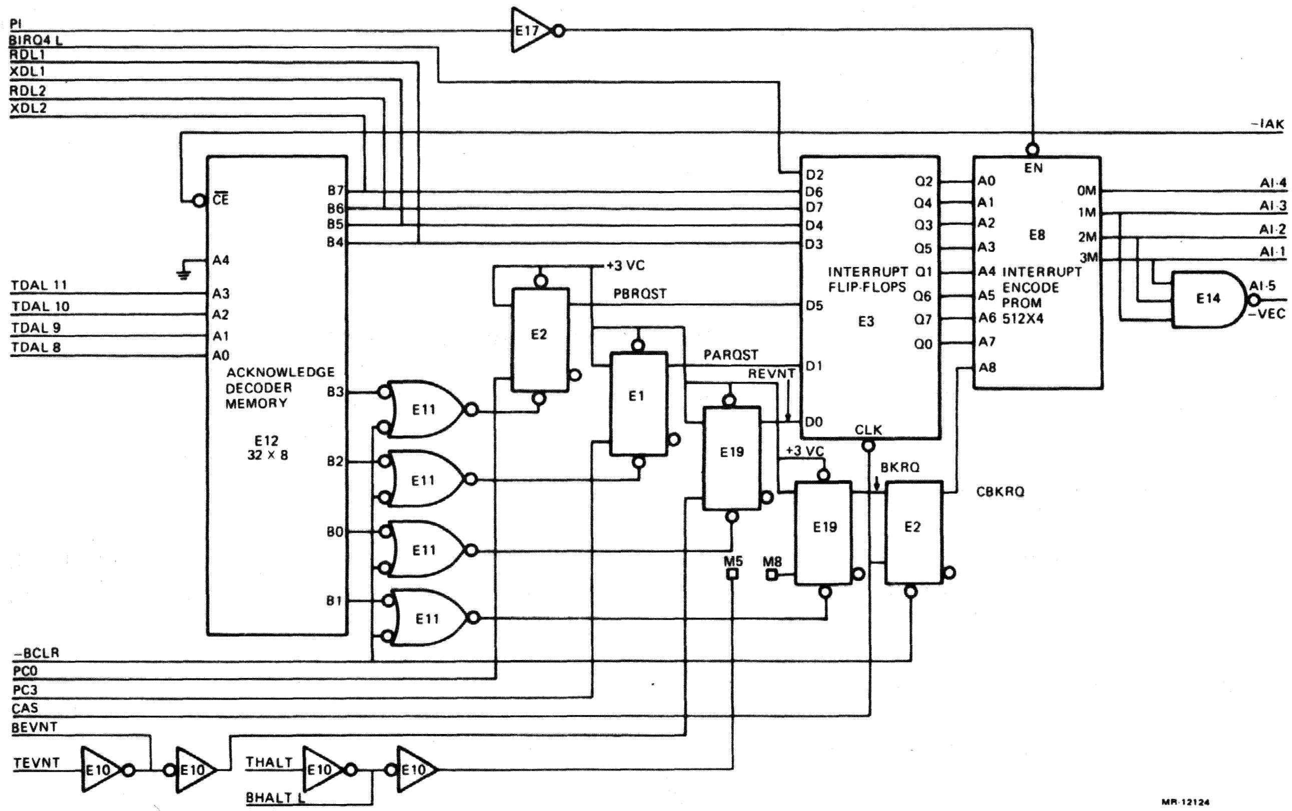


Figure 8-11 Interrupt Control Logic

The interrupt codes and their priority levels are listed in Table 8-2. When the PI output is enabled, the microprocessor looks at the interrupt inputs and will initiate an IAK transaction for an interrupt with the correct priority following the completion of a read transaction. The coded input to the microprocessor is placed on the TDAL bus using bits 8–12. Bit 8 represents the AI-1 input; bit 11 represents the AI-4 input. These four TDAL bus bits are inputs to the acknowledge decoder memory that is enabled when the microprocessor starts the IAK transaction and the –IAK input goes low. These inputs are decoded to determine which interrupt was acknowledged and will output a low to negate that interrupt. The interrupt flip-flop is reset by the clear line for that interrupt, switching the output of the selected AND gate low. The E4 and E13 transmitter and receiver interrupt lines are latched outputs and are reset by wire OR-ing and asserting low the output of the acknowledge decoder PROM. The LSI-11 bus interrupt is an exception to this process. This interrupt code enables the inputs of NAND gate E14, and the low output enables the

Table 8-2 Designated Interrupts

Interrupt Source	Input Signal	Priority Level	Coded Input AI-1	AI-2	AI-3	AI-4	AI-5	Vector Address
HALT	HLTRQ	Nonmaskable	X	X	X	X	X	Restart address
Power fail	PFAIL	Nonmaskable	X	X	X	X	X	24
LSI-11 bus signal BHALT	BKRQ	7	0	0	0	0	1	140
LSI-11 bus signal BEVNT	REVNT	6	0	1	0	0	1	100
SLU2 REC	RDL2	5	1	0	0	0	1	120
SLU2 XMIT	XDL2	5	1	0	0	1	1	124
Parallel I/O B	PBRQST	5	1	0	1	0	1	130
Parallel I/O A	PARQST	5	1	0	1	1	1	134
SLU1 REC	RDL1	4	1	1	0	0	1	60
SLU1 XMIT	XDL1	4	1	1	0	1	1	64
LSI-11 bus signal BIRQ4	IRQ4	4	1	1	1	0	0	Read from LSI-11 bus

HALT and power fail (PFAIL) interrupts are not generated by the coded inputs AI-1 to AI-5. All signals are listed in the order of descending priority.

–VEC (AI-5) input to the microprocessor. This input instructs the microprocessor to receive the vector address from the TDAL bus. TDAL 12 represents the state of the –VEC input when the microprocessor acknowledges the interrupt and is used to determine that the LSI-11 bus interrupt acknowledge handshake protocol must be initiated. The LSI-11 bus interrupt is not reset by the acknowledge decoder PROM, but it should be reset when the TDIN and TIAKO signals are received by the bus device during the interrupt acknowledge sequence.

Before continuing the discussion of the interrupt system, ready logic is discussed in Paragraph 8.4.2.

8.4.2 Ready Logic

The ready logic (Figure 8-12) provides the READY input to the microprocessor and is used to control the cycle slip function. The microprocessor will cycle slip when the READY input is being clocked while RAS is asserted; the cycle slip function will be inhibited when the READY input is set high. The output of the ready flip-flop and the COUT input go to the G62 OR gate and generate the READY input. When the –CSLIP input to G61 is high and the TSYNC input is high, the output of the G61 AND gate goes high. When –DRRPLY is not asserted and –TCLKSP and the output of G62 are high, the output of G3 is high. This enables G4 and the preset input to the G5 flip-flops to go low. The flip-flop output is low at OR gate

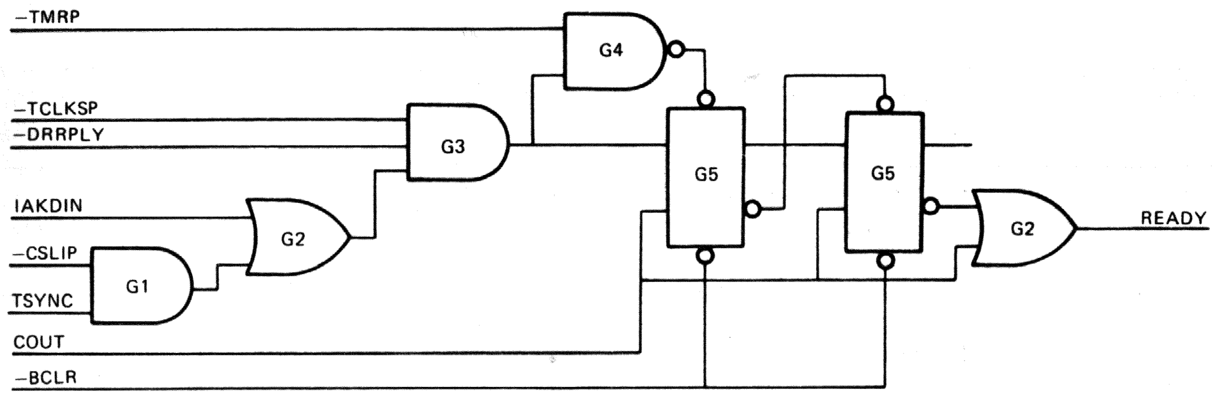


Figure 8-12 Ready

G2, and it enables the READY input with every COUT. When the IAKDIN input goes high and the −CSLIP and TSYNC inputs are negated, the output of the G3 AND gate goes high. It allows the G4 NAND gate output to go low and forces the preset terminal of the G5 flip-flop low. The output of the flip-flop to the OR gate is now low. This allows the COUT input to clock the READY output. The microprocessor will continue to cycle slip while this input is being pulsed. The −TMRP input to the NAND gate will go low when either the BRPLY or TMER input from the bus is received. This will remove the low from the preset input of the first flip-flop. Immediately after the −TMRP input goes low, the −DRRPLY input also goes low and forces a high to the input of the flip-flop. The high is clocked through by the COUT clock, and the flip-flop output to E14 will go high. This disables the READY input to the microprocessor and allows the transaction to be completed.

The second G5 flip-flop is required to ensure that data is stable at the microprocessor or at the peripheral preceding transaction completion. The ready circuit is inactive during local address references.

8.4.3 IAK Data In (IAKDIN)

The IAKDIN output is enabled by the output of the NOR gate G10 as shown in Figure 8-13. The microprocessor acknowledges an external interrupt request, asserts −SEL1, and negates SEL0. When the microprocessor has to read the interrupt vector from the bus, the TDAL 12 input is low as a result of AI-5 being low during the interrupt request read. This allows the IAKDIN output to go high and assert TDIN to the bus. The RAS input is high; this enables the TIAKO flip-flop G12. IAKDIN is clocked by the TCOUT input and causes TIAKO to go high. The inverter E15 sets the BIAKO output low. The BIAKO

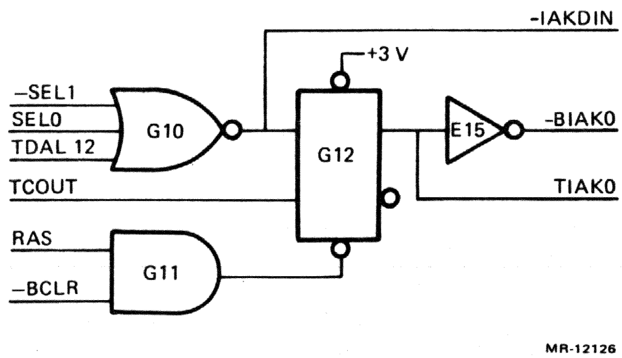


Figure 8-13 IAKDIN

output goes to the bus as an interrupt acknowledge. The TIAKO output goes to the bus transceiver logic and enables the low byte transceivers to receive the vector. The IAKDIN output goes to the ready logic and allows the microprocessor to cycle slip until the interrupting device asserts the −BRPLY input or a time-out occurs. When either response is received, the SEL0 input goes high to disable the IAKDIN output and signals that the microprocessor has read the vector. The RAS goes low to clear the TIAKO flip-flop.

The microprocessor cannot abort the reading of a vector if a time-out occurs and will read a vector of zero in all cases if −BRPLY is not asserted and the time-out counter triggers.

8.4.4 HALT Interrupt

The HALT interrupt (Figure 8-14) is defined as −CTMER and goes to the microprocessor AI-7 input. The G22 flip-flop is clocked by the input to TMER; this asserts the G23 flip-flop input. The assertion of CAS clocks the G23 flip-flop and enables the −CTMER output. The CTMER output is set high and goes to the NAND gate G20. The assertion of PI during a microprocessor read or fetch transaction latches −CTMER into the microprocessor and simultaneously switches the G20 NAND gate output low. This sets the output of the G21 AND gate low to reset the G22 flip-flop for the next HALT interrupt. The G23 flip-flop is cleared by the next CAS strobe. The microprocessor AI-7 input is pseudo edge-sensitive; it must be negated for one PI time before another trap to the restart address can be started.

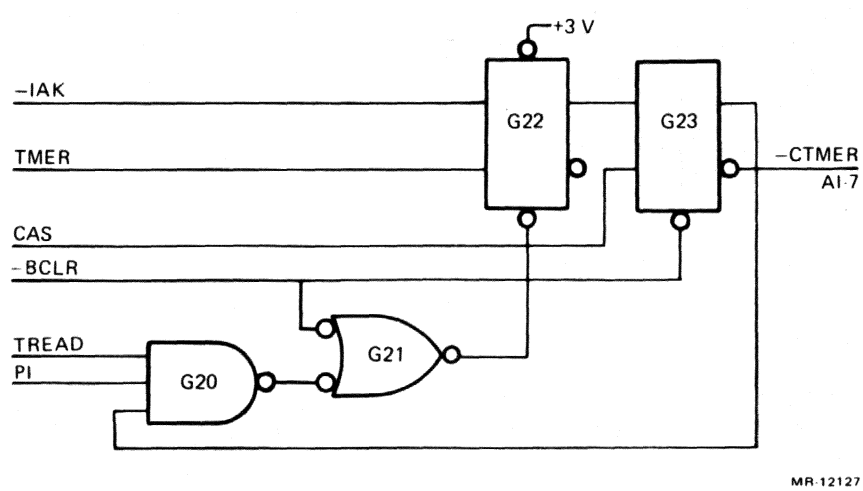


Figure 8-14 HALT Interrupt

8.4.5 Power Fail (−PFAIL)

The −PFAIL output is connected to the AI-6 input of the microprocessor and is recognized as the power fail interrupt which is nonmaskable. This is the second highest priority interrupt and it does not initiate an IAK transaction. When acknowledged, the microprocessor traps through octal addresses 24 and 26 to access the PC and PSW for the user's power fail routine. This routine should include a RESET instruction, any other instructions needed to initialize the bus and the module, an MTPS instruction that will load 340 into the PSW, and a WAIT instruction to inhibit the assertion of any LSI-11 bus control signal when battery backup is being used.

As an option to the MTPS instruction, 340 may be stored at location 26. Then, when the microprocessor vectors through 24, 340 will automatically be loaded into the PSW.

NOTE

BDCOK should not be used as a microprocessor reset signal. However, should it be used as a reset the BDCOK pulse must be at least 100 μs wide. BPOK should remain inactive during this reset operation.

8.4.6 Local

The on-board local interrupts are listed in Table 8-2 and use a coded input on the AI-1 through AI-5 inputs to the microprocessor. Some of these interrupt functions are determined by the user when configuring the module. There are eight local interrupts which are all maskable. The multiple interrupts are arbitrated, and the interrupt with the highest priority is serviced by the microprocessor. All local interrupts initiate an IAK transaction, and their vector addresses are internal to the microprocessor. During IAK, the serviced interrupt is driven on TDAL lines 11–8 to address the interrupt acknowledge PROM. The outputs of the PROM reset the interrupts. TDAL bits 7–0 are ignored. The microprocessor pushes the present PSW and PC onto the stack and receives a new PC and PSW from the vector address location and the next location.

8.4.7 External

A level 4 LSI-11 bus interrupt also uses a coded input on the AI-1 through AI-4 inputs to the microprocessor. The interrupt is maskable. For the bus interrupt, the AI-5 input to the microprocessor is taken low to indicate that the vector address must be read from LSI-11 bus bits 7–2. The microprocessor does an IAK transaction and places the BDIN and BIAKO signals on the bus to the requesting peripheral device. This device responds with –BRPLY, and the vector address is read from the LSI-11 bus. The microprocessor pushes the current PC and PSW onto the stack and reads a new PSW and PC from the vector address location and the next location.

If the interrupting peripheral device fails to assert the BRPLY bus signal within 10 μs after BDIN is asserted, the module time-out signal TMER is enabled. The microprocessor completes the IAK transaction and receives a vector address of zero because there is nothing driving the bus. The new PSW and PC are then read from locations 0 and 2. Optionally, the user can connect the time-out signal TMER to the HALT interrupt, and the interrupt can then be processed. The HALT interrupt pushes the current PSW and PC, which were read from locations 0 and 2, onto the stack and then loads the PC with the restart address and the PSW with 340. If the HALT is ignored for the vector time-out, only a vector through locations 0 and 2 will occur.

8.4.8 DMA Interrupt

The DMA request is connected to the AI-0 input to the microprocessor. The DMA request is received by the microprocessor during any read, write, fetch, or ASPI transaction. The request is not acknowledged by an IAK transaction, but is acknowledged by the microprocessor asserting the SEL0 and SEL1 outputs to initiate a DMA transaction. (See Paragraph 8.15 for a discussion of DMA transactions.)

8.5 DC004 PROTOCOL

The DC004 protocol logic chip (see Figure 8-1, sheet 1) interfaces the LSI-11 read/write signals with the module read/write signals. The –CSQB input goes high and is strobed by RSYNC to enable the logic. The BDIN L input goes low to request read data and switches the –READ output low. The BDOUT L input goes low to strobe write data and switches the –WHB and –WLB outputs low if the BWTBT L input is high. When the BWTBT L input is low, the BDAL0 L input will select either the –WHB or the –WLB. A low on the BDAL0 L input switches the –WLB output low. The BRPLY L output is controlled by the –CSQB input. When –CSQB input is high, this indicates that the LSI-11 bus was not selected. The

BRPLY L output is enabled and is switched low, after an RC delay, when BSYNC L and either the BDIN L or BDOUT L outputs are switched low. If the −CSQB input is low, the LSI-11 bus is selected and the BRPLY L output is disabled. The BDAL0, 1, and 2 inputs control the −SEL6 output. The output goes low when the BDAL1 L and BDAL2 L inputs are low and the BDAL0 L is high.

8.6 ADDRESS LATCH

The address latching logic (see Figure 8-1, sheet 1) has sixteen transparent latches designated E34 and E36. The latches are always enabled by grounding the output control input. The TDAL bus bits 1–15 and the I/O page select signal RBS7 are monitored. The status of inputs is latched to the address bus as bits AD1 through AD15 by the RSYNC input going high. The address bus and the latched LBS7 signal go to the memory address decode logic. The address bus is common to the module memories and the I/O circuits and remains stable while RSYNC is asserted.

8.7 MEMORY ADDRESS DECODE

The memory decode logic (see Figure 8-1, sheet 1) has a field programmable logic array (FPLA) that decodes the applied address bits and the latched LBS7 signal. The FPLA selects a predetermined output according to the selected memory map. The module address range includes the on-board memory, the I/O interface registers, and LSI-11 bus addresses. Four different memory maps, described in Figure 8-15, are available to the user. The M25 and M21 wirewrap pins, described in Chapter 2, allow the user to select one of these maps. The FPLA is enabled if the DCLO input is low. An address location in the RAM memory enables the −CSRAM output, and an address location of either socket set A or B of PROM enables either the −CSKTA or −CSKTB outputs. A register address for either SLU1 or SLU2 will enable the −CSDL0 or −CSDL1 outputs. The −CSPL output is enabled when a register of the parallel I/O logic is addressed. The −CSLIP output is low for all the above address conditions. The −CSLIP output goes high only when the address is accessed on the LSI-11 bus and the −CSQB output is enabled low. The −CSLIP output allows the processor to cycle slip during the LSI-11 bus read/write and IAK transactions.

8.8 RAM MEMORY

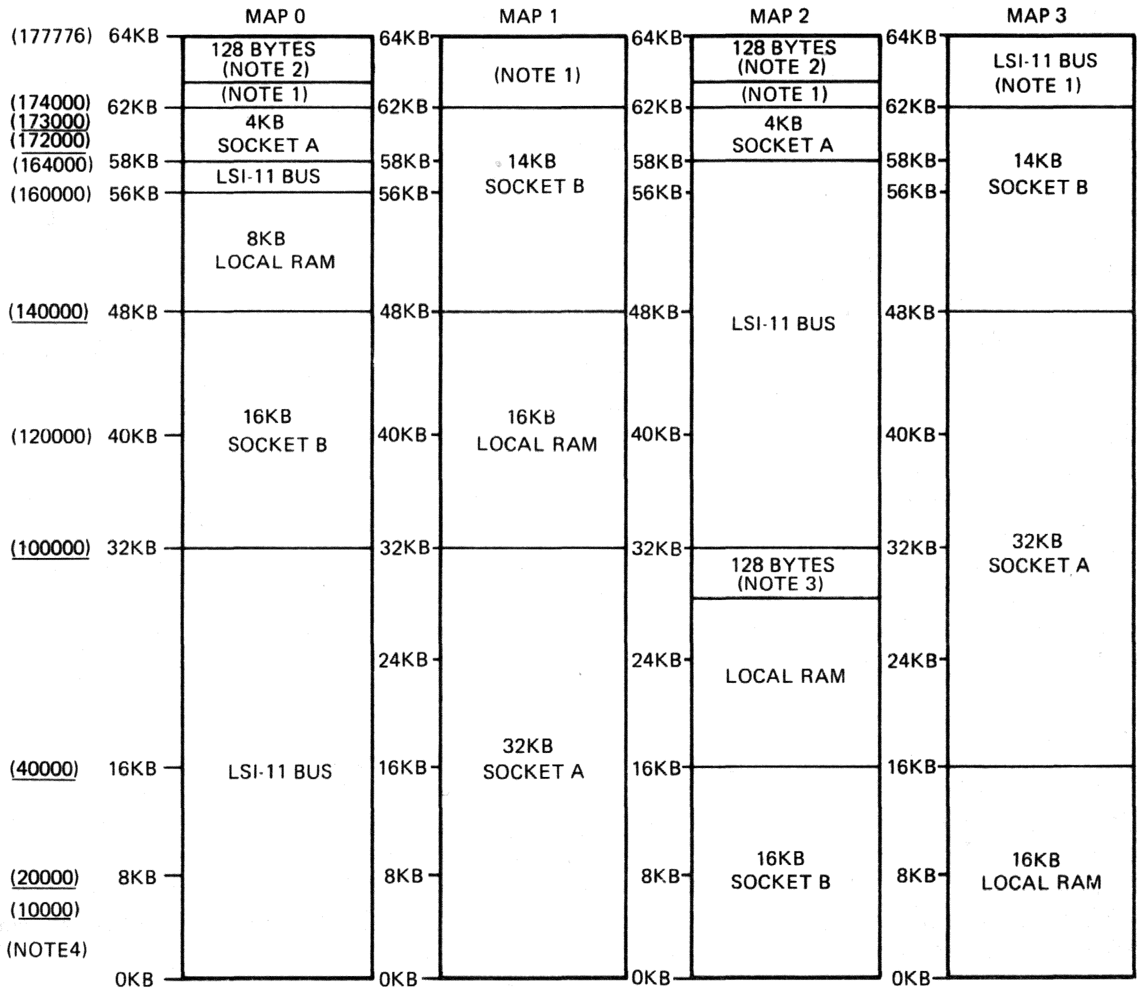
The static RAM memory, shown in Figure 8-16, is a 8K × 16-bit memory that has a 8K × 8-bit high byte chip and a 8K × 8-bit low byte chip. The memory is selected by the −CS RAM input going low to the CS pin. The memory is addressed by address bits AD1–AD13, and 16-bit data is read from or written to via TDAL bits 0–15. The memory is read by the −READ input going low to produce a low input to the OE pin of the memories. The −WLB selects the low byte, and the −WHB selects the high byte. The −WHB and −WLB inputs to the WE pin enable the write function, and −READ input goes to the OE pin of the memories to enable data output during read.

8.9 ROM/RAM MEMORY SOCKETS

The ROM/RAM memory, shown in Figure 2-5, provides the user with four 28-pin sockets to accept either 24-pin or 28-pin industry standard +5 V chips. The sockets can hold up to 32Kb of UV PROMs, EPROMS, PROMs, or ROMs and up to 32Kb of static RAM. The socket sets are defined as A and B, and each has a high byte socket and a low byte socket. The sockets use the −CSKTA and −CSKTB outputs from the memory address decode (see Figure 8-15 for the memory maps). The −READ, −WHB, and −WLB signals from the DC004 protocol are used to provide a high byte chip enable (HBCE) and a low byte chip enable (LBCE). There are sixteen wirewrap jumper pins available for the memory configuration. See Chapter 2 for detailed information.

NOTE

When a memory chip is placed into a socket wired for a larger capacity part, for example a 2K × 8 chip in a 4K × 8 socket, the addresses above the 2K boundary will wrap around into the start of the memory. This should be noted when selecting the memory map configuration.



NOTES:

1. THIS SECTION CONTAINS THE LOCAL I/O ADDRESSES FOR THE SLUS AND PPI. ALL UNASSIGNED ADDRESSES ARE ASSUMED TO RESIDE ON THE LSI-11 BUS.
2. ADDRESSES 177777 - 177600 IN MAPS 0 AND 2 ARE RAM SCRATCHPAD LOCATIONS USED BY MACRO-ODT.
3. ADDRESSES 77777 - 77600 IN MAP 2 ARE ALLOCATED TO THE LSI-11 BUS.
4. UNDERLINED ADDRESSES ARE JUMPER SELECTABLE START ADDRESSES. (SEE TABLE 2-4)

MR-12119

Figure 8-15 Memory Maps

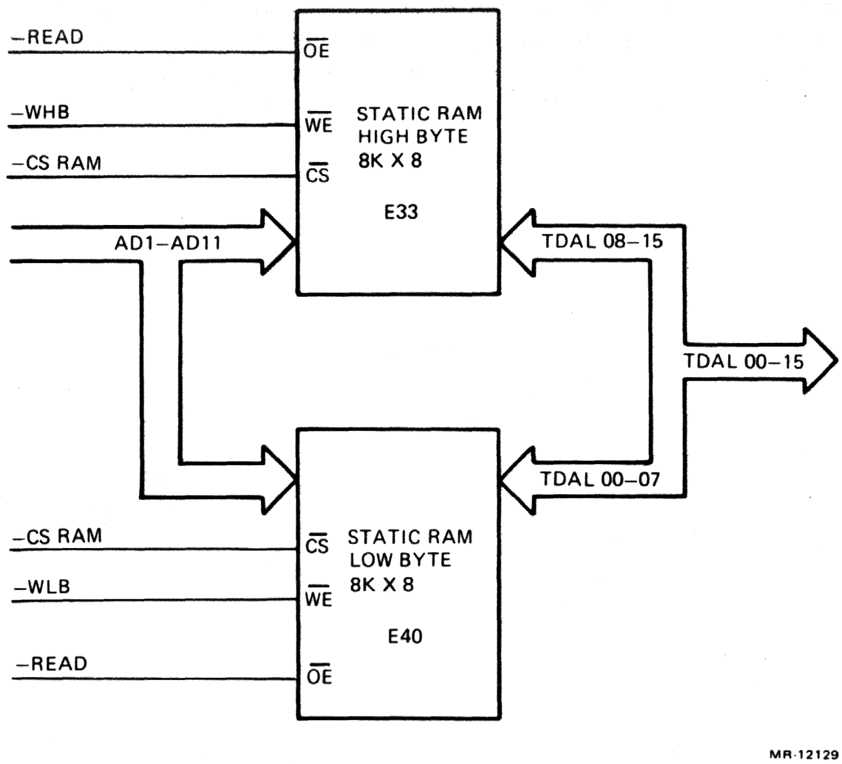


Figure 8-16 RAM Memory

8.10 SERIAL LINE INTERFACE UNITS

There are two asynchronous serial line units, SLU1 and SLU2, that provide serial I/O interface through J1 and J2 as shown in Figure 8-17. Configurations are discussed in Chapter 2.

The SLUs transmit or receive 8-bit, byte-oriented data, with no parity, one start bit, and one stop bit. SLU1 provides the XDL1 and RDL1 interrupts for transmit and receive and the BREAK output that is wired to pin M17. The user can jumper the BREAK output to the HALT interrupt (pin M20) and use SLU1 as a system console. SLU2 provides the XDL2 and RDL2 interrupts for transmit and receive and three real-time clock interrupts at 50 Hz, 60 Hz, and 800 Hz. These interrupts are wired to pins M27, M19, and M28 for use with the TEVNT interrupt (pin M23).

When the serial line units are addressed, the –CSDL0 input selects SLU1 and the –CSDL1 input selects SLU2 by enabling the chip select (CS) inputs. Address bits AD2 and AD1 are used to select individual registers within the SLUs. These registers are listed in Table 8-3 with their address and the logic states for AD2 and AD1 to access them. The –READ input will read the 16-bit register selected by –CSDL0 or –CSDL1, AD2, and AD1 by placing the contents onto the TDAL bus if the –WLB input is not asserted low. When asserted low, the –WLB input will write the low byte of the TDAL bus into the register selected by –CSDL0 or –CSDL1, AD2, and AD1. However, only the register bits defined as read/write will be written into. The DLCLK input is a crystal-controlled clock reference used by the SLU to generate baud rates and real-time clocks. The BCLR input is asserted during a RESET instruction; the RCVIE bit of the RCSR register and the XMITIE, MAINT, and XMIT BRK bits of the XCSR register are reset. When the DCLO input is asserted during power-up, it disables all SLU outputs and resets all internal logic and registers. The baud rate will be set at 300 baud after the SLU is initialized by DCLO.

The RS232 and RS423 signals for the interface connector are provided by 9636 (E22) and 9637 (E23) dual line drivers and dual line receivers. The slew rate for both channels is controlled by resistor R6. The factory configuration uses a 22 kΩ resistor to provide a 2 μs slew rate for operating at a 38.4K baud rate. See Chapter 2 for the configuration requirements at other baud rates.

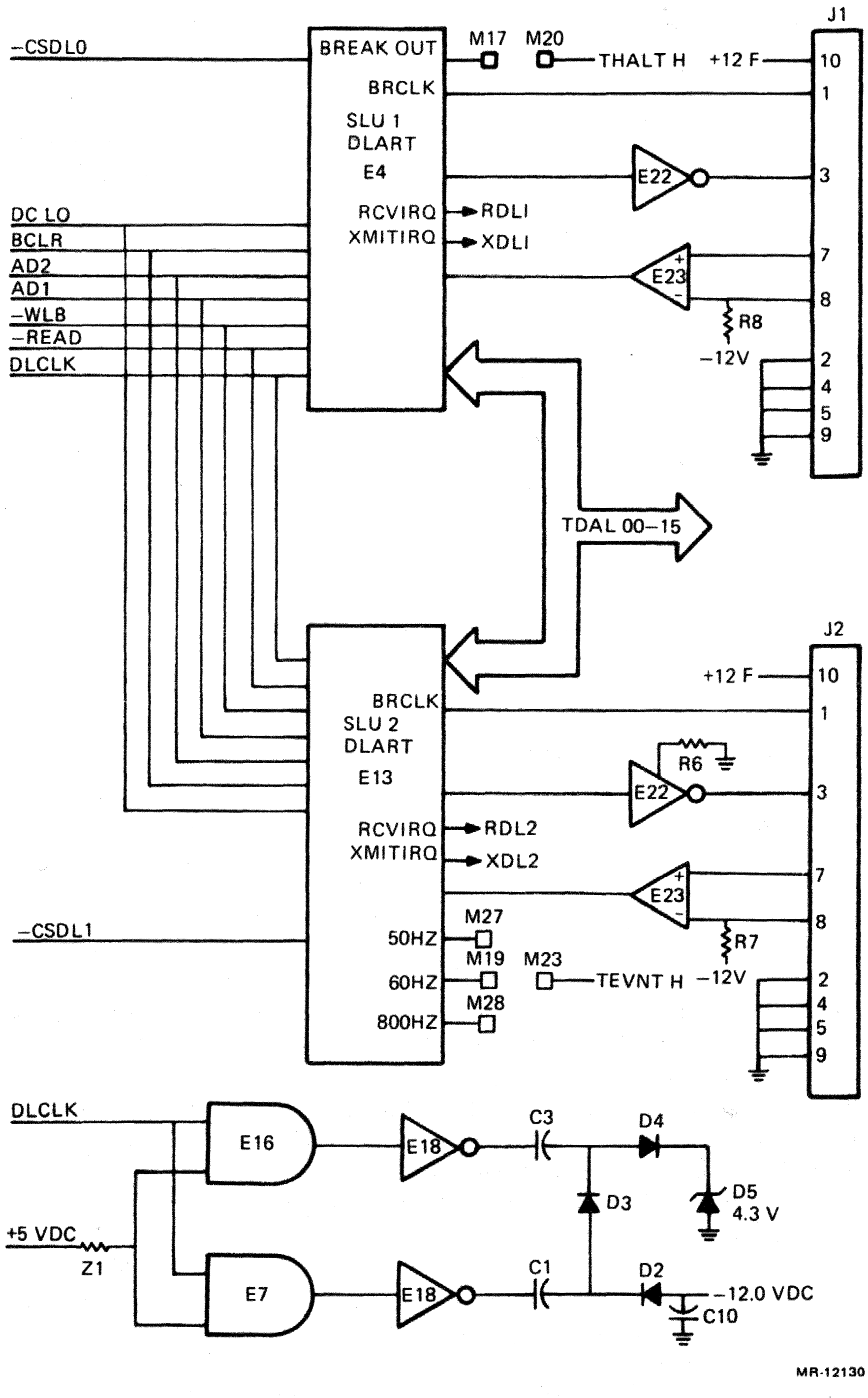


Figure 8-17 Serial Line Interface Units

Table 8-3 Serial Line Unit Registers

Register	Description	Address	AD2	AD1
SLU1				
RCSR	Receiver control/status	177560	0	0
RBR	Receiver buffer	177562	0	1
TCSR	Transmitter control/status	177564	1	0
TBR	Transmitter buffer	177566	1	1
SLU2				
RCSR	Receiver control/status	176540	0	0
RBR	Receiver buffer	176542	0	1
TCSR	Transmitter control/status	176544	1	0
TBR	Transmitter buffer	176546	1	1

8.11 PARALLEL I/O INTERFACE

The programmable parallel I/O provides a 30-pin connector for transferring parallel data into or out of the SBC-11/21 PLUS module. The parallel I/O uses an 8255A-5 programmable interface chip, two 8-bit transceiver chips, and an 8-bit buffer chip as illustrated in Figure 8-18. The 8255A-5 chip has three input/output ports defined as port A, port B, and port C. Port A and port B outputs are connected to 8-bit bidirectional transceivers that are controlled by wirewrap pins M49 and M52. When a logical one is applied to these pins, the data lines function as inputs to the module. When a logical zero is applied to these pins, the data lines function as outputs from the module. The user can configure these as inputs or outputs by using wirewrap pins M51 and M50 or as programmable inputs/outputs by programming the PC4 and PC6 lines (M54, M58) of port C as described in the configuration description in Chapter 2. The port C outputs are connected to directional buffers and are used for interrupts and the handshake control for ports A and B. PC0 and PC3 are wired as outputs, PC3 enables the parallel interrupt request for port A, and PC0 enables the parallel interrupt request for port B. PC4 and PC6 can be used as acknowledge or strobe inputs or can be configured to dynamically control the direction of ports A and B from either the 8255A-5 interface or the external peripheral device. PC1, PC5, and PC7 are wired as outputs, and PC7 is wired to an LED that can be program controlled. PC2 is wired as an input and has a current limiting resistor for protection when PC2 might be programmed as an output from the 8255A-5 interface. See Chapter 2 for detailed configuration requirements and Chapter 6 for programming information.

The 8255A-5 programmable peripheral interface (PPI) is enabled by the –CSPL input from the memory address decode chip when the 176200-176207 addresses are selected. The AD1 and AD2 address lines are decoded to select one of the four registers listed in Table 8-4. The port A, port B, and port C registers are read/write registers, and the control word register is a write only register. The addressed register is written into with the data on the TDAL 7–0 bus when the –WLB input is asserted. The content of the addressed register is placed on the TDAL 7–0 bus when the –READ input is asserted. The –SEL6 L input to NAND gate E7 inhibits the read strobe from the control word register, and therefore, any read of the control word register produces invalid data to the microprocessor. Only the low byte of the TDAL bus is used with the PPI, and any data on the high byte is always considered invalid. The –BCLR input is used to reset the PPI when it is asserted, and all twenty-four 8255A-5 I/O lines are then defined as inputs. The buffer outputs to the connector will be driven high.

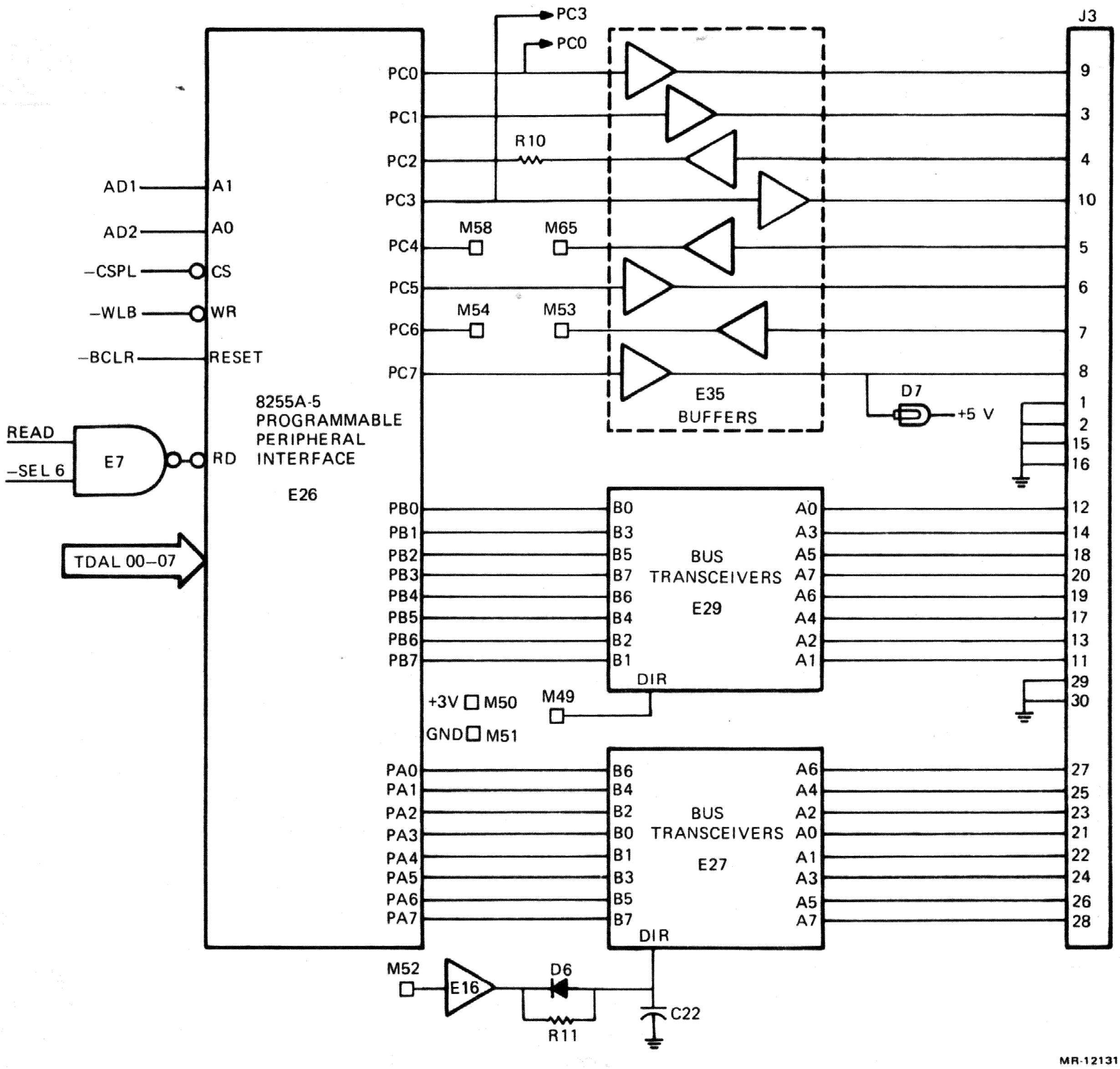


Figure 8-18 Parallel I/O Interface

Table 8-4 PPI Addressable Registers

Register	Address	Status
Port A	176200	Read/write
Port B	176202	Read/write
Port C	176204	Read/write
Control word	176206	Write only

8.12 POWER-UP

The power-up circuits (Figure 8-19) sense the application of +5 V NCR power source to the module and initiate a power-up sequence. When the +5 V NCR input is first applied, the input at the inverter G30 is low and causes the clear input of the PUP flip-flop G31 to be low, therefore keeping its output low. When the input to the NAND gate G32 is low, the −PUP output is high and the microprocessor is held reset and asserts the −BCLR output. The +5 V NCR input charges C8 through R4 until the threshold level of inverter G30 is reached. This occurs at approximately 2.6 Vdc and 70 ms after +5 V NCR was applied. This causes the reset input to the PUP flip-flop to go high and the set input to go low, setting the flip-flop. The −PUP output of the NAND gate G32 goes low. This initiates the power-up sequence of the processor.

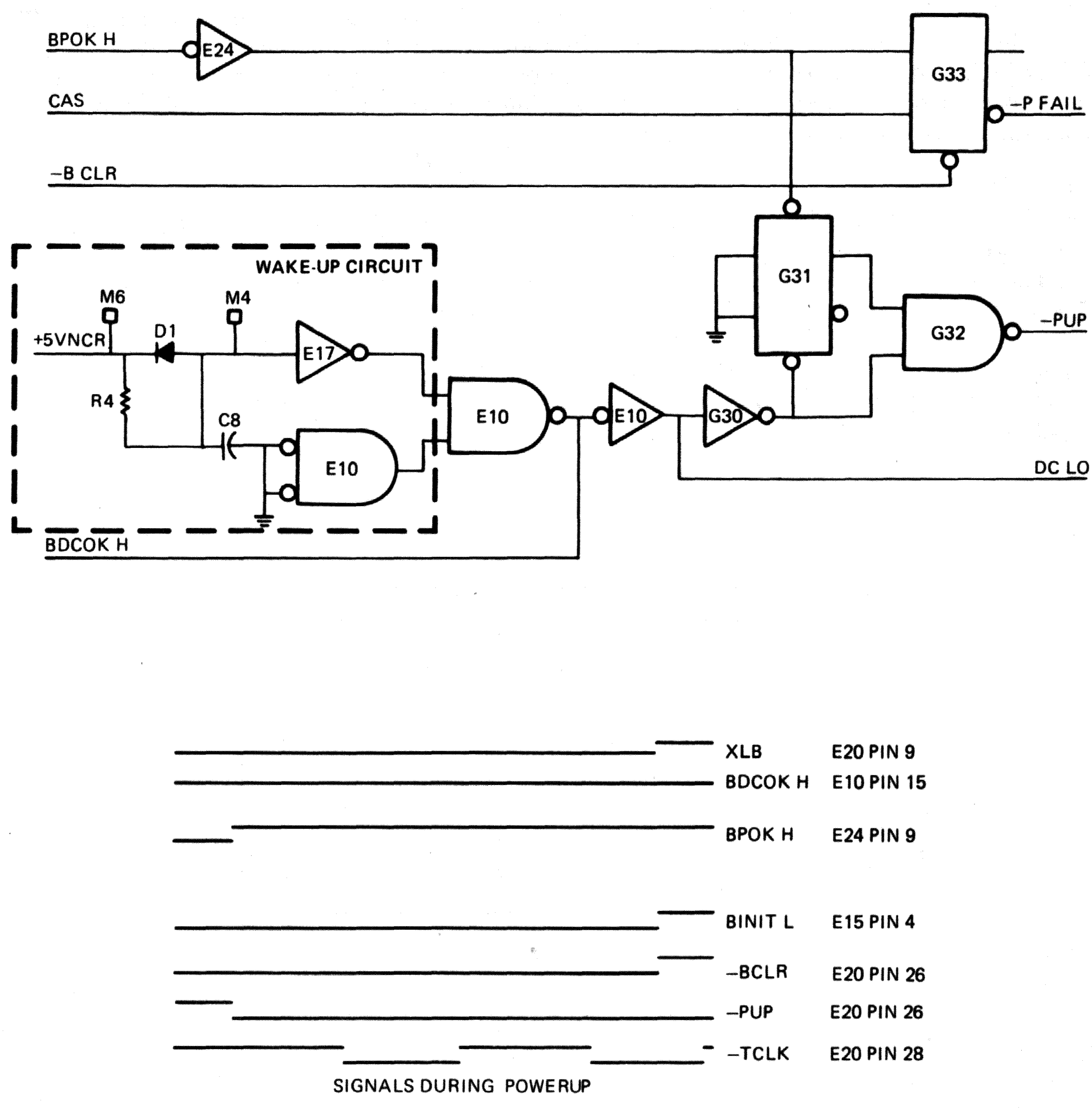


Figure 8-19 Power-up

The power-up delay circuit can be bypassed by inserting a jumper between M4 and M6. This allows the BDCOK H and BPOK H bus signals to control the PUP output. The +5 V NCR input goes directly to the inverter G30 driving input to the NAND gate E10 low. The E10 output is then controlled by BDCOK. The BDCOK H signal is low until the power supply stabilizes, causing the reset input to the PUP flip-flop to be low. The BPOK H signal is also low and causes the preset input to the flip-flop to be high. The low input to NAND gate G32 drives the −PUP output high. The microprocessor then asserts the −BCLR output resetting the PFAIL flip-flop. After a minimum of 3 ms, the BDCOK bus input goes high and allows the PUP flip-flop G31 reset to go high. After a minimum of 70 ms, the BPOK H bus input goes high causing the PUP preset input to go low. This allows the output to go high, and when both inputs to NAND gate G32 are high, the −PUP output is low. This initiates the power-up sequence of the microprocessor.

The BPOK H bus input also goes to the PFAIL flip-flop G33. During the power-up sequence, −BCLR resets the PFAIL flip-flop. The flip-flop remains reset until the BPOK input goes low indicating a power fail. The next CAS input clocks the PFAIL flip-flop and sets it. This causes the power fail interrupt, and the microprocessor traps to location 24. The flip-flop must be reset for at least one microprocessor read before another assertion will be recognized by the microprocessor.

8.13 CLOCK

The module uses a 19.6608 MHz crystal oscillator as the basic time base reference. The oscillator output goes to the clock control logic (see Figure 8-20) and to the G34 binary counters. The counters are always enabled. The 19.6608 MHz output is divided by 32, and the DLCLK output, at 614.4 kHz, goes to the serial line units and to the charge pump. The 19.6608 MHz output is also divided by 4, and the 4.91 MHz output goes to the pulse sync circuit G35. When the TCLKSP input is low, the circuits are enabled and the output goes to the next pulse sync circuit. When the TCLKSP input is high, the circuits are inhibited and there is no output. The second pulse sync circuit is controlled by the PUP input. When the PUP input is low, TCLK to the XTL1 input is enabled. When the PUP input is high, the XTL1 input is inhibited.

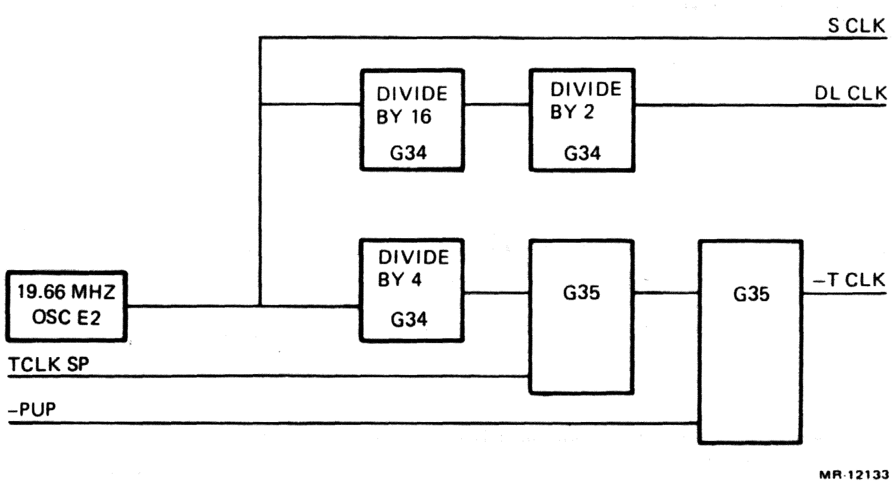


Figure 8-20 Clock

8.14 CLOCK CONTROL

The clock control logic (Figure 8-21) stops the XTL1 input to the microprocessor and forces the microprocessor to stop or wait until the XTL1 input is enabled again. The TCLKSP output is normally low to enable XTL1 and is controlled by the TMRP input being high. TMRP forces a low for both inputs to the OR gate G44, and the low output is clocked through the TCLKSP flip-flop by the 19.6 MHz input. When TMRP goes low, this removes the low inputs to the AND gate G43 and the IAK flip-flop G42. The TSYNC input is high for read/write and fetch transactions, and when the –CAS input goes high, the AND gate G43 output also goes high. The output of AND gate G43 is clocked through the TCLKSP flip-flop, and the output goes high to stop the 4.91 MHz clock output of G35. The TSYNC input is low for DMA and IAK transactions so that input to the AND gate G43 holds the output low. However, the IAK flip-flop G42 is set when the –IAK clock input goes high at the end of an external interrupt transaction and the G44 output goes high. The G44 output is clocked through the TCLKSP flip-flop, and the output goes high to stop the 4.91 MHz clock output of G35. The microprocessor XTL1 input will stay stopped until the TMRP input goes high again because either BRPLY or TMER have been negated. This forces the IAK flip-flop G42 output to go low. This negates the TCLKSP output and enables the XTL1 input to the microprocessor.

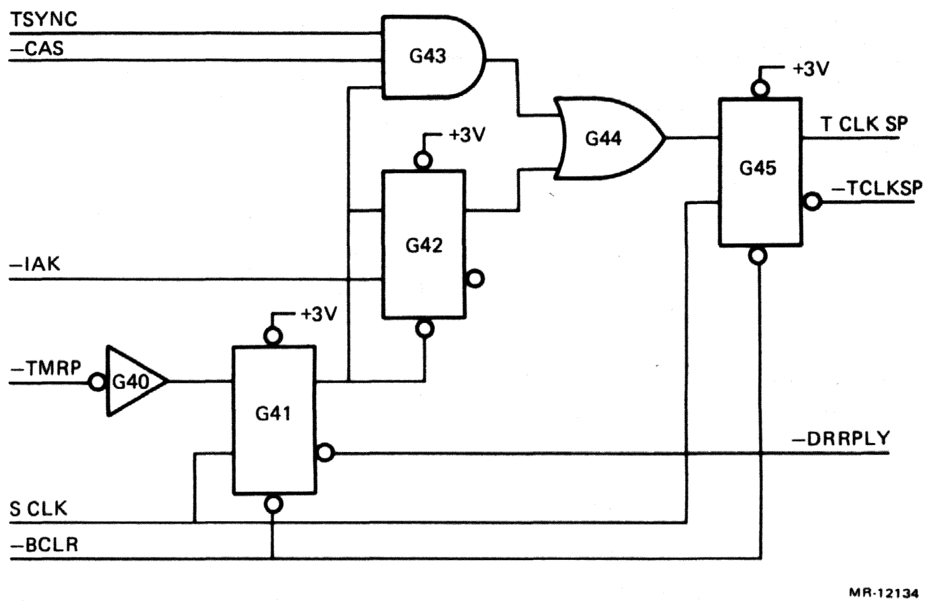


Figure 8-21 Clock Control

8.15 DMA

The DMA logic (Figure 8-22) controls the bus and microprocessor for DMA transactions. The BDMR L input goes low to start a DMA request. The output of the inverter goes high and is clocked through flip-flop G50 by COUT. The low output goes to the G56 NOR gate, and the high output goes to flip-flop G50. The high output is clocked through by COUT and enables the two NAND gates G52 and G57. The high output is also clocked through flip-flop G54 by the CAS input. The high output enables the NAND gate G55, and the –CDMRQ output (AI-0 input) is switched low. The –CDMRQ output is the DMA interrupt to the microprocessor and it starts a DMA transaction. The microprocessor acknowledges the request by setting SEL1 and SEL0 high to NAND gate G51. The preset of flip-flop G53 goes low to set the DMG output high and the –DMG output low. The DMG high input to NAND gate G55 switches the output low and goes to NOR gate G56. The BSACK L input is normally high and, when inverted by E24, is a low input to the NOR gate G56. All three inputs to the NOR gate G56 are now low causing the output to switch high. Two high inputs to the NAND gate G57 switch BDMGO hi, and then low thru E6 onto the bus to the originator of the DMA request. The requesting device then sets the bus signal BSACK L low and the BDMR L input high. BSACK L is inverted by E24 and removes the low from the NOR gate G56 and the high input to the NAND gate G57 causing the BDMGO output to go low. It also provides a high input to NAND gate G52 causing the output to switch low. This low goes to the preset input of the flip-flop G50 and clamps the output high; this holds the microprocessor in the DMA mode. The requesting device maintains the BSACK L input low for the duration of the DMA transfer and then sets it high. This removes the low from the preset input of flip-flop G50 and enables the flip-flop. Previously, the BDMR L input went high and was inverted as a low to flip-flop G50. This low was clocked through by COUT and provided a low input to the enabled flip-flop G50. The low is now clocked through causing the −CDMRQ output to go high. This removes the request from the microprocessor. The microprocessor completes the DMA interrupt transaction and negates the SEL1 and SEL0 outputs. The preset input of flip-flop G53 is no longer low, and the low data input is clocked through when RAS goes high. The DMG output goes low, and the −DMG output goes high to complete the DMA transaction.

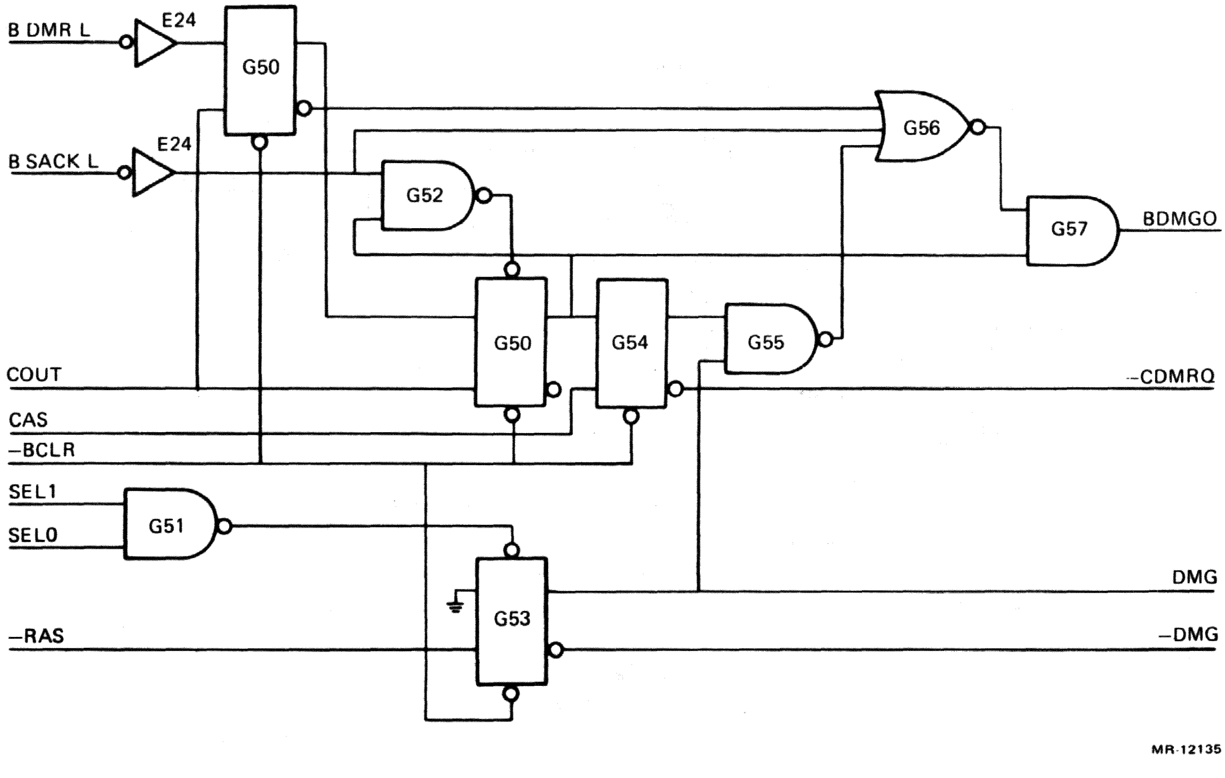


Figure 8-22 DMA

8.16 TSYNC

The TSYNC output (Figure 8-23) is normally high for the microprocessor controlled fetch/read and write transactions and low for IAK and DMA transactions. These conditions follow the –SEL1 input which is high and low for the same transactions. The exclusive OR gate G61 is wired as a noninverting buffer, and when RAS goes high, the –SEL1 input of the TSYNC flip-flop G60 is clocked through as the output. When the –CSYNC clear input goes low, it forces the output of the TSYNC flip-flop G60 to go low. The CSYNC flip-flop G60 normally has the clear input pulled low by TCLKSP and, the output to the AND gate G62 is high. When the TCLKSP input goes high, the input of the CSYNC flip-flop is enabled. At this time, the –DRRPLY clock input is low and goes high to clock the flip-flop before the TCLKSP input gets reset. If a DMA transaction is in progress, the –DMG input is high and the CSYNC flip-flop output stays low when clocked by –DRRPLY going high. For any transaction other than the DMA, the –DMG input is low and the CSYNC flip-flop output goes high when clocked by –DRRPLY going high. This allows the CSYNC output to go high and clear the TSYNC flip-flop G60, the write byte flip-flop G75, and the disable flip-flop G73 as shown in Figure 8-25.

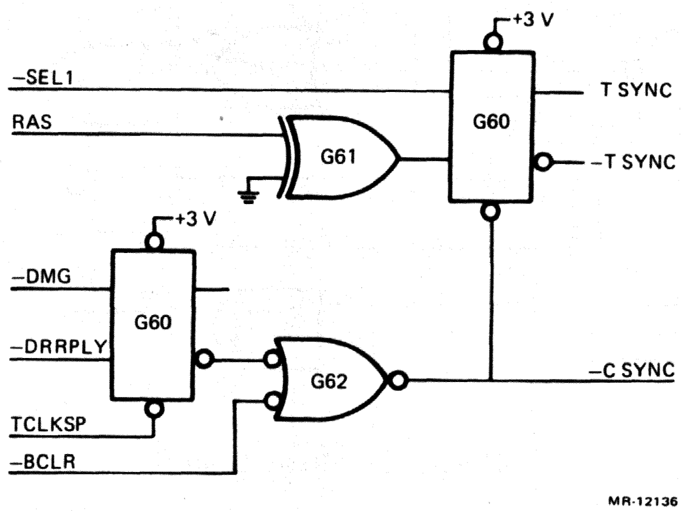


Figure 8-23 TSYNC

8.17 READ/WRITE

The read/write logic (Figure 8-24) controls the read, write, and fetch transactions for the microprocessor and supports the IAK and DMA transactions. The microprocessor controls the R/–WLB and R/–WHB inputs to select either BDIN, BDOUT, or BWTBT bus signals. To select the BDIN output, the microprocessor sets both R/–WLB and R/–WHB inputs high to NAND gate G72. The output goes low to enable the NOR gate G76 and disables the AND gates G74 and G80. The –TSYNC input to G76 is low for read/write transactions. When the –CAS input goes low, the TREAD output goes high. The TDIN output of OR gate G79 goes high, and the BDIN output of NAND gate G76 goes low. During interrupt transactions, the IAKDIN input to G79 is enabled high and causes TDIN to go high and BDIN to go low.

The microprocessor determines any write condition by setting either or both the R/–WLB or R/–WHB inputs low. The output of NAND gate G72 goes high and enables the AND gates G80 and G74. The output of flip-flop G73 is high, and the –CAS input to AND gate G74 is high. The output of AND gate G79 goes high, and the output of OR gate G78 goes high, and allows the BWTBT output to go low. At this time, the write destination address is written onto the bus. The logic now determines if the data being written is a word or a byte. The exclusive OR gate G71 monitors the R/–WLB and R/–WHB inputs, and the output goes high when the inputs are different. A high output indicates that the data is a byte; a low output indicates that the data is a word. The output goes to flip-flop G75.

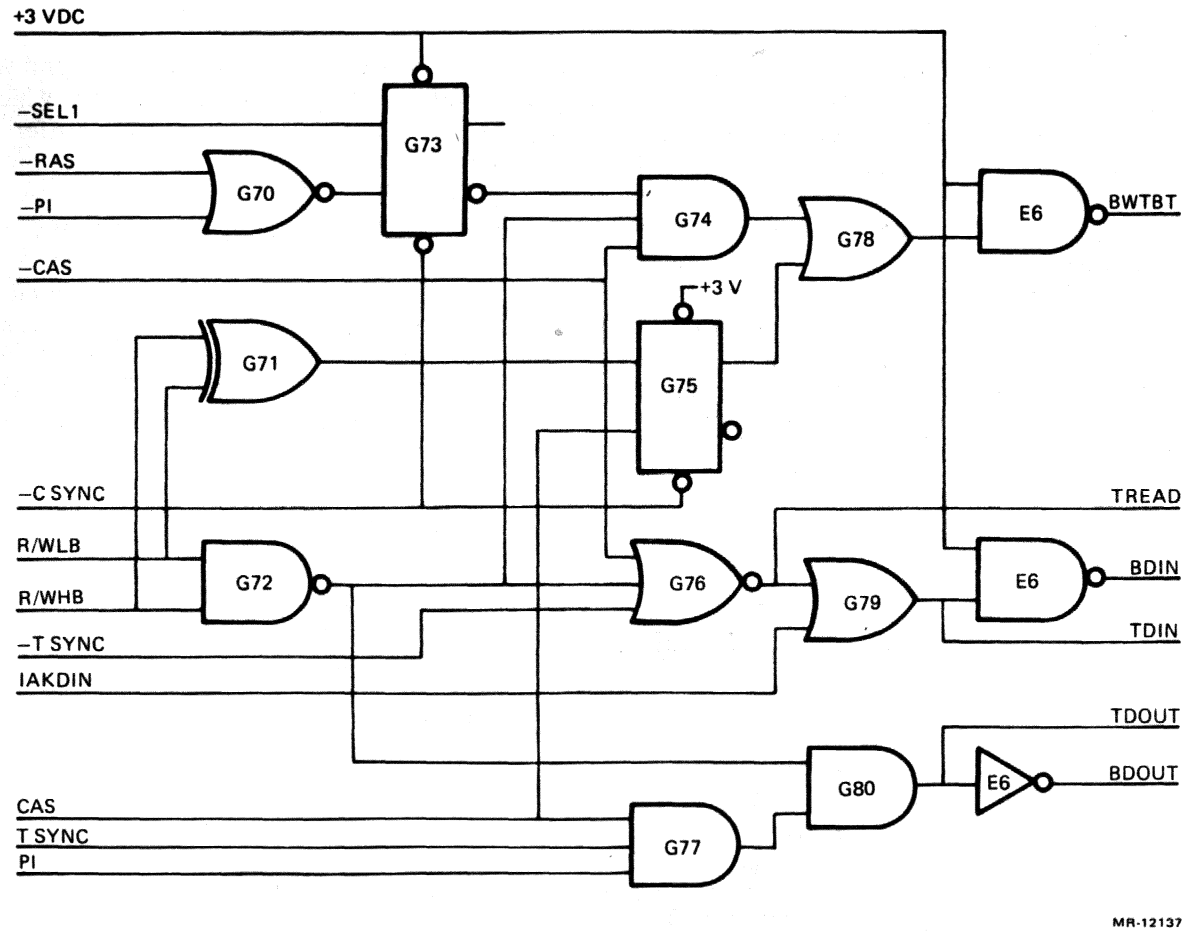


Figure 8-24 Read/Write

The microprocessor asserts CAS. The CAS input to G75 and G77 goes high, and –CAS input to G74 goes low. The –CAS input to AND gate G74 switches the output low to remove BWTBT, but the CAS input clocks flip-flop G75 and enables the WBYTE signal to G78. The output of the flip-flop G75 is high for byte transactions and low for word transactions. The BWTBT L signal will either stay asserted low for a byte transaction or be negated high for a word transaction. The TSYNC and CAS inputs to AND gate G77 are set high, and when the PI input goes high, the gate output goes high. The AND gate G80 is enabled, and the output of G77 switches the TDOUT output high. The TDOUT is inverted. The BDOUT output is enabled by going low and it writes the data word.

At the same time, the –RAS and –PI inputs to NOR gate G70 are both low, switching the output high. The high clocks flip-flop G73, and the output goes low. This inhibits the AND gate G74 when the –CAS input goes high again. The flip-flops are reset by CSYNC at the end of the transaction.

8.18 REPLY TIME-OUT

The reply time-out logic (Figure 8-25) monitors the bus BRPLY L input to indicate that an LSI-11 bus device responds to an address. The TMER flip-flop G83 output is normally set low by the RAS input to clear the flip-flop. The BRPLY L input is high and inverted so the RRPLY output is low. The –TMRP NOR gate inputs are both low, and the –TMRP output is high. The bus transaction is started by either TDIN or TDOUT inputs going high. This enables the 10 μs time-out (50 cycle slips) to start. The microprocessor starts to cycle slip while waiting for the BRPLY L input to go low, indicating the bus transaction can complete. When BRPLY L switches low, the RRPLY output goes high and the –TMRP output goes low. The TMER output stays low. If the BRPLY L does not go low and the 50 μs time-out circuit allows the 50 cycle slips, the TMER flip-flop is clocked and the TMER output goes high. TMER also forces the –TMRP output to go low. The assertion of the TMER output goes to the halt logic, and the microprocessor action is dependent upon the configuration of the module. The –TMRP output goes to the clock control and the ready logic. The RRPLY output goes to the bus control logic and enables bus data to be received during LSI-11 bus device reads.

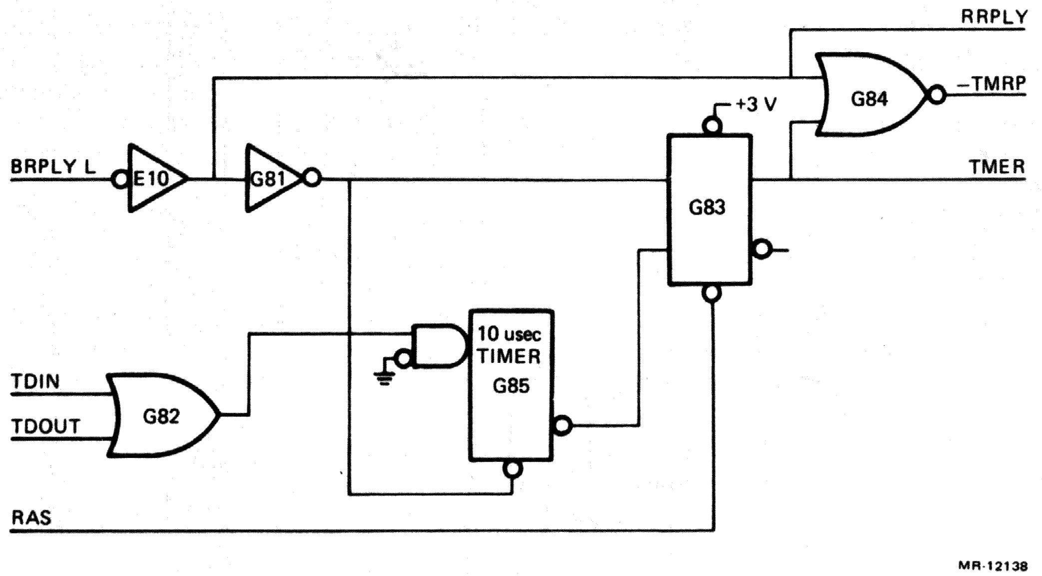


Figure 8-25 Reply Time-out

8.19 BUS CONTROL

The bus control logic (Figure 8-26) controls the transmit and receive functions of the bus transceivers. The transceivers are in transmit mode for microprocessor controlled read/write and fetch transactions to local memory, local I/O, and during LSI-11 bus writes. The transceivers go to the receive mode during an LSI-11 bus read. During DMA, the transceivers go to the receive mode to accept the local device address and will stay in this mode until the device is addressed. When a read transaction occurs, the transceivers go into the transmit mode. When the –BCLR input is high, the transceivers are able to transmit data. When –BCLR is asserted low, the transceivers are disabled. During an IAK transaction, the –IAK input to AND gate G92 goes low to disable the transceiver high byte, and the low byte goes to the receive mode to accept the vector.

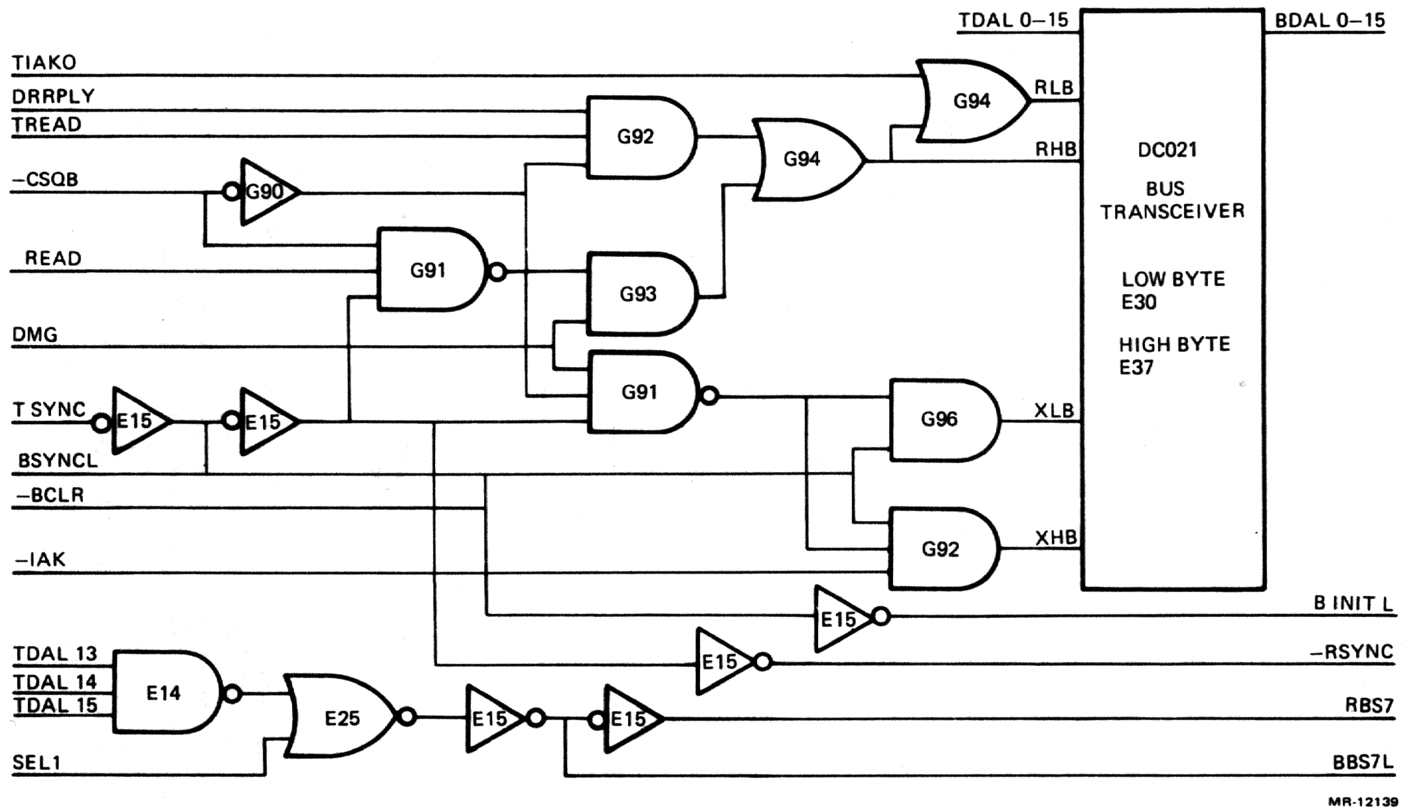


Figure 8-26 Bus Control

The receive function of the bus transceivers will override the transmit function any time the receive inputs are enabled high. When data is to be read from an LSI-11 bus device, the –CSQB input is low and inverter G90 makes it a high input to AND gate G92. The TREAD input to AND gate G92 is set high for the receive function. When the data is on the bus, the RRPLY input to AND gate G92 goes high and the output of the gate goes high. The two OR gates G94 allow the high output to enable the receive low byte and receive high byte inputs to the transceivers. The data is now read onto the TDAL bus. During an interrupt transaction, the TIAKO input goes high and enables only the receive low byte input of the transceivers.

The DMA transaction grants bus control to the external device that requested the direct memory grant. The DMG input goes high for the duration of the DMA transaction. This input enables AND gate G93 and NAND gate G91. The BSYNC L input is high and inverted low to the two NAND gates G91. This switches the NAND gate outputs high, and the receive and transmit functions are both enabled. However, the receive function overrides the transmit function, and the TDAL bus receives data from the BDAL bus. This condition stays until the bus master asserts the BDIN L input low. It is inverted high and enables the NAND gate G91. The –CSQB input is dependent upon the address received from the BDAL bus. This input is low if the address is a bus location and high if the address is for the local memory or I/O device. A low input sets the output of NAND gate G91 high and enables the receive function of the transceivers. At the same time, the –CSQB low input is inverted high, and the output of NAND gate G91 is switched low to disable the transmit function. When the –CSQB input is high indicating the local memory is being addressed, the NAND gate G91 is enabled. The –CSQB high input is also inverted low to NAND gate G91 and enables the receive function. The bus master now asserts either BDIN L or BDOUT L bus signals. The –READ input goes low for the BDIN L signal and goes high for the BDOUT L signal. If –READ goes high, it is inverted low and switches the output of NAND gate G91 high to enable the receive function. If –READ goes low, it is inverted high and switches the output of NAND gate G91 low to inhibit the receive function. The transmit function stays enabled. Therefore, when the bus master asserts the BDIN L bus signal, the data is transmitted from the module and when it asserts the BDOUT L bus signal, the data is received by the module even if it was not addressed.

The BBS7 L bus signal is enabled low when the bus addresses the I/O page during the address part of a transaction. This is the upper eight kilobytes from 56Kb to 64Kb. This page is normally reserved for I/O devices on the LSI-11 bus.

To address this page, the TDAL bus bits 13, 14, and 15 are set high and are inputs to NAND gate E14. The output is switched low and goes to the NOR gate E25. The SEL1 input to NOR gate E25 is low for read, write, and fetch transactions. When both inputs to NOR gate E25 are low, the output is switched high. This is inverted to a low for BBS7 L output and is inverted again to set RBS7 high.

CHAPTER 9
LSI-11 BUS

9.1 INTRODUCTION

The LSI-11 bus provides interconnections for LSI-11 type modules, such as processors, memories, and interfaces, to communicate with each other. Not all of the bus functions are supported by the SBC-11/21 PLUS, and only the supported functions are described in this chapter. For a complete explanation of the LSI-11 bus, see the *PDP-11 Bus Handbook*.

The LSI-11 bus has forty signal lines; eighteen are used for data and twenty-two are used for control. The SBC-11/21 PLUS supports only sixteen data lines and eighteen control lines.

There are four groups of control lines.

1. Six data transfer control lines:

   a. BBS7
   b. BDIN
   c. BDOUT
   d. BRPLY
   e. BSYNC
   f. BWTBT

2. Four direct memory access control lines:

   a. BDMGI
   b. BDMGO
   c. BDMR
   d. BSACK

3. Six interrupt control lines:

   a. BIAKI
   b. BIAKO
   c. BIRQ4
   d. BIRQ5 (not used by SBC-11/21 PLUS)
   e. BIRQ6 (not used by SBC-11/21 PLUS)
   f. BIRQ7 (not used by SBC-11/21 PLUS)

4. Six system control lines:

   a. BDCOK
   b. BPOK
   c. BHALT
   d. BINIT
   e. BREF (not used by SBC-11/21 PLUS)
   f. BEVNT

Most LSI-11 bus signals are bidirectional and use terminations for a negated (high) signal level. Modules connect to these lines via high impedance bus receivers and open collector drivers. The asserted state is produced when a bus driver asserts the line low. Although bidirectional lines are electrically bidirectional (any point on the line can be driven or received), certain lines are functionally unidirectional. These lines communicate to or from a bus master or signal source, but not both. Interrupt acknowledge (BIAK) and direct memory access grant (BDMG) signals are physically unidirectional in a daisy chain. These signals start at the processor output signal pins. Each is received on device input pins (BIAKI or BDMGI) and conditionally passed on via device output pins (BIAKO or BDMGO). The BIAK and BDMG signals are received from higher priority devices and are passed onto lower priority devices along the bus.

9.2 SBC-11/21 PLUS SINGLE-BOARD COMPUTER

The SBC-11/21 PLUS module functions on the LSI-11 bus and can act as a bus master, a bus slave, or a bus arbitrator. The module allows a DMA master to access the on-board functions. It supports only sixteen data/address lines and terminates the other lines. It also contains its own on-board memory and accesses the bus for external memory or devices. However, while accessing its on-board devices, the SBC-11/21 PLUS asserts bus control signals as it does when communicating with the LSI-11 bus. The memory maps defining on-board and external addressing are described in Chapter 2. The SBC-11/21 PLUS microprocessor supports an on-board multilevel interrupt structure, and the BIRQ4 bus interrupt control line is an active bus interrupt with a level 4 priority. Therefore, the BIRQ5, BIRQ6, and BIRQ7 bus control interrupt lines are not recognized or accepted by the SBC-11/21 PLUS module. The DMA request is recognized by the module at the lowest interrupt level. Once the DMA master has accessed the bus, there are no other interrupts until the transfer is complete or the DMA master relinquishes the bus. The module does not use or support the BREF control line for refreshing dynamic memory.

9.3 MASTER/SLAVE RELATIONSHIP

Communication between devices on the bus is asynchronous. A master/slave relationship occurs during each bus transaction. At any time, there is one device that has control of the bus. This controlling device is the bus master. The master device controls the bus when communicating with another device on the bus, the slave. The bus master (the processor or a DMA device) starts a bus transaction. The slave device responds by acknowledging the transaction in progress and by receiving data from, or transmitting data to, the bus master. LSI-11 bus control signals transmitted or received by the bus master or bus slave device must complete the sequence according to bus protocol.

The processor controls bus arbitration, i.e., which device becomes bus master at any given time. A typical example of this relationship is the processor, as master, fetching an instruction from memory, which is always a slave. Another example is a disk, as master, transferring data to memory as slave. Communication on the LSI-11 bus is interlocked so that for certain control signals issued by the master device, there must be a response from the slave in order to complete the transfer. It is the master/slave signal protocol that makes the LSI-11 bus asynchronous. The asynchronous operation eliminates the need for synchronizing with, and waiting for, clock pulses.

A bus cycle completion by the bus master requires a response from the slave device. Each bus master must include a time-out error circuit that will abort the bus cycle if the slave device does not respond to the bus transaction within 10 μs. The actual time before a time-out error occurs must be longer than the response time of the slowest peripheral or memory device on the bus. Table 9-1 provides a summary of signals that appear on the LSI-11 bus.

Table 9-1 Signal Assignments

Number of Pins	Functional Category	Signal Names
16	Data/address	BDAL0, BDAL1, BDAL2 · · · BDAL15
6	Data control	BDOUT, BRPLY, BDIN, BSYNC, BWTBT, BBS7
3	Interrupt control	BIRQ4, BIAKO, BIAKI
4	DMA control	BDMR, BDMGO, BDMGI, BSACK
5	System control	BHALT, BDCOK, BPOK, BEVNT, BINIT
3	+5 Vdc	
2	+12 Vdc	
2	−12 Vdc	
1	+5 B (battery)	
8	GND	
8	SSPARES	
4	MSPARES	
2	PSPARES	

9.4 DATA TRANSFER BUS CYCLES

Data transfer bus cycles are listed and defined in Table 9-2.

NOTE

The SBC-11/21 PLUS microcomputer performs a read transaction before every write transaction. It does not perform DATIO or DATIO(B) bus transactions as one address. It executes read-modify-write instructions by addressing the source as one transaction and addressing the destination as another transaction.

These bus cycles, executed by bus master devices, transfer 16-bit words or 8-bit bytes to or from slave devices. The bus signals that are listed in Table 9-3 are used in the data transfer operations that are described in Table 9-2. Data transfer bus cycles can be lowered to two basic types; DATI, and DATO(B). These transactions occur between the bus master and one slave device selected during the addressing section of the bus cycle.

Table 9-2 Data Transfer Operations

Bus Cycle Mnemonic	Description	Function (with respect to the bus master)
DATI	Data word input	Read
DATO	Data word output	Write
DATO(B)	Data byte output	Write byte

Table 9-3 Bus Signals Used in Data Transfer Operations

Mnemonic	Description	Function
BDAL<15:00> L	16 data/address lines	BDAL<15:00> L are used for word and byte transfers
BSYNC L	Bus cycle control	Strobe signal
BDIN L	Data input indicator	Strobe signal
BDOUT L	Data output indicator	Strobe signal
BRPLY L	Slave's acknowledge of bus cycle	Strobe signal
BWTBT L	Write/byte control	Control signal
BBS7	I/O device select; indicates address is in the I/O page	Control signal

9.4.1 Bus Cycle Protocol

Before starting a bus cycle, the previous bus transaction must have been completed (BSYNC L negated) and the device must become bus master. The bus cycle can be divided into two parts; an addressing section and a data transfer section. During the addressing section, the bus master outputs the address for the correct slave device, memory location, or device register. The selected slave device responds by latching the address bits and holding this condition for the duration of the bus cycle until BSYNC L becomes negated. During the data transfer section, the actual data transfer occurs.

Device Addressing – The device addressing section of a data transfer bus cycle has an address set-up and deskew time and an address hold and deskew time. During the address set-up and deskew time, the bus master:

1. Asserts BDAL<15:00> L with the correct slave device address bits.

2. Asserts BBS7 L if a device in the I/O page (56Kb–64Kb for SBC-11/21 PLUS) is being addressed. (Devices in the I/O page ignore BDAL<15:13> and decode BBS7 L with BDAL<12:00>.)

3. Asserts BWTBT L if the cycle is a DATO(B) bus cycle. (Inactive BWTBT L indicates a DATI or DATIO(B) operation.)

4. Asserts BSYNC at least 150 ns after BDAL<15:00> L, BBS7 L, and BWTBT L are valid.

The BBS7 L address and BWTBT L signal must be asserted at the slave bus receiver for at least 75 ns before BSYNC goes active. The address hold and deskew time start after BSYNC L is asserted.

The slave device uses the active BSYNC L bus receiver output to clock BDAL address bits, BBS7 L and BWTBT L, into its internal logic. BDAL<15:00> L, BBS7 L, and BWTBT L will stay active for 25 ns (minimum) after the BSYNC L bus receiver goes active. BSYNC L stays active for the duration of the bus cycle.

Memory devices usually do not respond to addresses in the I/O page; however, some system applications may permit memory to reside in the I/O page for use as DMA buffers, read only memory bootstraps, or diagnostics, etc.

DATI – The DATI bus cycle, shown in Figure 9-1, is a read operation. During DATI, data is input to the bus master. Data uses 16-bit word transfers over the bus. During the data transfer section of the DATI bus cycle, the bus master asserts BDIN L 100 ns (minimum) after BSYNC L is asserted. In response to BDIN L active, the slave device:

1. Asserts BRPLY L after receiving BDIN L and 125 ns (maximum) before BDAL bus driver data bits are valid.

2. Asserts BDAL<15:00> L with the addressed data.

When the bus master receives BRPLY L, the bus master:

1. Waits at least 200 ns deskew time and then accepts input data at BDAL<15:00> L bus receivers.

2. Negates BDIN L 150 ns (minimum) to 2 μs (maximum) after BRPLY L goes active.

NOTE

Continuous assertion of BSYNC L keeps control of the bus under the bus master, and the previously addressed slave device remains selected. Also, a slow slave device can hold off data transfers to itself by keeping BRPLY L asserted. This will cause the master to keep BSYNC L asserted.

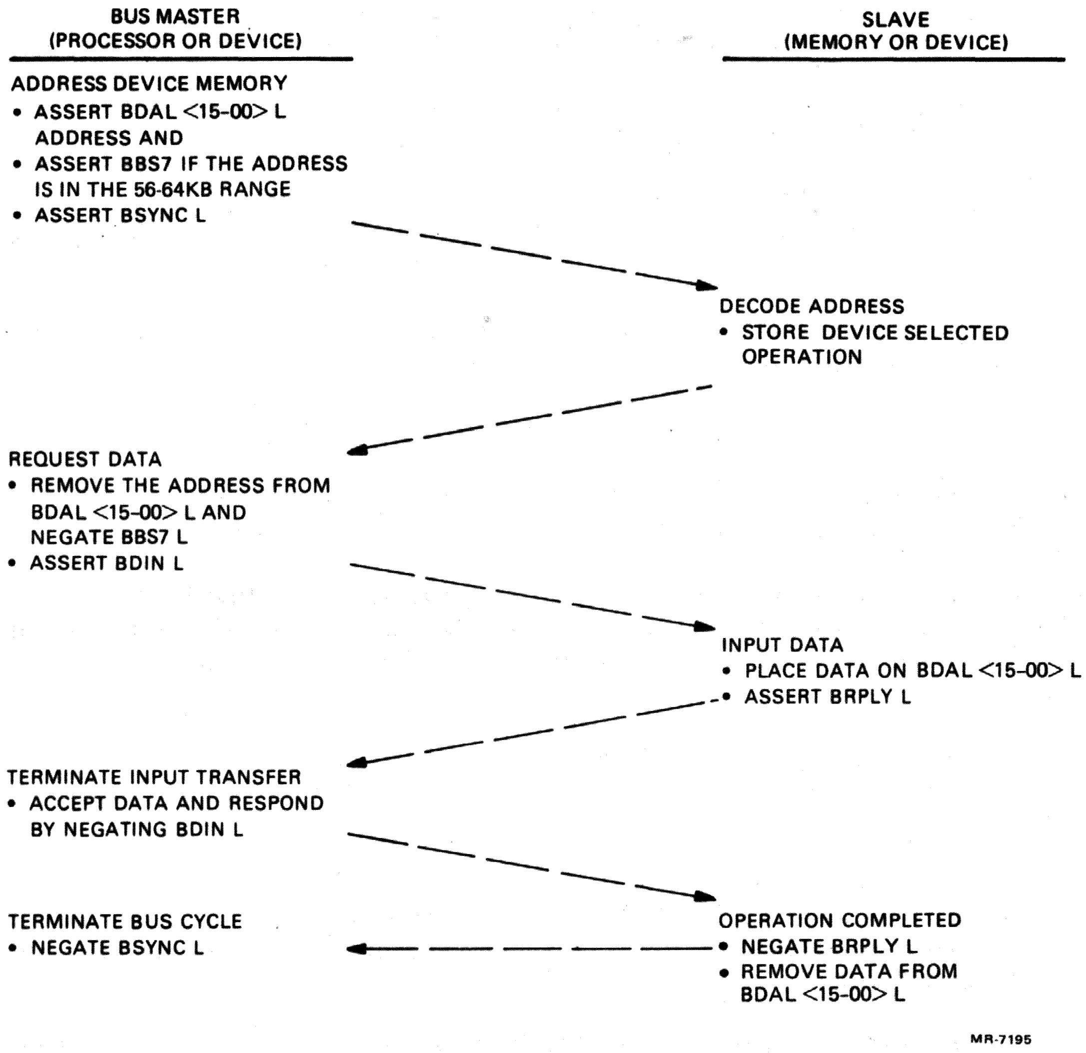


Figure 9-1 DATI Bus Cycle

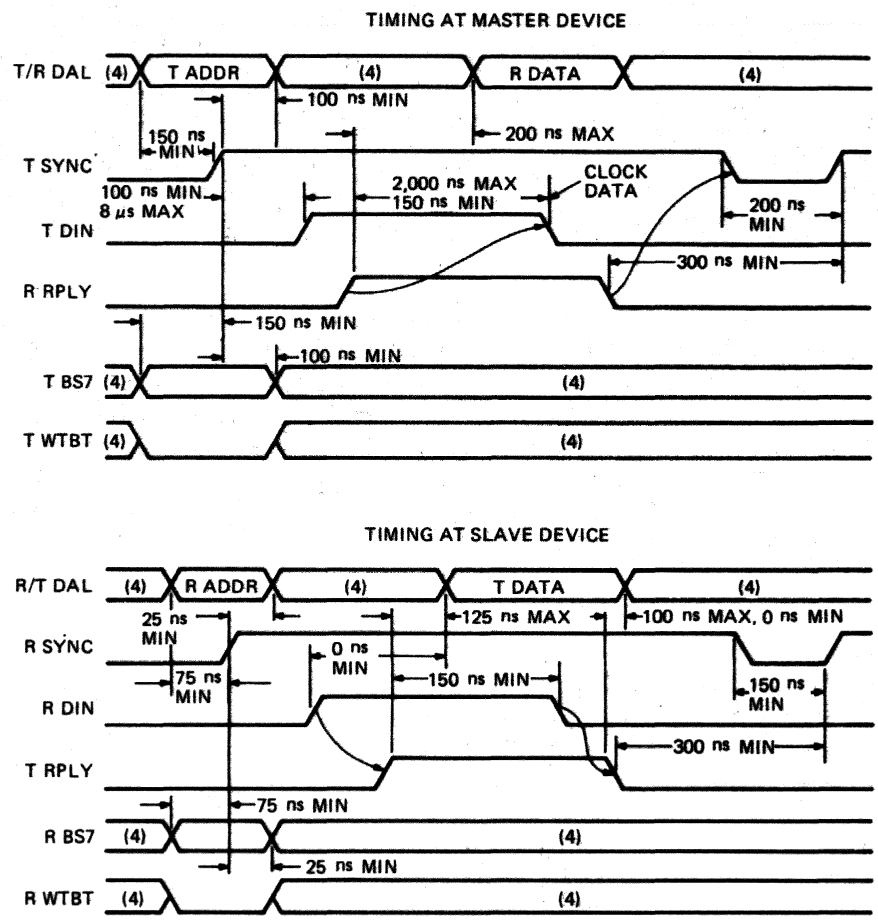
The slave device responds to BDIN L negation by negating BRPLY L and removing read data from BDAL bus drives. BRPLY L must be negated 100 ns (maximum) before removal of read data. The bus master responds to the negated BRPLY L by negating BSYNC L.

Two conditions must be met for the next BSYNC L assertion:

1. BSYNC L must remain negated for 200 ns (minimum).

2. BSYNC L must not become asserted within 300 ns of the previous BRPLY L negation.

Figure 9-2 illustrates DATI bus cycle timing.



NOTES:

1. TIMING SHOWN AT MASTER AND SLAVE DEVICE BUS DRIVER INPUTS AND BUS RECEIVER OUTPUTS
2. SIGNAL NAME PREFIXES ARE DEFINED BELOW:
   T = BUS DRIVER INPUT
   R = BUS RECEIVER OUTPUT
3. BUS DRIVER OUTPUT AND BUS RECEIVER INPUT SIGNAL NAMES INCLUDE A "B" PREFIX
4. DO NOT CARE CONDITION

MR-7180

Figure 9-2 DATI Bus Cycle Timing

DATO(B) – DATO(B), illustrated in Figure 9-3, is a write operation. Data is transferred in 16-bit words (DATO) or 8-bit bytes (DATO(B)) from the bus master to the slave device. The data transfer output can occur after the addressing section of a bus cycle when BWTBT L has been asserted by the bus master.

The data transfer section of a DATO(B) bus cycle makes a data set-up and deskew time and a data hold and deskew time. During the data set-up and deskew time, the bus master outputs the data on BDAL<15:00> L at least 100 ns after the BSYNC L is asserted. If it is a word transfer, the bus master negates BWTBT L at least 100 ns after BSYNC L assertion. BWTBT L stays negated for the length of the bus cycle. If the transfer is a byte transfer, BWTBT L remains asserted. During a byte transfer, BDAL 00 L selects the high or low byte. This occurs while in the addressing section of the cycle. If asserted, the high byte (BDAL<15:08> L) is selected; otherwise, the low byte (BDAL<07:00> L) is selected. The bus master asserts BDOUT L at least 100 ns after BDAL and BWTBT L bus drives are stable. The slave device responds by asserting BRPLY L within 10 μs to avoid bus time-out. This completes the data set-up and deskew time.

During the data hold and deskew time, the bus master receives BRPLY L and negates BDOUT L. BDOUT L must stay asserted for at least 150 ns after receiving BRPLY L before being negated by the bus master. BDAL<15:00> L bus drivers stay asserted for at least 100 ns after BDOUT L negation. The bus master then negates BDAL inputs. During this time, the slave device senses BDOUT L negation. The data is accepted, and the slave device negates BRPLY L. The bus master responds by negating BSYNC L. However, the processor will not negate BSYNC L for at least 175 ns after negating BDOUT L. This completes the DATO(B) bus cycle. Before the next cycle, BSYNC L must stay unasserted for at least 200 ns. Figure 9-4 shows the DATO(B) bus cycle timing.

9.4.2 Direct Memory Access

DMA is started after the processor (normally bus master) has passed bus mastership to the highest priority DMA device that is requesting the bus. The processor arbitrates all requests and grants the bus to the DMA device electrically closest to it. A DMA device remains a bus master until it relinquishes its mastership. The following control signals are used during bus arbitration.

1.	BDMGI L	DMA grant input
2.	BDMGO L	DMA grant output
3.	BDMR L	DMA request line
4.	BSACK L	Bus grant acknowledge

A DMA transaction can be divided into three phases:

1. Bus mastership acquisition phase
2. Data transfer phase
3. Bus mastership relinquish phase

During the bus mastership acquisition phase, a DMA device requests the bus by asserting BDMR L. The processor arbitrates the request and starts the transfer of bus mastership by asserting BDMGO L. The maximum time between BDMR L assertion and BDMGO L assertion is DMA latency. This is processor dependent. BDMGO L/BDMGI L is one signal that is daisy chained through each module in the backplane. It is driven out of the processor on the BDMGO L pin, enters each module on the BDMGI L pin, and exits on the BDMGO L pin. This signal passes through the modules in descending order of priority until it is stopped by the requesting device. The requesting device blocks the output of BDMGO L and asserts BSACK L. If BDMR L is continuously asserted, the bus will be hung.

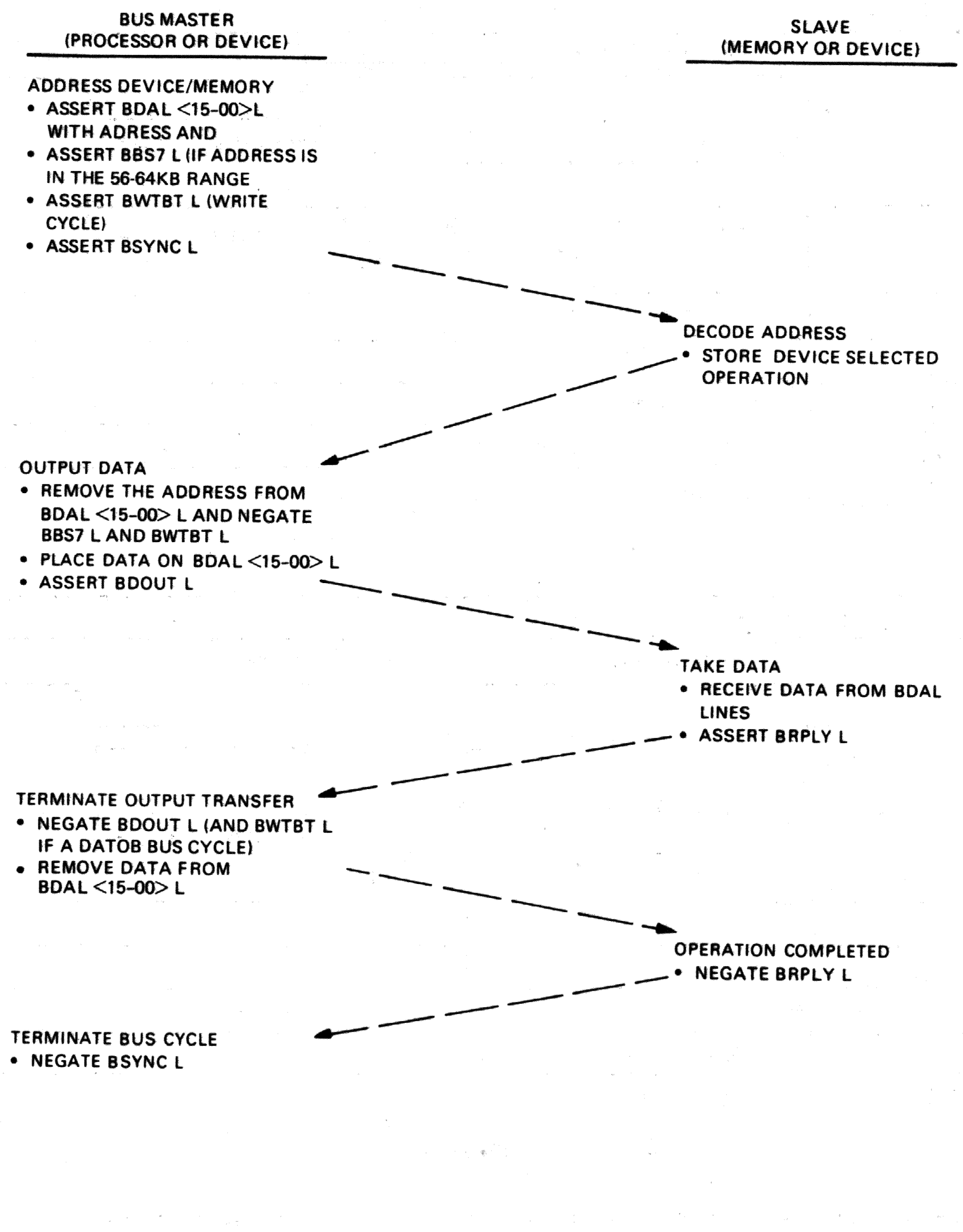
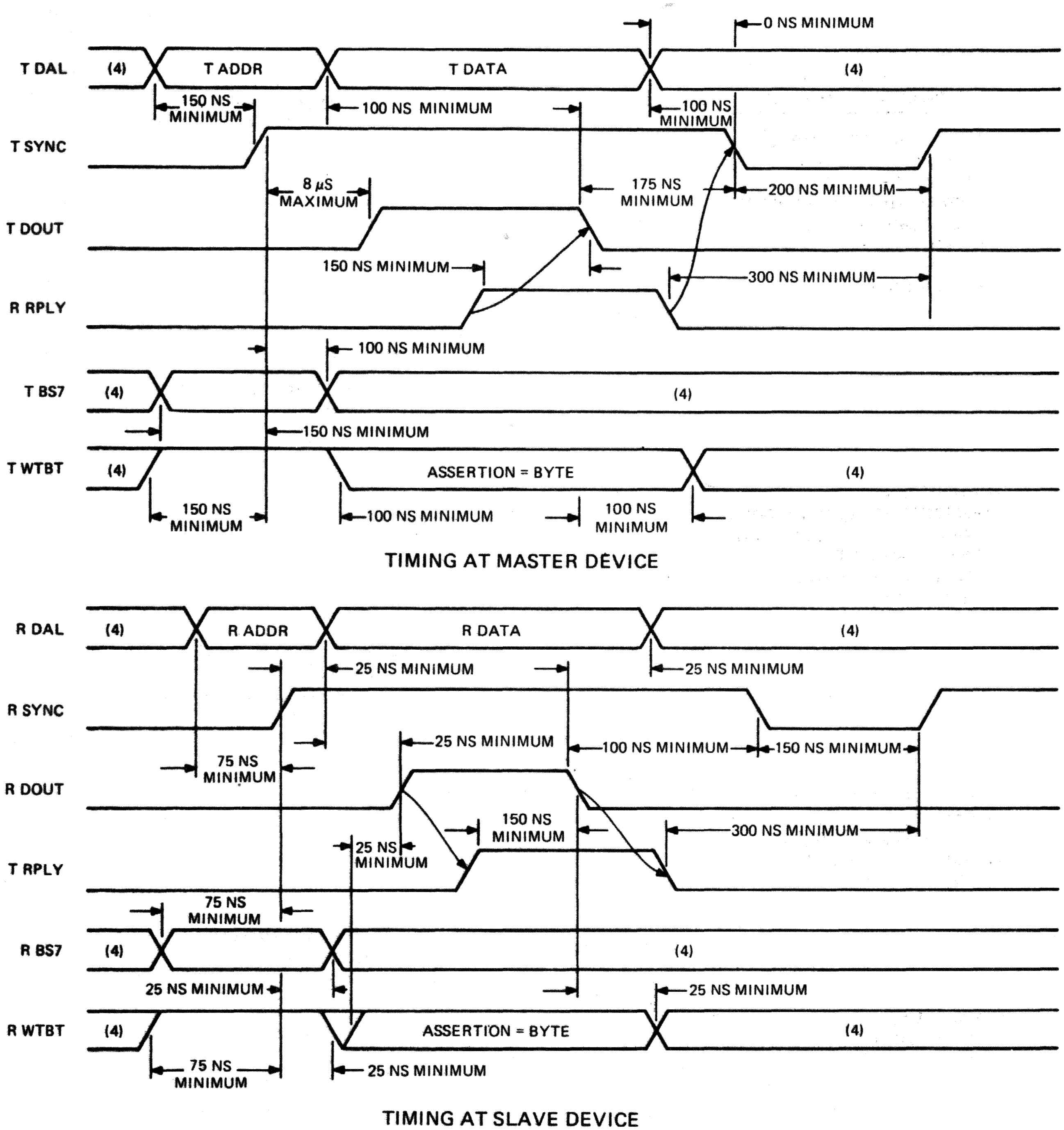


Figure 9-3 DATO or DATOB Bus Cycle



NOTES:

1. TIMING SHOWN AT MASTER AND SLAVE DEVICE BUS DRIVER INPUTS AND BUS RECEIVER OUTPUTS.
2. SIGNAL NAME PREFIXES ARE DEFINED BELOW:
   T = BUS DRIVER INPUT
   R = BUS RECEIVER OUTPUT
3. BUS DRIVER OUTPUT AND BUS RECEIVER INPUT SIGNAL NAMES INCLUDE A "B" PREFIX.
4. DON'T CARE CONDITION.

MR 1179

Figure 9-4 DATO or DATOB Bus Cycle Timing

During the data transfer phase, the DMA device continues asserting BSACK L. The actual data transfer is performed as described previously.

NOTE

If multiple data transfers are performed during this phase, consideration must be given to the use of the bus for other system functions.

The DMA device can assert BSYNC L for a data transfer 250 ns (minimum) after it receives BDMGI L and its BSYNC L and BRPLY L become negated.

During the bus mastership relinquish phase, the DMA device relinquishes the bus by negating BSACK L. This occurs after completing (or aborting) the last data transfer cycle (BRPLY L negated). BSACK L may be negated up to a maximum of 300 ns before negating BSYNC L. Figure 9-5 shows the DMA protocol, and Figure 9-6 shows the DMA request/grant timing.

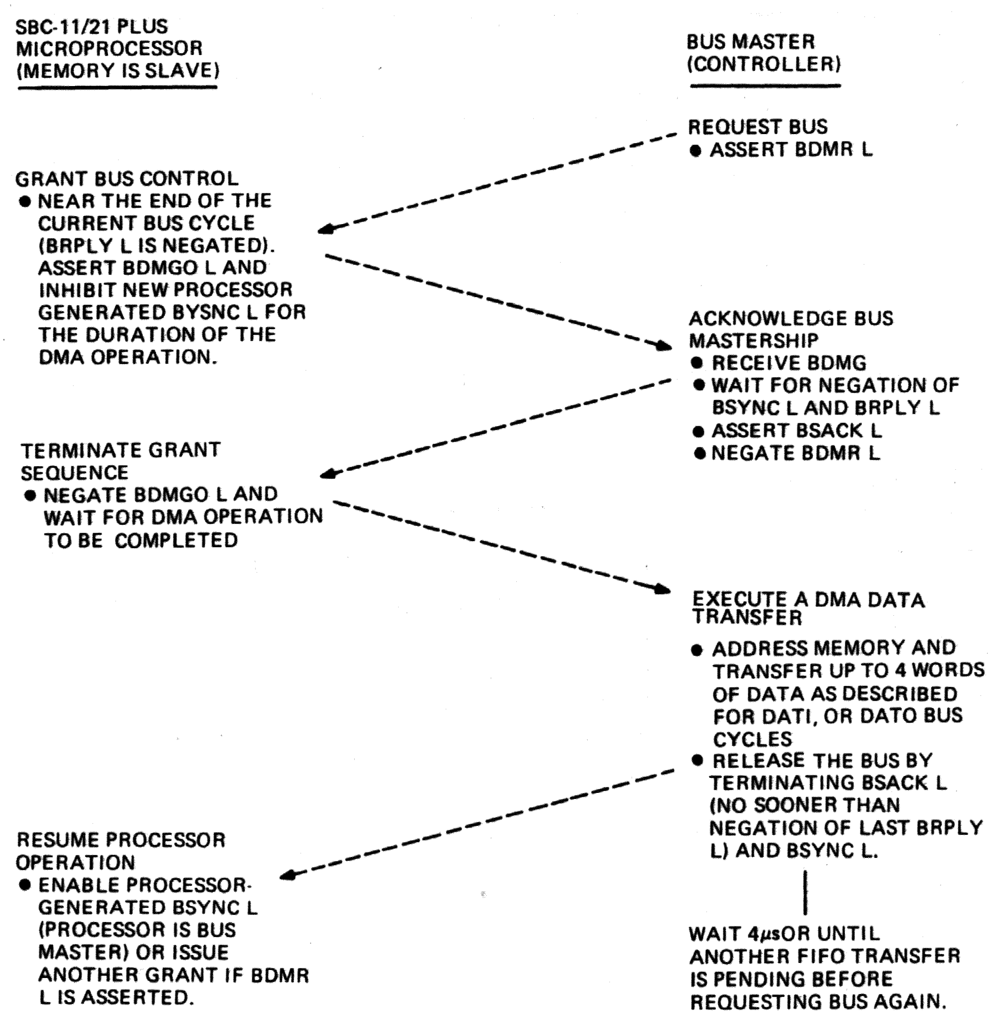


Figure 9-5 DMA Protocol

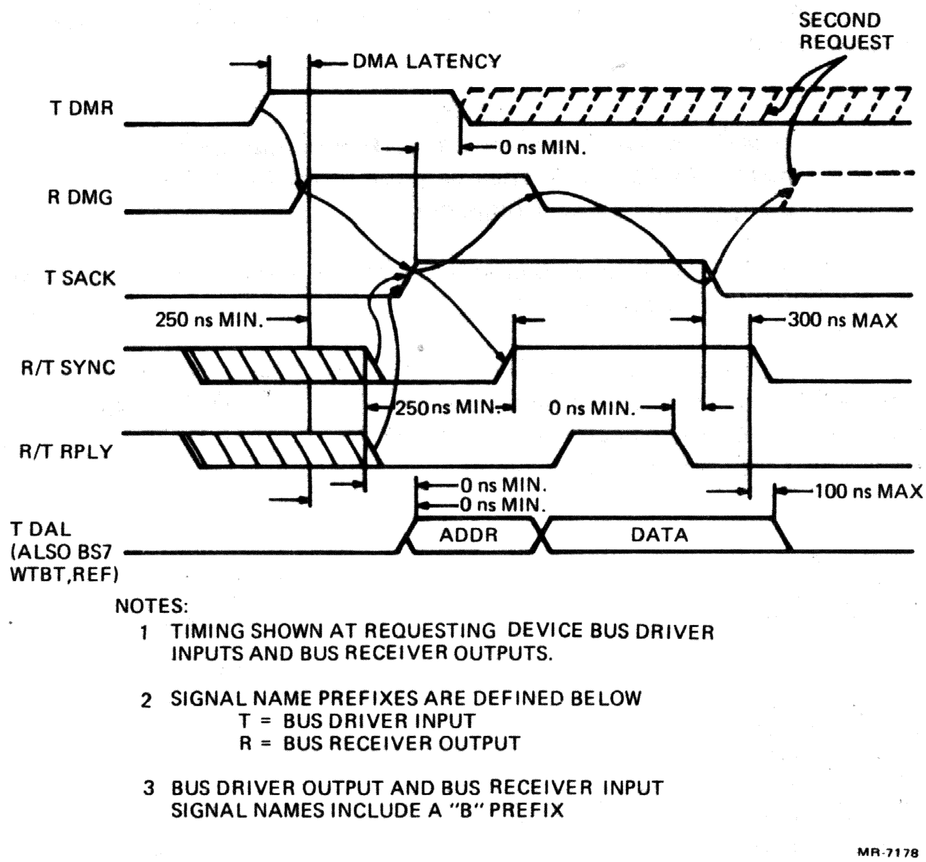


Figure 9-6 DMA Request/Grant Timing

9.5 INTERRUPTS

The LSI-11 bus signals used in interrupt transactions are:

1.	BIRQ4 L	Interrupt request priority level 4
2.	BIAKI L	Interrupt acknowledge input
3.	BIAKO L	Interrupt acknowledge output
4.	BDAL<15:00> L	Data/address lines
5.	BDIN L	Data input strobe
6.	BRPLY L	Reply

9.5.1 Device Priority

The SBC-11/21 PLUS supports only one method of device priority arbitration; position defined arbitration (priority is determined only by electrical position on the bus). The closer a device is to the processor, the higher its priority.

9.5.2 Interrupt Protocol

Interrupt protocol on the SBC-11/21 PLUS has three phases:

1. Interrupt request phase
2. Interrupt acknowledge and priority arbitration phase
3. Interrupt vector transfer phase

Figure 9-7 shows the interrupt request/acknowledge sequence.

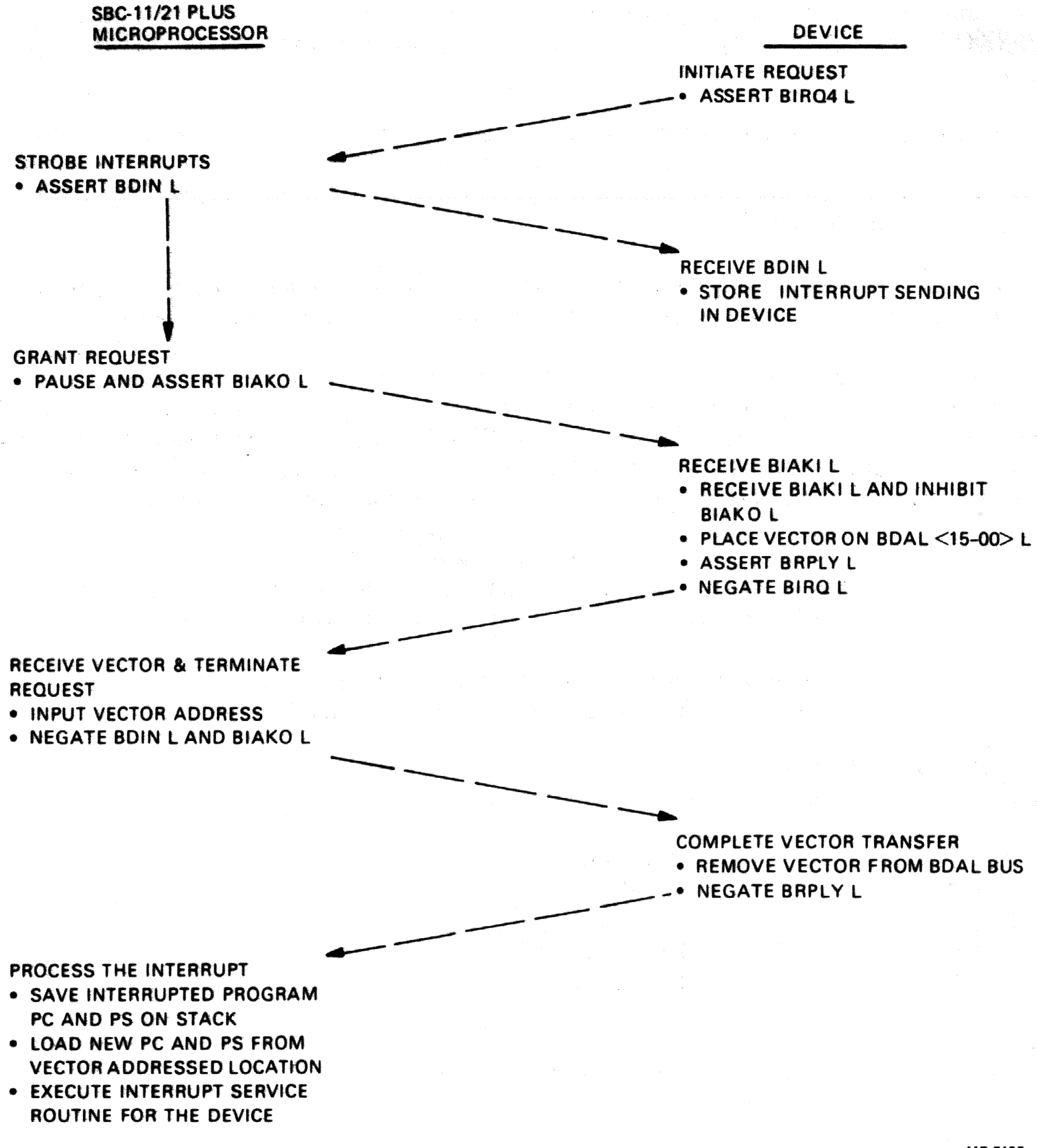


Figure 9-7 Interrupt Request/Acknowledge Sequence

The interrupt request phase starts when a device meets its specific conditions for interrupt requests (e.g., the device is ready, done, or an error has occurred). The interrupt enable bit in a device status register must be set. The device then sets up the interrupt by asserting the interrupt request line. BIRQ4 L is the only hardware priority level on the SBC-11/21 PLUS and is asserted for all interrupt requests. The interrupt request line stays asserted until the request is acknowledged.

During the interrupt acknowledge and priority arbitration phase, the SBC-11/21 PLUS processor will acknowledge interrupts under the following conditions:

1. The device interrupt priority is higher than the current PS<7:5>.

2. The processor has completed instruction execution, and no additional bus cycles are waiting.

The processor acknowledges the interrupt request by asserting BDIN L, and, 225 ns (minimum) later, asserting BIAKO L. The device electrically closest to the processor receives the acknowledge on its BIAKI L bus receiver.

When the device receives the acknowledge, it reacts as follows:

1. If not requesting an interrupt, the device asserts BIAKO L, and the acknowledge moves to the next device on the bus.

2. If the device was requesting an interrupt, the acknowledge is blocked using the leading edge of BDIN L and arbitration is granted. The interrupt vector transfer phase begins.

The interrupt vector transfer phase is enabled by BDIN L and BIAKI L. The device responds by asserting BRPLY L and its BDAL<15:00> L bus driver inputs with the vector address bits. The BDAL bus driver inputs must be stable within 125 ns (maximum) after BRPLY L is asserted. The processor then inputs the vector address and negates BDIN L and BIAKO L. The device then negates BRPLY L and, 100 ns (maximum) later, removes the vector address bits. The processor then enters the device's service routine.

NOTE

Propagation delay from BIAKI L to BIAKO L must not be greater than 500 ns per LSI-11 bus slot.

The device must assert BRPLY L within 10 μs (maximum) after the processor asserts BIAKI L.

9.6 CONTROL FUNCTIONS

The following LSI-11 bus signals provide control functions.

1.	BHALT L	Processor halt
2.	BINIT L	Initialize
3.	BPOK H	Power OK
4.	BDCOK H	DC power OK
5.	BEVNT L	External event

9.6.1 Halt

Refer to Chapter 2 for an explanation of the BHALT L response.

9.6.2 Initialization

Devices on the bus are initialized when BINIT L is asserted. The microprocessor can assert BINIT L as a result of executing a RESET instruction or as part of a power-up sequence. BINIT L is asserted for approximately 17 μs when RESET is executed.

9.6.3 Power Status

Power status protocol is controlled by two signals, BPOK H and BDCOK H. These signals are driven by some external device (usually the power supply).

BPOK H – When asserted, BPOK H indicates that there is at least an 8 ms reserve of dc power and that BDCOK H has been asserted for at least 70 ms. Once BPOK H has been asserted, it must stay asserted for at least 3 ms. The negation of this line, the first event in the power fail sequence, indicates that power is failing and that only 4 ms of dc power reserve remain.

BDCOK H – When asserted, BDCOK H indicates that dc power has been stable for at least 3 ms. Once asserted, this line stays asserted until the power fails. Its negation indicates that only 5 μs of dc power reserve remain.

9.6.4 Power-Up/Power-Down Protocol

Power-up protocol (Figure 9-8) begins when the power supply applies power with BDCOK H negated. This forces the processor to assert BINIT L. When the dc voltages are stable, the power supply, or other external device, asserts BDCOK H. The processor responds by clearing the PSW. BINIT L remains asserted until the assertion of BDCOK H. The processor continues to test for BPOK H until it is asserted. The power supply asserts BPOK H 70 ms (minimum) after BDCOK H is asserted. The processor then performs its power-up sequence. Normal power must be maintained at least 3 ms before a power-down sequence can start.

A power-down sequence starts when the power supply negates BPOK H. When the current instruction is completed, the microprocessor traps to a power-down routine at location 24. The routine must provide for loading 340 into the PSW, execute a RESET instruction, and terminate in a WAIT instruction or branch on itself. There should be no DMA requests issued after the RESET is executed. This prevents any possible memory destruction in the battery supported system as the dc voltages fail.

NOTE

SBC-11/21 PLUS does not generate BINIT L during the power-down sequence. The power-down routine must therefore include a RESET instruction to set bus devices into a known state.

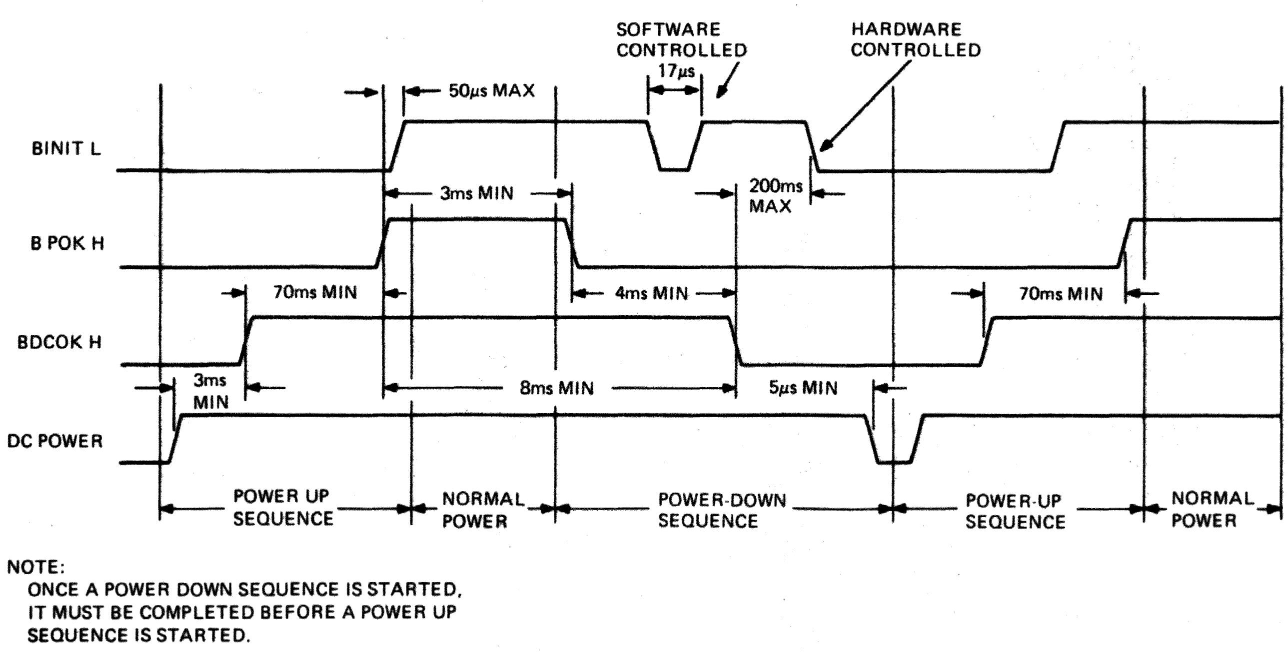


Figure 9-8 Power-Up/Power-Down Timing

9.7 LSI-11 BUS ELECTRICAL CHARACTERISTICS

Configuring LSI-11 bus systems requires an understanding of its transmission line characteristics. For a discussion of these characteristics, see the *PDP-11 Bus Handbook*.

9.8 MODULE CONTACT FINGER IDENTIFICATION

All Digital plug-in modules, including the SBC-11/21 PLUS, use the same contact finger (pin) identification system. The LSI-11 bus is based on the use of double-height modules that plug into a 2-slot bus connector. Each slot contains thirty-six lines (eighteen each on both the component and solder sides of the circuit board).

Slots, shown as row A and row B in Figure 9-9, include a numeric identifier for the side of the module. The component side is defined as side 1; the solder side is defined as side 2. Letters A through V (except G, I, O, and Q) identify a specific pin on a side of a slot. Table 9-4 lists and identifies the bus pins of the double-height module. For a summary, refer to Table 1-1. The bus pin identifier terminating with a 1 is found on the component side of the board; a bus pin identifier terminating with a 2 is found on the solder side of the board. A typical pin is defined as follows:

AE2: row A, pin E, side 2

The positioning slot between the two rows of pins matches with a guide on the connector block for correct module positioning.

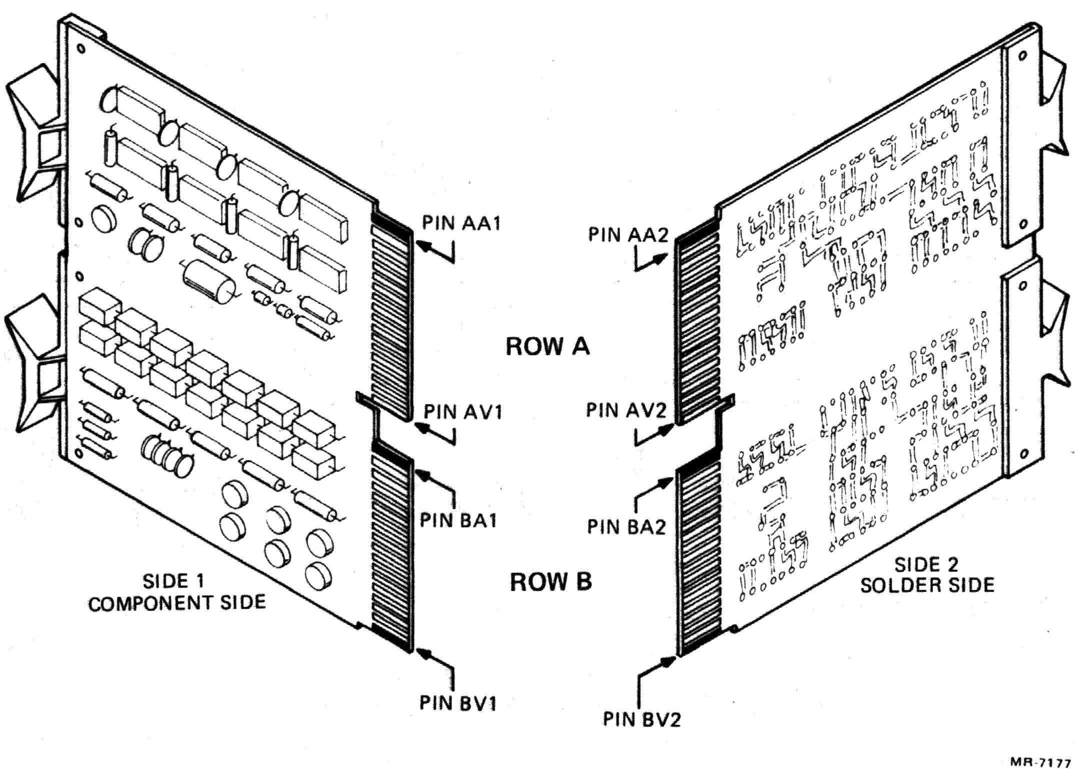


Figure 9-9 Double-Height Module Contact Finger Identification

Table 9-4 Bus Pin Identifiers

Bus Pin	Mnemonic	Description
AE1	SSPARE1 (alternate +5B)	Special spare – not assigned or bused in Digital cable or backplane assemblies; available for user connection. Optionally, this pin may be used for +5 V battery (+5B) backup power to keep critical circuits alive during power failures. A jumper is required on LSI-11 bus options to open (disconnect) the +5B circuit in systems that use this line as SSPARE1.
AF1	SSPARE2	Special spare – not assigned or bused in Digital cable or backplane assemblies; available for user interconnection.
AJ1	GND	Ground – system signal ground and dc return.
AK1	MSPAREA	
AL1	MSPAREA	Maintenance spare – normally connected together on the backplane at each option location (not a bused connection).
AM1	GND	Ground – system signal ground and dc return.
AN1	BDMR L	Direct memory access (DMA) request – a device asserts this signal to request bus mastership. The processor arbitrates bus mastership between itself and all DMA devices on the bus. If the processor is not bus master (it has completed a bus cycle, and BSYNC L is not being asserted by the processor), it grants bus mastership to the requesting device by asserting BDMGO L. The device responds by negating BDMR L and asserting BSACK L.
AP1	BHALT L	Processor halt – refer to Chapter 2.
AT1	GND	Ground – system signal ground and dc return.
AU1	PSPARE1	Spare – not assigned; customer usage not recommended; prevents damage when modules are inserted upside down.
AV1	+5B	+5 V battery power – secondary +5 V power connection. Battery power can be used with certain devices.
BA1	BDCOK H	DC power OK – power supply-generated signal that is asserted when there is sufficient dc voltage available to sustain reliable system operation.
BB1	BPOK H	Power OK – asserted by the power supply 70 ms after BDCOK. Negated when ac power drops below the value required to sustain power (approximately 75% of nominal). When negated during processor operation, a power fail trap sequence is initiated.

Table 9-4 Bus Pin Identifiers (Cont)

Bus Pin	Mnemonic	Description
BH1	SSPARE8	Special spare – not assigned or bused in Digital cable and backplane assemblies; available for user interconnection.
BJ1	GND	Ground – system signal ground and dc return.
BK1 BL1	MSPAREB MSPAREB	Maintenance spare – normally connected together on the backplane at each option location (not a bused connection).
BM1	GND	Ground – system signal ground and dc return.
BN1	BSACK L	This signal is asserted by a DMA device in response to the processor's BDMGO L signal, indicating that the DMA device is bus master.
BR1	BEVNT L	External event interrupt request – when asserted, the processor responds (if PS bit 7 is zero) by entering a service routine via vector address 100. A typical use of this signal is a line time-clock interrupt.
BS1	PSPARE4	Power spare 4 – not assigned a function; not recommended for use.
BT1	GND	Ground – system signal ground and dc return.
BU1	PSPARE2	Power spare 2 – not assigned a function; not recommended for use. If a module is using −12 V (on pin AB2) and if the module is accidentally inserted upside down in the backplane, −12 Vdc appears on pin BU1.
BV1	+5	+5 V power – normal +5 Vdc system power.
AA2	+5	+5 V power – normal +5 Vdc system power.
AB2*	−12	−12 V power – −12 Vdc (optional) power for devices requiring this voltage.
AC2	GND	Ground – system signal ground and dc return.
AD2	+12	+12 V power – 12 Vdc system power.
AE2	BDOUT L	Data output – BDOUT, when asserted, implies that valid data is available on BDAL<0:15> L and that an output transfer, with respect to the bus master device, is taking place. BDOUT L is deskewed with respect to data on the bus. The slave device responding to the BDOUT L signal must assert BRPLY L to complete the transfer.

Table 9-4 Bus Pin Identifiers (Cont)

Bus Pin	Mnemonic	Description
AF2	BRPLY L	Reply – BRPLY L is asserted in response to BDIN L or BDOUT L and during IAK transactions. It is generated by a slave device to indicate that it has placed its data on the BDAL bus or that it has accepted output data from the bus.
AH2	BDIN L	Data input – BDIN L is used for two types of bus operation: 1. When asserted during BSYNC L time, BDIN L implies an input transfer with respect to the current bus master and requires a response (BRPLY L). BDIN L is asserted when the master device is ready to accept data from a slave device. 2. When asserted without BSYNC L, BDIN L indicates that an interrupt operation is occurring. The master device must deskew input data from BRPLY L.
AJ2	BSYNC L	Synchronize – BSYNC L is asserted by the bus master device to indicate that it has placed an address on BDAL<0:15> L. The transfer is in process until BSYNC L is negated.
AK2	BWTBT L	Write/byte – BWTBT L is used in two ways to control a bus cycle: 1. It is asserted at the leading edge of BSYNC L to indicate that an output sequence is to follow (DATO or DATO(B)), rather than an input sequence. 2. It is asserted during BDOUT L, in a DATO(B) bus cycle, for byte addressing.
AL2	BIRQ4 L	Interrupt request priority level 4 – a level 4 device asserts this signal when its interrupt enable and interrupt request flips-flops are set. If the PSW bit 7 is zero, the processor responds by acknowledging the request by asserting BDIN L and BIAKO L.
AM2 AN2	BIAKI L BIAKO L	Interrupt acknowledge – in accordance with interrupt protocol, the processor asserts BIAKO L to acknowledge receipt of an interrupt. The bus transmits this to BIAKI L of the device electrically closest to the processor. This device accepts the interrupt acknowledge under two conditions: 1. The device requested the bus by asserting BIRQ4 L.

Table 9-4 Bus Pin Identifiers (Cont)

Bus Pin	Mnemonic	Description
		2. The device has the highest priority interrupt request on the bus at that time. If these conditions are not met, the device asserts BIAKO L to the next device on the bus. This process continues in a daisy chain fashion until the device with the highest interrupt priority receives the interrupt acknowledge signal.
AP2	BBS7 L	Bank 7 select – the bus master asserts this signal to reference the I/O page (including that portion of the I/O page reserved for nonexistent memory). The address in BDAL<0:12> L when BBS7 L is asserted is the address within the I/O page.
AR2 AS2	BDMGI L BDMGO L	Direct memory access grant – the bus arbitrator asserts this signal to grant bus mastership to a requesting device according to bus mastership protocol. The signal is passed in a daisy chain from the arbitrator (as BDMGO L) through the bus to BDMGI L of the next priority device (electrically closest device on the bus). This device accepts the grant only if it requested to be bus master (by a BDMR L). If not, the device passes the grant (asserts BDMGO L) to the next device on the bus. This process continues until the requesting device acknowledges the grant.
AT2	BINIT L	Initialize – this signal is used for system reset. All devices on the bus are to return to a known, initial state (i.e., registers are reset to zero, and logic is reset to state zero). Exceptions should be completely documented in programming and engineering specifications for the device.
AU2 AV2	BDAL0 L BDAL1 L	Data/address lines – these two lines are part of the sixteen-line data/address bus over which address and data information are communicated. Address information is first placed on the bus by the bus master device. The same device then either receives input data from, or outputs data to, the addressed slave device or memory over the same bus lines.
BA2	+5	+5 V power – normal +5 Vdc system power.
BB2	−12	−12 V power – −12 Vdc (optional) power for devices requiring this voltage.
BC2	GND	Ground – system signal ground and dc return.
BD2	+12	+12 V power – +12 V system power.

Table 9-4 Bus Pin Identifiers (Cont)

Bus Pin	Mnemonic	Description
BE2	BDAL2 L	Data /address lines – these fourteen lines are part of the sixteen-line data/address bus previously described.
BF2	BDAL3 L	
BH2	BDAL4 L	
BJ2	BDAL5 L	
BK2	BDAL6 L	
BL2	BDAL7 L	
BM2	BDAL8 L	
BN2	BDAL9 L	
BP2	BDAL10 L	
BR2	BDAL11 L	
BS2	BDAL12 L	
BT2	BDAL13 L	
BU2	BDAL14 L	
BV2	BDAL15 L	

* LSI-11 modules that require negative voltages contain an inverter circuit (on each module) that generates the required voltage(s). Hence, −12 V power is not required with Digital-supplied options.

APPENDIX A
INSTRUCTION TIMING

The fetch and execute times listed in Table A-1 assume that the SBC-11/21 PLUS is transacting with local devices that do not require cycle slips when accessed.

Table A-1 Instruction Timing

Single Operand Instructions	Destination Mode	Fetch and Execute Time (μs)	Number of Bus Transactions	Number of Microcycles
CLR(B), COM(B),	0	2.44	1	4
INC(B), DEC(B),	1	4.27	3	7
NEG(B), ROR(B),	2	4.27	3	7
ROL(B), ASR(B),	3	5.49	4	9
ASL(B), SWAP,	4	4.88	3	8
ADC(B), SBC(B),	5	6.10	4	10
SXT, MFPS,	6	6.10	4	10
XOR	7	7.32	5	12
	0	2.44	1	4
	1	3.66	2	6
	2	3.66	2	6
TST(B)	3	5.49	3	8
	4	4.27	2	7
	5	5.49	3	9
	6	5.49	3	9
	7	6.71	4	11
	0	4.88	1	8
	1	6.10	2	10
	2	6.10	2	10
MTPS	3	7.32	3	12
	4	6.71	2	11
	5	7.93	3	13
	6	7.93	3	13
	7	9.16	4	15

Table A-1 Instruction Timing (Cont)

Double Operand Instructions	Source Mode	Source Mode Time (μs) Includes Fetch	Number of Bus Transactions	Number of Microcycles
MOV(B), CMP(B),	0	1.83	1	3
ADD, SUB,	1	3.05	2	5
BIT(B), BIC(B),	2	3.05	2	5
BIS(B)	3	4.27	3	7
	4	3.66	2	6
	5	4.88	3	8
	6	4.88	3	8
	7	6.10	4	10
Double Operand Instructions	**Destination Mode**	**Destination Mode Time (μs)**	**Number of Bus Transactions**	**Number of Microcycles**
MOV(B), CMP(B),	0	0.61	0	1
ADD, SUB,	1	2.44	2	4
BIT(B), BIC(B),	2	2.44	2	4
BIS(B)	3	3.66	3	6
	4	3.05	2	5
	5	4.27	3	7
	6	4.27	3	7
	7	5.49	4	9
	0	0.61	0	1
	1	1.83	1	3
	2	1.83	1	3
CMP(B), BIT(B)	3	3.05	2	5
	4	2.44	1	4
	5	3.66	2	6
	6	3.66	2	6
	7	4.88	3	8
Jump and Subroutine Instructions	**Destination Mode**	**Fetch and Execute Time (μs)**	**Number of Bus Transactions**	**Number of Microcycles**
	1	3.05	2	5
	2	3.66	2	6
JMP	3	3.66	3	6
	4	3.66	2	6
	5	4.27	3	7
	6	4.27	3	7
	7	5.49	4	9

Table A-1 Instruction Timing (Cont)

Single Operand Instructions	Destination Mode	Fetch and Execute Time (μs)	Number of Bus Transactions	Number of Microcycles
JSR	1	5.49	4	9
	2	6.10	4	10
	3	6.10	5	10
	4	6.10	4	10
	5	6.71	5	11
	6	6.71	5	11
	7	7.90	6	13
RTS	NA	4.27	2	7
SOB	NA	3.66	1	6

Branch, Trap, and Interrupt Instructions	Destination Mode	Fetch and Execute Time (μs)	Number of Bus Transactions	Number of Microcycles
BR, BNE, BEQ, BPL, BMI, BVC, BVS, BCC, BCS, BGE, BLT, BGT, BLE, BHI, BLOS, BHIS, BLO	NA	2.44	1	4
EMT, TRAP, BPT, IOT	NA	9.77	7	16
RTI	NA	4.88	3	8
RTT	NA	6.71	3	11

Miscellaneous and Condition Code Instructions	Destination Mode	Fetch and Execute Time (μs)	Number of Bus Transactions	Number of Microcycles
HALT	NA	8.54	5	14
WAIT	NA	2.44	1	4 then loop
RESET	NA	22.28	1	39

Table A-1 Instruction Timing (Cont)

Single Operand Instructions	Destination Mode	Fetch and Execute Time (μs)	Number of Bus Transactions	Number of Microcycles
NOP	NA	3.66	1	6
CLC, CLV, CLZ, CLN, CCC, SEC, SEV, SEZ, SEN, SCC	NA	3.66	1	6
MFPT	NA	3.05	1	5

The measure of LSI-11 bus interrupt latency is the time from the assertion of BIRQ until BIAKI is accepted by the interrupting device electrically closest to the processor on the LSI-11 bus.

The measure of local interrupt latency is the time from assertion of the request until the time the microprocessor is ready to fetch the first instruction in the interrupt service routine. This time is primarily comprised of the time to perform two pushes and a PC and PSW restore.

Interrupt Latency:	LOCAL	23.2 μs
	LSI-11 BUS	9.3 μs

NOTE

Assume that the stack and vector memory reside on the SBC-11/21 PLUS and that the LSI-11 bus device can assert BRPLY and vector within 600 ns after receiving IAKI. The service latency (time from BIRQ until the time the microprocessor is ready to fetch the first instruction in the interrupt service routine) depends on the response time of the interrupting device (i.e., RDIN to TRPLY and negation of TRPLY).

DMA latency is the period of time between a device asserting its BDMR and receiving BDMGI when it resides on the LSI-11 bus as the electrically closest DMA device to the processor.

DMA latency: 1.3 μs (minimum) 11.0 μs (maximum)

WAIT instruction latencies:

Internal vector:	11.8 μs
External vector:	12.4 μs
DMA:	5.06 μs

APPENDIX B
PROGRAMMING DIFFERENCE LIST

DIFFERENCES BETWEEN THE SBC-11/21 PLUS, LSI-11/2, AND LSI-11/23

Table B-1 presents a concise comparison of the SBC-11/21 PLUS, LSI-11/2, and LSI-11/23 modules.

Table B-1 SBC-11/21 PLUS, LSI-11/2, and LSI-11/23 Comparisons

Activity	SBC-11/21 PLUS	LSI-11/2	LSI-11/23
OPR %R,(R)+ or OPR %R,–(R) using the same register as both source and destination: contents of 'R' are incremented (decremented) by two before being used as the source operand.	X		X
OPR %R,@(R)+ or OPR %R,@–(R) using the same register as both source and destination: contents of 'R' are incremented (decremented) by two before being used as the source operand.	X		X
In the previous two cases, initial contents of 'R' are used as the source operand.		X	
OPR PC,X(R); OPR PC,@X(R); OPR PC,@A; or OPR PC,A: location A will contain the PC of OPR + 4.	X		X
In the previous case, location A will contain the PC of OPR + 2.		X	
JMP (R)+ or JSR reg,(R)+: initial contents of 'R' are used as the new PC.	X	X	X
JMP %R or JSR reg,%R traps to 4 (illegal instruction).	X	X	X

Table B-1 SBC-11/21 PLUS, LSI-11/2, and LSI-11/23 Comparisons (Cont)

Activity	SBC-11/21 PLUS	LSI-11/2	LSI-11/23
Only one LSI-11 bus interrupt level (BR4) exists.	X	X	
Four local interrupt levels exist.	X		
Four LSI-11 interrupt levels exist.			X
Stack overflow not implemented.	X	X	
A stack overflow trap exists.			X
The first instruction in an interrupt routine will not be executed if another interrupt occurs at a higher priority level than assumed by the first interrupt.	X	X	X
Eight general-purpose registers.	X	X	X
PSW address 177776 not implemented. Must use MTPS and MFPS instructions.	X	X	
Only implicit references (RTI, RTT, traps, and interrupts) can load T-bit. Console cannot load T-bit.	X	X	X
If an interrupt occurs during an instruction that has the T-bit set, the T-bit trap is acknowledged before the interrupt.	X	X	X
If RTI sets the T-bit, T-bit trap is acknowledged immediately following RTI.	X	X	X
T-bit trap will sequence out of WAIT instruction.	X		X
If RTT sets the T-bit, the T-bit trap occurs after the instruction following RTT.	X	X	X
RESET instruction consists of 10 μs of INIT followed by a 90 μs pause. Power fail is not recognized until the instruction is complete.		X	X

Table B-1 SBC-11/21 PLUS, LSI-11/2, and LSI-11/23 Comparisons (Cont)

Activity	SBC-11/21 PLUS	LSI-11/2	LSI-11/23
RESET instruction consists of 17 μs of INIT followed by a minimum 3.2 μs pause. Power fail is not recognized until the instruction is complete.	X		
Odd address references using the SP do not trap.	X		
Nonexistent address references using the SP trap to the restart address.	X		
MOVB instruction does a read (DATI) and a write (DATO) bus sequence for last memory cycle.		X	
MOV instruction does a write (DATO) bus sequence for the last memory cycle.		X	X
MOV instruction does a read (DATI) and a write (DATO) bus sequence for last memory cycle.	X		
CLR(B) and SXT do a read (DATI) and a write (DATO) sequence for the last bus cycle.	X		
CLR(B) and SXT do a read (DATI) and a write (DATO) bus sequence for the last bus cycle.		X	
CLR(B) and SXT do a write (DATO) bus sequence for the last bus cycle.			X
MARK instruction.		X	X
SOB, RTT, SXT, XOR instructions.	X	X	X
SWAB clears V.	X	X	X
ASH, ASHC, DIV, MUL instructions.		X	X

Table B-1 SBC-11/21 PLUS, LSI-11/2, and LSI-11/23 Comparisons (Cont)

Activity	SBC-11/21 PLUS	LSI-11/2	LSI-11/23
Register addresses (177700–177717) are handled as regular memory addresses. No internal registers are addressable from either the bus or the console.	X		
Register addresses (177000–177717) time-out when used as program addresses by the CPU.		X	X
If PC contains a nonexistent memory address and a bus error occurs, PC will have been incremented.	X	X	X
If register contains a nonexistent memory address in mode 2 and a bus error occurs, register will be incremented.	X	X	X
If register contains an odd value in mode 2 and a bus error occurs, register will be incremented.	X	X	X
HALT in user mode traps to 10.			X
HALT instruction pushes PC and PSW on the stack and loads the PSW with 340 and the PC with the restart address.	X		
Only power-up mode 2 implemented.	X		
Resident ODT microcode.		X	X
Instruction execution runs to completion regardless of bus error.	X		
BEVNT line interrupt on level 6.	X		X
Bus error traps to restart address. Instruction runs to completion before trap.	X		

Table B-1 SBC-11/21 PLUS, LSI-11/2, and LSI-11/23 Comparisons (Cont)

Activity	SBC-11/21 PLUS	LSI-11/2	LSI-11/23
Bus error during IAK vectors through 0 and traps to restart address. The first instruction of service routine is guaranteed to execute.	X		
Only 16-bit addressing supported.	X	X	
The no-BSACK 18 μs time-out implemented. If time-out occurs BDMGO aborted.			X
Bus halt line is a jumper configured nonmaskable interrupt. Acknowledgement causes PC and PSW to be stacked and the processor vectors through level 7 internal vector 140.	X		
Vector address accepted only on BDAL<7:2>. This limits vector address space to 374.	X		
Certain vector addresses are reserved for local devices other than BEVNT.	X		

Table B-2 Illegal Address Traps

From	Through	Response	11/21 PLUS	LSI 11/2	11/23	
210	217	Trap to 10	X	*	X	Reserved instruction
210	227	Trap to 10	X	X	X	Reserved instruction
70000	73777	Trap to 10	X	**	**	Extended instruction set
75000	75037	Trap to 10	X	X	**	Floating point
75040	75777	Trap to 10	X	**	X	Reserved instruction
170000	177777	Trap to 10	X	**	**	Reserved instruction

*Maintenance instructions
**Response depends on processor options

SBC-11/21 PLUS Priorities
Priority of DMA, system traps, external interrupts, internal interrupts, HALT trap, and WAIT:

DMA (highest priority)
HALT trap (time-out request)
Power fail trap
Traps (illegal instruction, T-bit, EMT)
Internal interrupt request
External interrupt request
WAIT instruction (lowest priority)

APPENDIX C
SOFTWARE DEVELOPMENT

C.1 GENERAL

This appendix describes programming notes that may help application programmers to gain familiarity with the SBC-11/21 PLUS. The following five topics are discussed:

1. Running RT-11 V1.5 operating system
2. Running MicroPower/Pascal V1.5
3. Running standalone programs
4. The software development process
5. An application example

A method of creating, loading, and running standalone programs is explained. This is followed by a discussion of the software development process as it applies to a ROM based single-board computer. The last section of this appendix presents a practical example of a real-time program written to run on the SBC-11/21 PLUS. The output selected for the program is deliberately simple, however, the methodology is applicable to more complex programs. The program has been tested, and studying it should be informative to first time users of the SBC-11/21 PLUS.

C.2 RUNNING RT-11 V5.1 OPERATING SYSTEM

SBC-11/21 PLUS supports both the single job (SJ) as well as the foreground background (FB) operating system (see Chapter 3; Supported Software Options). An application may be developed on the SBC-11/21 PLUS using any language supported by RT-11. See RT-11 documentation for additional information concerning RT-11 developed applications.

C.3 RUNNING MICROPOWER/PASCAL

SBC-11/21 PLUS supports MicroPower/Pascal V1.5 or subsequent versions. An application is developed on a host system running either RT-11 or RSX-11M and then loaded into the SBC-11/21 PLUS for execution.

C.4 RUNNING STANDALONE PROGRAMS FROM TU58 OR RX01/02

The user can develop standalone programs, programs not needing an operating system, on a separate RT-11 based system. The .SAV image can then be loaded into the SBC-11/21 PLUS and run from either TU58 or RX01/02 storage devices. The Macro-ODT option is needed to load the program and to run it.

If the standalone program is to be used with Macro-ODT, it must have the address of Macro-ODT BREAK service routine in location 140 and a PSW value of 300 in location 142. This will enable the program to transfer control to Macro-ODT when the BREAK key is pressed.

To load the standalone program from the mass storage device into the SBC-11/21 PLUS, the device's boot block must be modified. This change extends to locations 0, 2, 4, and 6. Location 0, which normally contains 240, must be changed to 260. When the device is booted, this tells the Macro-ODT that the mass storage device contains a standalone program. Macro-ODT will then interpret the contents of locations 2, 4, and 6 as a RADIX-50 encoded six-character file name and search the directory of the volume for that file. The volume must have the RT-11 file structure. When the file is found, the complete file is loaded into contiguous memory starting at location 0. Then Macro-ODT loads register R0 with the number of the unit or drive and register R1 with the CSR address of the booted device.

The stack pointer (SP) is loaded with the contents of location 42, the program counter (PC) is loaded with the contents of location 40, and the program starts execution. A standalone program developed on an RT-11 based system will have had the correct values for PC and SP in locations 40 and 42. This information may be of use to the standalone program if it uses overlays.

The detailed procedure for performing these modifications in the boot block and the standalone program follows, and will be done on an RT-11 based system using the SIPP utility.

In the following examples, the program that is to be loaded and run from the standalone volume is named FOOBAR.SAV and resides on DK. The characters entered by the operator are underlined. '<CR>' is a carriage return and not the four characters '<', 'C', 'R', and '>'. The ' C' and ' Y' symbols are obtained by holding down the 'CTRL' key and typing 'C' or 'Y' before releasing 'CTRL'. 'XXXXXX' is a string of octal digits whose value can be anything but does nothing to the process.

First, modify the standalone program:

```
. R SIPP <CR>                                  ;Run the SIPP utility

* DK:FOOBAR.SAV <CR>                           ;Name of file to be
                                                patched

Base? <CR>                                     ;Defaults to zero

Offset? 140 <CR>

Base     Offset    Old      New?

000000   000140    xxxxxx   170000 <CR>        ;Load address of BREAK
                                                routine at BREAK vector

000000   000142    xxxxxx   300 <CR>           ;PSW during BREAK routine

000000   000144    xxxxxx   ^Y <CR>            ;Exit patching

*^C                                            ;Exit SIPP
```

NOTE

If you are using your own BREAK intercepting routine, put its address at location 140 in place of the value 170000.

Now modify the boot block:

.R SIPP <CR>

*DK:/A <CR>

Base? <CR>

Offset? <CR>

Base	Offset	Old	New?
000000	000000	xxxxxx	000260
000000	000002	xxxxxx	;RFOO <CR>
000000	000004	xxxxxx	;RBAR <CR>
000000	000006	xxxxxx	;RSAV <CR>
000000	000010	xxxxxx	∧Y <CR>

*∧C

C.5 THE SOFTWARE DEVELOPMENT PROCESS

Software development for the SBC-11/21 PLUS can be considered as four discrete steps. These steps are illustrated in Figure C-1.

1. Design the software and code the source tasks.
2. Enter, edit, and assemble the tasks that make up the application.
3. Build the application into a runnable memory image.
4. Load the program into the SBC-11/21 PLUS and execute the application program. This step includes the debugging of the application.

C.5.1 Design of the Software

An important consideration in the design of application software is the run-time memory configuration. Because the SBC-11/21 PLUS is a ROM/RAM system, the location of the ROM/RAM boundaries must be defined. All instructions and constants must be arranged separately for location in the ROM section of memory. Variable information must be arranged together for location in the RAM section of memory. During the development process, the separation of ROM and RAM information must be maintained. See the *MACRO-11 Language Reference Manual* for a description of the methods of data and code separation.

C.5.2 Editing and Assembly

The second step in the development cycle is the entry, editing, and assembly of the application software. Entering and creating the application software includes the use of an editor on the development system. Once the application software is entered and the designer is satisfied with the contents, it can be saved on a mass storage device. The assembler must then be used to convert the source code instructions into executable code. The result of the assembly process is an object file.

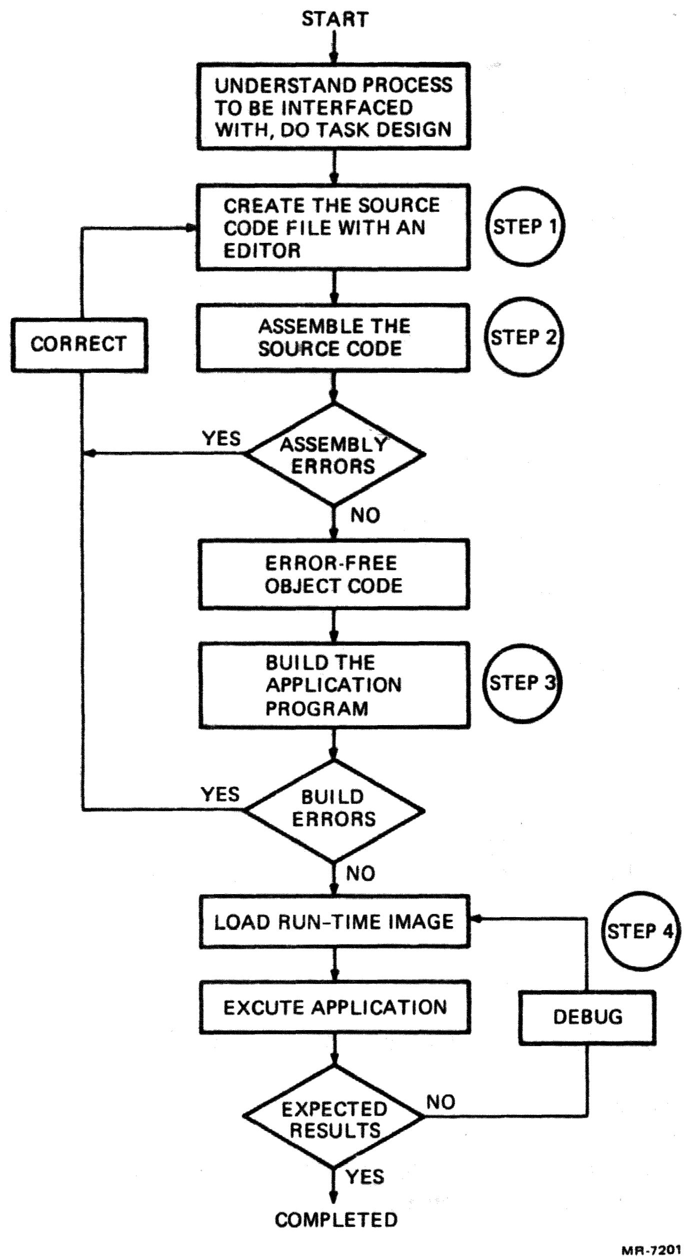


Figure C-1 Overview of Software Development

The assembler detects common assembly language coding errors and issues appropriate warnings. If errors are detected, corrections should be made by re-editing the source and reassembling. Once the application software has been translated error free into object form, it is ready for the next step.

C.5.3 Building Process

The third step in the development cycle is the building process to create a runnable memory image. The build process uses the linking of the tasks that make up the application software into a single memory image. The building process takes an object module or modules and assigns absolute memory references to the information contained in the object code. The user assigns these locations by the sectioning of the code that took place during design. The result of the build phase is an executable run-time memory image that can be loaded and tested.

C.5.4 Running and Debugging the Program

The fourth step in the development cycle is the loading of the runnable memory image into the SBC-11/21 PLUS. Once loaded, the program can be run and debugged. There are three methods that can be used to transfer the software to the target.

1. ROM transfer. This method uses the programming of ROMS via a PROM blasting utility, such as PB-11, and places the PROMs into the target configuration. This simple loading method resembles the final target configuration because actual ROM storage is used.

2. Media transfer. When this method is used, the application program is loaded, in standalone form, into the target from a mass storage system. The directions on creating a standalone bootable program are provided in Paragraph C-2. The target configuration uses LSI-11 bus RAM memory in place of the SBC-11/21 PLUS on-board ROM during initial startup and debug. The SBC-11/21 PLUS configuration must contain the Macro-ODT ROMs described in Chapter 4. The ODT ROMs provide the means of loading the application program and are used during program debug. Media transfer does not reflect the final configuration, but execution from RAM makes debugging and testing easier. The speed of the program in this mode is approximately half that of the ROM based system.

3. Down-line loading. This method of loading allows transfer of the controller software from the development system to the target system via a serial communication link. The down-line loader must be a development system utility. The target configuration is similar to the media transfer configuration. In addition to the LSI-11 bus, RAM, and the Macro-ODT ROMs, one of the serial I/O lines on the SBC-11/21 PLUS must be dedicated to the communication with the development system.

When the correct loading method is implemented, the final phase of development is to debug and run. The loading method used defines the solution that will be taken during debug.

If the application is being loaded via the ROM transfer method, initial testing and debugging is difficult. When ROM transfer is used, there must be embedded code in the application that will report the state of the control system regularly. Another way to check the system is to note changes that occur in the external devices. If errors are found, a complete reprogramming of the PROMs is necessary. This type of testing and debugging can be difficult.

When the application is loaded via media transfer, the testing and debugging becomes easier than the ROM method. Once the application program is loaded into LSI-11 bus RAM or into on-board RAM, it can be run using the features of Macro-ODT. The designer can also include reporting tasks and halts in the application to examine the current state of the system. Executing out of LSI-11 bus RAM during debug is approximately twice as slow as running out of the SBC-11/21 PLUS on-board memory. If errors are found, minor changes can be made in the application code because testing is being done in RAM. This deletes the loop of making new run-time memory images for every change. Once the target system is running successfully with all of the tasks integrated, the run-time configuration can be set up. The last step is to load the application program to ROM and run in the SBC-11/21 PLUS.

C.6 AN APPLICATION EXAMPLE

A sample application is illustrated in Figure C-2 and shows the development of a controller program using MACRO-11. The sample program will only light the LED used by port C of the SBC-11/21 PLUS. The LED will light for 10 s when an input is detected on the console port (SLU1).

The controller program for this simple system is best operated by using an interrupt driven environment. An interrupt service routine is used to monitor the console port. When an input is received, a routine is entered that will set the timer for 10 s and light the LED. A second interrupt service routine is used to count up to 10 s and then turn off the LED. This routine is serviced by the BEVNT interrupts. In addition

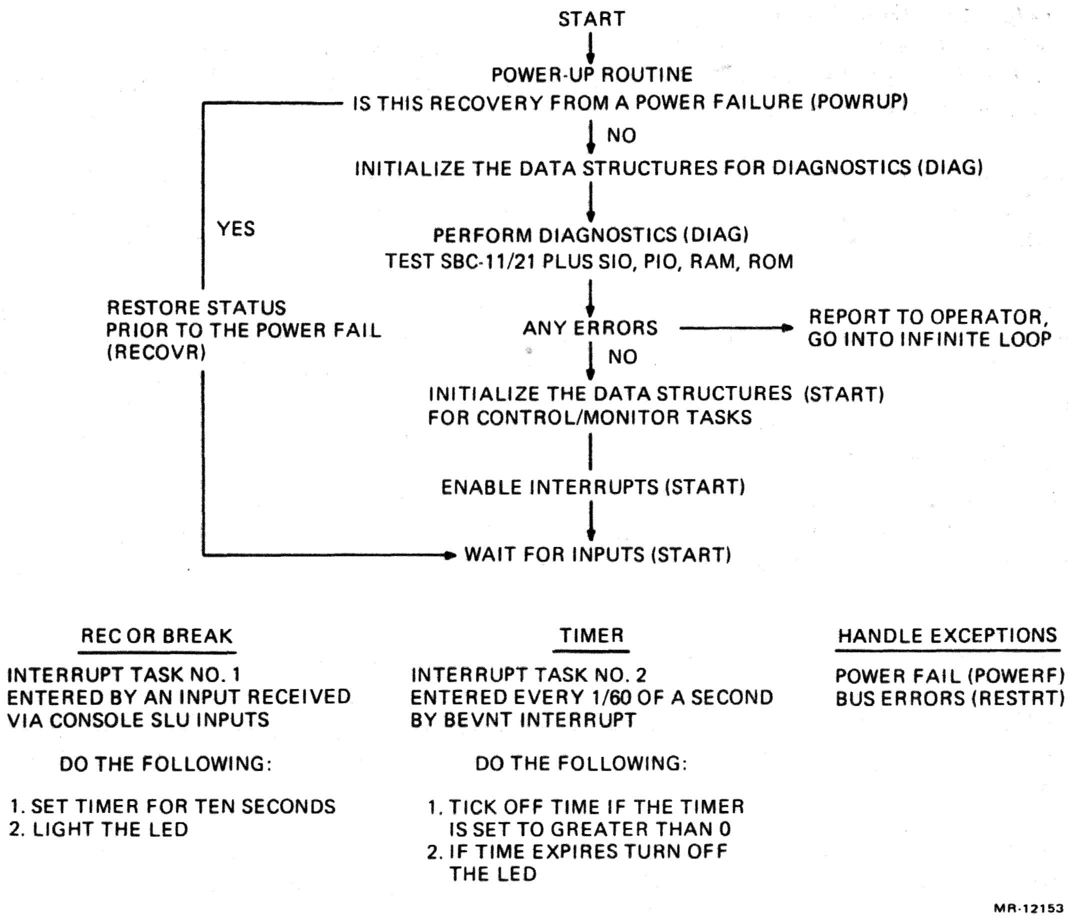


Figure C-2 Application Overview

to the application tasks, there are tasks to initialize the input/output devices and data structures. There are also diagnostic programs for the SBC-11/21 PLUS and programs used to handle any exceptions. The controller program is developed as individual tasks and then integrated into a complete final program.

The monitor program, shown in Figure C-3, consists of power-up programs, diagnostic programs, task programs, and exception programs. The power-up programs consist of POWRUP and RECOVR, which is started by POWRUP. The diagnostic programs consist of SLUTST, PIOTST, RAMTST, and ROMTST. The task programs consist of TIMER, REC, and BREAK. The exception programs consist of POWERF, RESTRT, and PRINT. All these programs with the constants and instruction data are stored in the ROM memory. The variable data for the application and the stack are stored in the RAM memory. Memory map 1 is assumed (Table 2-8), with the program in ROM socket set B, and data in battery backed RAM starting at 160000. The load map in Figure C-4 shows actual memory locations assigned to code and data.

C.6.1 Power-Up Programs

The controller program starts when the system power is applied. The microprocessor accesses location 0, which is the jumper configured start address. This location contains a jump to the power-up routine POWRUP (see Figure C-5). This routine determines if this is a normal power-up or a recovery from a power failure. This is determined by checking the power fail flag in the RAM memory. If the flag is set to indicate that the system is recovering from a power fail condition, the program jumps to the RECOVR program (see Figure C-6). This program restores the system to the conditions that existed before the power fail and continues program execution. If the flag is not set, an initial power-up program is executed and the program then branches to the diagnostic programs.

C.6.2 Diagnostic Programs
The diagnostic programs are entered via a diagnostic initialization routine. The SLUTST program (see Figure C-7) is the first diagnostic, and it tests the auxiliary serial line unit on the SBC-11/21 PLUS. The diagnostic enables the SLU maintenance mode and transmits many test patterns. After a certain amount of time, the program checks to see that the test patterns were correctly received. The SLU maintenance mode allows data to be transmitted to the EIA port as well as through the internal loopback. Therefore, if a device is connected to the port, it will respond to this data.

The second diagnostic, RAMTST (see Figure C-8), tests the RAM memory. The test is performed by writing known data into a RAM location and checking that the correct information is in that location.

The third diagnostic, ROMTST (see Figure C-9), checks the ROM memory. This test calculates a checksum on the actual control and monitoring tasks. If there is a checksum error, there is a potential failure at some ROM location.

The last diagnostic, PIOTST (see Figure C-10), checks the parallel I/O port on the SBC-11/21 PLUS. This test verifies that the parallel I/O registers can be addressed. The send/receive capability cannot be checked unless there is a loopback connector installed on the J3 connector. When data is written into these registers and a device is connected to the port, the device can respond to the data.

When any of the above diagnostics detect a failure, the program will set an error flag. The diagnostic program will check the status of all error flags before it enters the task programs. If an error is found, the operator is informed that a diagnostic test failed, and the program enters a loop to wait for the operator to interrupt. Each diagnostic will print a message to the operator indicating success or failure. If there are no failures, a success message is printed and the program enters the task programs.

C.6.3 Control Task Programs
The control task programs (see Figure C-11) complete the initialization of the system by clearing the receive buffer, enabling the interrupts, and lowering the microprocessor priority to accept interrupts. The operator is then informed that the system is running and waiting for interrupts. The TIMER receives a BEVNT input sixty times per second. The REC program is entered when an interrupt is received from the console. The program will then turn on the LED and load the 10 s counter. The BREAK program is entered when a BREAK is detected and performs the same task as REC. A TIMER program will decrement the 10 s counter, if it is enabled, every time BEVNT is received. When the 10 s counter is decremented to zero, the program will turn off the LED. If the LED is turned on and another BREAK or interrupt occurs, the 10 s counter is reset for 10 s. The program also allows any exception conditions.

C.6.4 Exception Programs
The system is now running and the exception programs are entered only when a power fail occurs or a bus time-out occurs. The print program is entered only to communicate with the operator.

A time-out will occur when an address does not respond or if a device does not respond to an interrupt acknowledge. When a time-out occurs, the SBC-11/21 PLUS will trap to location 4, the restart address. The start address is defined as location 0, and restart address is defined as location 4 by the factory configuration. The RSTRT program is entered via location 4; it informs the operator that a run-time error has occurred and waits for the operator to interrupt.

A power failure is detected when the system power is going down. This enables the power fail interrupt and causes a trap to location 24. The POWERF program (see Figure C-12) is entered via location 24, and the power fail flags are set in the RAM memory. The RAM memory includes the battery backup feature of the SBC-11/21 PLUS module. Program information contained in the general-purpose registers, the stack pointer, and other necessary data are stored in the nonvolatile RAM memory. The program then puts the bus into a known state with the RESET instruction and waits for the power loss to occur. When power is restored, the POWRUP routine is executed and data is recovered as the system restarts.

```
FALCON DEVELOPMENT EXAMPLE      MACRO V04.00  8-FEB-82 04:20:36 PAGE 1

                                .TITLE FALCON DEVELOPMENT EXAMPLE
                                .ENABL  LC
                                .GLOBL  POWERF,BREAK,REC,TIMER,RECOVR,SLUTST,PIOTST,RAMTST,ROMTST
                                .GLOBL  POWER1,POWER2,ERROR,STACK,RESTRT
                        ;+
                        ; This is an example of a simple controller application for the KXT11-AA.
                        ;-


                                .SBTTL  Program section definitions
                        ;+
                        ;  Define the three program sections that will be used
                        ;-
 000000                         .ASECT                          ; Assign absolute memory locations
 000000                         .PSECT  ROM                     ; For insructions and constant data
                                                                ; that will be stored in rom memory
 000000                         .PSECT  RAM,D                   ; To define all RAM locations


                                .SBTTL  Equates
                        ;+
                        ; Constant definitions
                        ;-
          176540                RCSR1   == 176540               ; Auxiliary SLU addresses

          177560                RCSRC   == 177560               ; Console  SLU addresses
          000052                CONBR   == 52                   ; Programmable baud rate mask (9600)

          176200                PPA     == 176200               ; Parallel port A
          176206                PCW     == 176206               ; Parallel control word
          000261                LEDON   == 261                  ; Parallel CSW to light the LED
          000017                LEDOFF  == 17                   ; Parallel CSW to turn off the LED

          160010                RAMBGN  == 160010               ; Bottom of the user RAM
          167776                RAMTOP  == 167776               ; Top of the user RAM

          106016                CSUM    == 106016               ; Checksum value for the system tasks


                                .SBTTL  Macro definitions
                        ;+
                        ; Define macros that will be used by the application
                        ;-
                                .MACRO  PUSH ARG        ; stack push operation
                                 MOV    ARG,-(SP)               ; move the argument onto the stack
                                .ENDM

                                .MACRO  POP ARG         ; stack pop operation
                                 MOV    (SP)+,ARG               ; move the argument from the stack
                                .ENDM


                                .SBTTL  Entry points
                        ;+
                        ;  Define entry point, interrupt, and trap service routine addresses
                        ;-
 000000                         .ASECT
```

Figure C-3   Monitor Program

```
              000000                            .=0
       000000 000167  000000'                           JMP     POWRUP                  ; Jump to the power-up routine
       000004 000167  000000G                           JMP     RESTRT                  ; Jump to the restart routine
              000024                            .=24
       000024 000000G 000340                            .WORD   POWERF,340              ; Power fail service routine

              000060                            .=60
       000060 000000G 000300                            .WORD   REC,300                 ; Console receiver service routine

              000100                            .=100
       000100 000000G 000300                            .WORD   TIMER,300               ; Timer service routine

              000140                            .=140
       000140 000000G 000300                            .WORD   BREAK,300               ; Console break service routine


                                                        .SBTTL  Power up routine
       000000                                           .PSECT  ROM

       000000                                   POWRUP::
                                                ;+
                                                ; Come here first under all circumstances and decide if this is a normal
                                                ; power up or recovery from a power fail
                                                ;-
       000000 026727  000000G 123456                    CMP     POWER1,#123456          ; Is this recovery from  power failure
       000006 001006                                    BNE     DIAG                    ; or is it a normal power-up ?
       000010 026727  000000G 135724                    CMP     POWER2,#135724          ; Increase the chance to distinguish
       000016 001002                                    BNE     DIAG                    ; by checking against a 32 bit pattern

       000020 000167  000000G                           JMP     RECOVR                  ; This a recovery from power fail


                                                        .SBTTL Diagnostics

       000024                                   DIAG::
                                                ;+
                                                ; Do the system diagnostics
                                                ;-
       000024 012706  000000G                           MOV     #STACK,SP               ; Initialize the stack
       000030 005067  000000G                           CLR     ERROR                   ; Initialize the error flag
       000034 052737  000052  177564                    BIS     #CONBR,@#RCSRC+4        ; Initialize the console SLU

       000042 004767  000074                            CALL    PRINT                   ; Tell the operator that the power-
       000046 000174'                                   .WORD   DIAGM                   ; up diagnostics are running
       000050 004767  000000G                           CALL    RAMTST                  ; Perform the KXT11 RAM memory test
       000054 004767  000000G                           CALL    ROMTST                  ; Perform the KXT11 ROM memory test
       000060 004767  000000G                           CALL    SLUTST                  ; Perform the KXT11 serial line test
       000064 004767  000000G                           CALL    PIOTST                  ; Perform the KXT11 parallel I/O test

       000070 005767  000000G                           TST     ERROR                   ; Is the error flag zero
       000074 001404                                    BEQ     1$                      ; Yes, no errors proceed to init
       000076 004767  000040                            CALL    PRINT                   ; No, diagnostic failure
       000102 000332'                                   .WORD   EMESS
       000104 000777                                    BR      .                       ; Wait until there is operator action
       000106 004767  000030                    1$:     CALL    PRINT                   ; Indicate that things are OK
       000112 000253'                                   .WORD   HMESS                   ; and move on

```

Figure C-3   Monitor Program (Cont)

```
FALCON DEVELOPMENT EXAMPLE      MACRO V04.00  8-FEB-82 04:20:36 PAGE 1-2
DIAGNOSTICS

                                                 .SBTTL  Initialization completion, allow application tasks to run

 000114                                  START::
                                         ;+
                                         ; This is the start of the main body of the application
                                         ;-
 000114  105737  177562                          TSTB    @#RCSRC+2                ; Flush the receiver buffer
 000120  052737  000100  177560                  BIS     #100,@#RCSRC             ; Enable interrupt on the receiver
 000126  106427  000000                          MTPS    #0                       ; Allow interrupts to happen
 000132  004767  000004                          CALL    PRINT                    ; Tell the operator that the
 000136  001015'                                 .WORD   GO                       ; application is up and running

 000140  000777                                  BR                               ; Sit and wait for interrupts


 000142                                  PRINT::
                                         ;+
                                         ; This subroutine prints the actual messages
                                         ;-
                                                 .ENABL  LSB
 000142  017604  000000                          MOV     @(SP),R4         ; Point to the beginning of the message
 000146  005216                                  INC     (SP)             ; Increment beyond message address in the
 000150  005216                                  INC     (SP)             ; calling routine
 000152  112405                          1$:     MOVB    (R4)+,R5         ; Move the next character to be printed
 000154  001406                                  BEQ     3$               ; Is this the end of message marker ?
 000156  105737  177564                  2$:     TSTB    @#RCSRC+4        ; No, output another character
 000162  100375                                  BPL     2$               ; Transmitter ready
 000164  110537  177566                          MOVB    R5,@#RCSRC+6     ; Output the character
 000170  000770                                  BR      1$               ; Get another character
 000172  000207                          3$:     RETURN                   ; Go back
                                                 .DSABL  LSB


                                                 .SBTTL  Messages sent to the operator
                                                 .NLIST  BEX

 000174     015     012     040          DIAGM:: .ASCIZ  <15><12>/ The power-up diagnostics are running ... /<15><12>
 000253     015     012     040          HMESS:: .ASCIZ  <15><12>/ System checked out, there were no faults /<15><12>
 000332     015     012     040          EMESS:: .ASCIZ  <15><12>/ System did not pass initial power up test /<15><12>
 000412     015     012     007          FMESS:: .ASCIZ  <15><12><7><7><7><7>/ RUN-TIME FAILURE /<15><12>
 000445     015     012     040          SLUE::  .ASCIZ  <15><12>/ Serial line unit diagnostic failure /<15><12>
 000517     015     012     040          SLGOOD::.ASCIZ  <15><12>/ Serial line unit passed diagnostics /<15><12>
 000571     015     012     040          MESRA1::.ASCIZ  <15><12>/ RAM failure /<15><12>
 000613     015     012     040          RAGOOD::.ASCIZ  <15><12>/ RAM passed diagnostics /<15><12>
 000650     015     012     040          MESRO1::.ASCIZ  <15><12>/ ROM checksum error /<15><12>
 000701     015     012     040          ROGOOD::.ASCIZ  <15><12>/ ROM passed diagnostics /<15><12>
 000736     015     012     040          PGOOD:: .ASCIZ  <15><12>? Parallel input/output passed diagnostics ?<15><12>
 001015     015     012     040          GO::    .ASCII  <15><12>/ The application is running ... / <15><12>
 001061     040     040     040                  .ASCIZ  /        Type any key to light the KXT11-AA LED for 10 secs./
                                                 .EVEN


         000001                                  .END
```

Figure C-3   Monitor Program (Cont)

```
RT-11 LINK  V06.01C     Load Map          Mon 08-Feb-82 04:21:23
C      .SAV      Title:  FALCON  Ident:            /B:000400

Section  Addr    Size    Global  Value    Global  Value    Global  Value

 . ABS.  000000 000400    (RW,I,GBL,ABS,OVR)
                         LEDOFF  000017   CONBR   000052   LEDON   000261
                         CSUM    106016   RAMBGN  160010   RAMTOP  167776
                         PPA     176200   PCW     176206   RCSR1   176540
                         RCSRC   177560
 ROM     000400 157400    (RW,I,LCL,REL,CON)
                         POWRUP  000400   DIAG    000424   START   000514
                         PRINT   000542   DIAGM   000574   HMESS   000653
                         EMESS   000732   FMESS   001012   SLUE    001045
                         SLGOOD  001117   MESRA1  001171   RAGOOD  001213
                         MESRO1  001250   ROGOOD  001301   PGOOD   001336
                         GO      001415   RECOVR  001556   REC     001634
                         TIMER   001656   BREAK   001702   LAST    001716
                         POWERF  001720   RESTRT  001764   SLUTST  001774
                         RAMTST  002132   ROMTST  002212   PIOTST  002262
 RAM     160000 000332    (RW,D,LCL,REL,CON)
                         POWER1  160010   POWER2  160012   SAVER6  160014
                         ERROR   160016   TIME    160020   STACK   160332

Transfer address = 000001, High limit = 160330 = 28780. words
```

Figure C-4 Load Map

```
.MAIN.  MACRO V04.00  8-FEB-82 04:21:16 PAGE 1

                                 .ENABL  LC
  000000                         .PSECT  RAM,D
                         ;
                         ; The variable data is assigned to the user RAM space on the KXT11-AA
                         ;

  000000                         .BLKW   4                 ; Non existant KXT11-AA memory
  000010   000000        POWER1::.WORD   0                 ; Power failure 32-bit comparision
  000012   000000        POWER2::.WORD   0                 ; flag
  000014   000000        SAVER6::.WORD   0                 ; Stack pointer area for power failure
  000016   000000        ERROR:: .WORD   0                 ; Diagnostic error flag

  000020   000000        TIME::  .WORD   0                 ; Time flag
  000022                         .BLKW   100.              ; This is the stack
  000332                 STACK::

           000001                .END
```

Figure C-5 Power-up Task

```
.MAIN.  MACRO V04.00  8-FEB-82 04:21:11 PAGE 1

                                          .ENABL  LC
                                          .GLOBL  SAVER6,TIME,RCSRC,CONBR,LEDON,PCW
                                          .MCALL  POP
 000000                                   .PSECT  ROM

 000000                           RECOVR::
                                  ;+
                                  ;This routine is entered if a recovery from a power failure is taking place
                                  ;-
 000000  016706  000000G                  MOV     SAVER6,SP              ; Restore the stack pointer
 000004                                   POP     TIME                   ; Restore the any variable information
 000010                                   POP     R5                     ; Restore the general purpose
 000012                                   POP     R4                     ; registers
 000014                                   POP     R3
 000016                                   POP     R2
 000020                                   POP     R1
 000022                                   POP     R0
 000024  052737  000100  000000G          BIS     #100,@#RCSRC           ; Re-initialize console SLU, enable
 000032  052737  000000G 000004G          BIS     #CONBR,@#RCSRC+4       ; interrupts and set-up baud rate
 000040  005767  000000G                  TST     TIME                   ; Is the LED timer set
 000044  001403                           BEQ     3$                     ; No, continue
 000046  012737  000000G 000000G          MOV     #LEDON,@#PCW           ; Yes turn the LED on for the rest
                                                                         ; of the time prior to power-fail
 000054  000002                   3$:     RTI                            ; Return from point of power-fail
                                                                         ; interrupt

         000001                           .END
```

Figure C-6 Power Fail Recovery

```
.MAIN.  MACRO V04.00  8-FEB-82 04:20:45 PAGE 1

                                          .ENABL  LC,LSB
                                          .GLOBL  RCSR1,ERROR,PRINT,SLUE,SLGOOD
 000000                                   .PSECT  ROM

 000000                           SLUTST::
                                  ;+
                                  ;  This routine checks the auxiliary SLU port on the KXT11-AA
                                  ;-

 000000  012701  000000G                  MOV     #RCSR1,R1              ; Point to the address
 000004  105761  000002                   TSTB    2(R1)                  ; Flush the contents of RBUF
 000010  012761  000006  000004           MOV     #6,4(R1)               ; Set the SLU for maintenance and
                                                                         ; programmable baud rates
 000016  012702  000010                   MOV     #8.,R2                 ; Initialize the baud rate counter
 000022  012703  000132'          1$:     MOV     #PATERN,R3             ; Point to the test patterns
 000026  005005                   2$:     CLR     R5                     ; Initialize time out counter
 000030  105761  000004           3$:     TSTB    4(R1)                  ; Loop the pattern around
 000034  100402                           BMI     4$                     ; Branch if ready to send
 000036  077504                           SOB     R5,3$                  ; If not ready, bump time out counter
 000040  000422                           BR      100$                   ; IF timed out then - ERROR -
 000042  111361  000006           4$:     MOVB    (R3),6(R1)             ; Send the information out
 000046  005005                           CLR     R5                     ; Initialize the time out counter
 000050  105711                   5$:     TSTB    (R1)                   ; Is the receiver ready ?
 000052  100402                           BMI     6$                     ; Yes it is and branch
 000054  077503                           SOB     R5,5$                  ; If not ready, bump time out counter
 000056  000413                           BR      100$                   ; If timed out then -ERROR-
 000060  126113  000002           6$:     CMPB    2(R1),(R3)             ; Was the information sent OK ?
 000064  001010                           BNE     100$                   ; No it was not -ERROR-
 000066  105723                           TSTB    (R3)+                  ; All of the test paterns done ?
 000070  001356                           BNE     2$                     ; No, so do another pattern
 000072  005302                           DEC     R2                     ; All of the baud rates tested ?
 000074  001412                           BEQ     200$                   ; Yes, get out of this routine
 000076  062761  000010  000004           ADD     #10,4(R1)              ; No, set-up the next baud rate
 000104  000746                           BR      1$                     ; Do another loop, reinit patterns

 000106  005267  000000G          100$:   INC     ERROR                  ; Bump the error counter
 000112  004767  000000G                  CALL    PRINT                  ; Print the error message
 000116  000000G                          .WORD   SLUE
 000120  000403                           BR      150$                   ; Go back
 000122  004767  000000G          200$:   CALL    PRINT                  ; The test was successful
 000126  000000G                          .WORD   SLGOOD
 000130  000207                   150$:   RETURN                         ; Bye

 000132     177     040     000   PATERN: .BYTE   177,40,0               ; Test patterns for SLU
                                          .EVEN


                                          .DSABL  LSB
         000001                           .END
```

Figure C-7 SLU Diagnostic Task

```
.MAIN.  MACRO V04.00  8-FEB-82 04:20:54 PAGE 1

                                                .ENABL  LC,LSB
                                                .GLOBL  RAMBGN,PRINT,RAMTOP,MESRA1,RAGOOD,ERROR
000000                                          .PSECT  ROM

000000                                  RAMTST::
                                        ;+
                                        ;  This routine checks the user RAM on the KXT11-AA
                                        ;-

000000  011602                                  MOV     (SP),R2                 ; Save the return address
000002  016703  000000G                         MOV     ERROR,R3                ; Save the contents of the ERROR flag
000006  012700  000000G                         MOV     #RAMBGN,R0              ; Point to the start of the user RAM

000012  010010                          1$:     MOV     R0,(R0)                 ; Write the address
000014  020010                                  CMP     R0,(R0)                 ; Read it back
000016  001405                                  BEQ     2$                      ; Was the value read correctly
000020  004767  000000G                         CALL    PRINT                   ; No, report the failure,
000024  000000G                                 .WORD   MESRA1                  ;
000026  005203                                  INC     R3                      ; set the error flag,
000030  000407                                  BR      3$                      ; and go back
000032  005720                          2$:     TST     (R0)+                   ; Go onto the next location
000034  020027  000002G                         CMP     R0,#RAMTOP+2            ; Until there is no more to test
000040  103764                                  BLO     1$
000042  004767  000000G                         CALL    PRINT                   ; Indicate RAM test success
000046  000000G                                 .WORD   RAGOOD

000050  010216                          3$:     MOV     R2,(SP)                 ; Restore the return address
000052  010367  000000G                         MOV     R3,ERROR                ; Restore the ERROR flag
000056  000207                                  RETURN                          ; Test completed

                                                .DSABL  LSB
        000001                                  .END
```

Figure C-8   RAM Diagnostic Task

```
.MAIN.  MACRO V04.00  8-FEB-82 04:20:58 PAGE 1

                                                .ENABL  LC,LSB
                                                .GLOBL  REC,LAST,CSUM,PRINT,MESRO1,ROGOOD,ERROR
000000                                          .PSECT  ROM

000000                                  ROMTST::
                                        ;+
                                        ;  This routine will check the ROM on the KXT11-AA, this test checks
                                        ;  the portion of the ROM that contains the actual control/monitor tasks
                                        ;-

000000  012700  000000G                         MOV     #REC,R0                 ; Point to the control task address
000004  005001                                  CLR     R1                      ; Initialize checksum value
000006  062001                          1$:     ADD     (R0)+,R1                ; Update value
000010  022700  000002G                         CMP     #LAST+2,R0              ; Until there are no values to sum
000014  001374                                  BNE     1$                      ; If there are still some go get them
000016  022701  000000G                         CMP     #CSUM,R1                ; Are the checksums equal ?
000022  001406                                  BEQ     2$                      ; Yes, leave the test
000024  004767  000000G                         CALL    PRINT                   ; No, report the
000030  000000G                                 .WORD   MESRO1                  ; failure
000032  005267  000000G                         INC     ERROR                   ; Set the error flag
000036  000403                                  BR      3$                      ; Leave the test
000040  004767  000000G                 2$:     CALL    PRINT                   ; Report the test passed
000044  000000G                                 .WORD   ROGOOD
000046  000207                          3$:     RETURN

                                                .DSABL  LSB
        000001                                  .END
```

Figure C-9   ROM Diagnostic Task

```
.MAIN.  MACRO V04.00  8-FEB-82 04:20:50 PAGE 1

                                            .ENABL  LC,LSB
                                            .GLOBL  PPA,PRINT,PGOOD
 000000                                     .PSECT  ROM

 000000                             PIOTST::
                                    ;+
                                    ;  This routine checks the parallel ports on the KXT11-AA this only
                                    ;  test the ability to address the port
                                    ;-

 000000  012701  000003                     MOV     #3,R1               ; Initialize loop counter
 000004  005000                             CLR     R0                  ; Initialize counting index
 000006  005760  000000G            1$:     TST     PPA(R0)             ; Attempt to address PIO port if the
                                                                        ; attempt fails a trap through the
                                                                        ; restart will occur and report a run
                                                                        ; time error
 000012  005720                             TST     (R0)+               ; Increment the index, this will not
                                                                        ; time out since there is memory at
                                                                        ; locations 2-4
 000014  077104                             SOB     R1,1$               ; Do the port
 000016  004767  000000G                    CALL    PRINT               ; Indicate success
 000022  000000G                            .WORD   PGOOD

 000024  000207                             RETURN

                                            .DSABL  LSB
         000001                             .END
```

Figure C-10 Parallel I/O Diagnostic Task

```
.MAIN.  MACRO V04.00  8-FEB-82 04:21:02 PAGE 1
CONTROL AND MONITORING TASKS

                                            .SBTTL  CONTROL AND MONITORING TASKS

                                            .ENABL  LC
                                            .GLOBL  TIME,LEDON,PCW,RCSRC,LEDOFF
 000000                                     .PSECT  ROM

 000000                             REC::
                                    ;+
                                    ; This interrupt routine accepts an input from the console. When the input is
                                    ; received a ten second counter is initialized and the LED is turned on.
                                    ;-

 000000  012767  001130  000000G            MOV     #<10.* 60.>,TIME        ; Set timer for ten seconds
 000006  012737  000000G 000000G            MOV     #LEDON,@#PCW            ; Turn the LED on
 000014  105737  000002G                    TSTB    @#RCSRC+2               ; Flush the receive buffer
 000020  000002                             RTI                             ; Go back

 000022                             TIMER::
                                    ;+
                                    ; This interrupt routine when entered every clock tick will decrement the ten
                                    ; second counter and turn off the LED if the time is expired, otherwise it
                                    ; returns immediately.
                                    ;-

 000022  005767  000000G                    TST     TIME                    ; If the time is set update the
 000026  001406                             BEQ     GOBACK                  ; counter otherwise go back
 000030  005367  000000G                    DEC     TIME                    ; Yes, bump the counter and if it is
 000034  001003                             BNE     GOBACK                  ; The last tick then shut the LED off
 000036  012737  000000G 000000G            MOV     #LEDOFF,@#PCW           ; Otherwise go back
 000044  000002                     GOBACK: RTI

 000046                             BREAK::
                                    ;+
                                    ; This interrupt service routine will be entered if a break detected , this
                                    ; is treated as a regular input on the KXT11-AA console port.
                                    ;-

 000046  012767  001130  000000G            MOV     #<10. * 60.>,TIME       ; Set the timer for ten seconds
 000054  012737  000000G 000000G            MOV     #LEDON,@#PCW            ; Turn the LED on
 000062  000002                     LAST::  RTI                             ; Go back

         000001                             .END
```

Figure C-11 Control Task

```
.MAIN.  MACRO V04.00  8-FEB-82 04:21:06 PAGE 1

                                          .ENABL  LC
                                          .GLOBL  POWER1,POWER2,TIME,SAVER6,PRINT,FMESS
                                          .MCALL  PUSH
000000                                    .PSECT  ROM
000000                             POWERF::
                                   ;+
                                   ; This routine is entered when a power fail is detected and saves the
                                   ; pertinent information in non-volatile RAM
                                   ;-
000000  012767  123456  000000G           MOV     #123456,POWER1          ; Initialize the 32-bit power recovery
000006  012767  135724  000000G           MOV     #135724,POWER2          ; test pattern
000014                                    PUSH    R0                      ; Save the general purpose registers
000016                                    PUSH    R1                      ; and any pertinent data in a non-
000020                                    PUSH    R2                      ; volatile RAM area
000022                                    PUSH    R3
000024                                    PUSH    R4
000026                                    PUSH    R5
000030                                    PUSH    TIME
000034  010667  000000G                   MOV     SP,SAVER6               ; Save the stack pointer in the non-
                                                                          ; volatile ram area
000040  000005                            RESET                           ; Put the bus in a known state
000042  000777                            BR      .                       ; and wait for loss of power
000044                             RESTRT::
                                   ;+
                                   ; When a bus error occurs such as an interrupt time-out or bus time-out
                                   ; a trap thru the restart takes place and comes here
                                   ;-
000044  004767  000000G                   CALL    PRINT                   ; Indicate that a run-time error has
000050  000000G                           .WORD   FMESS                   ; occurred and wait for operator
000052  000777                            BR      .                       ; intervention
        000001                            .END
```

Figure C-12   Power Fail Task

APPENDIX D
MACRO-ODT ROM LISTING
FOR KXT11-A2 OPTION

Appendix D provides the user with the program listing of the KXT11-A2 Macro-ODT ROM firmware option.

TABLE OF CONTENTS


```
    3-    1      COPYRIGHT NOTICE
    4-    1      KXT11-A2 EDIT HISTORY
    5-    1      Equates
    6-    1      General DLART Equates
    8-    1      General PPI Equates
    9-    1      Program-specific Equates
   11-    1      MACRO DEFINITIONS
   13-    1      RAM Definition
   14-    3      TRAPS-Trap-handling routines
   14-    4      TRAPS-LTC Trap-killer
   14-    5      TRAPS-BREAK handler
   15-    1      RESTART-Introduction
   19-    1      RESTART-Entry point
   20-    1      RESTART-See if stack exists
   20-   19      RESTART-Exit if in IN-ROM state
   21-    1      RESTART-Cause determination
   22-    1      RESTART-Exits
   23-    1      POWERUP-Introduction
   24-    1      POWERUP-Turn on LED
   24-   22      POWERUP-Test console DLART
   25-    1      POWERUP-Test and set up I/O-page RAM
   26-    1      POWERUP-Turn off LED
   26-   29      POWERUP-Test for "low core"
   27-    1      POWERUP-Exit
   27-   20      POWERUP-Subroutine to initialize vectors
   28-    1      AUTOBAUD-Synchronize with Console
   30-    1      macroODT-Introduction
   32-    1      macroODT-Save status and print prompt
   33-    1      macroODT-Get ODT command
   35-    1      macroODT- Go and Proceed
   36-    1      macroODT-Register and PS command
   37-    1      macroODT-Examine and Deposit
   39-    1      macroODT-Get and echo character
   40-    1      macroODT-Type ASCII string
   41-    1      macroODT-Get octal digits
   42-    1      macroODT-OCTSTR--type binary in R0 as ASCII
   43-    1      macroODT-Output messages
   44-    1      DIAGNOSTICS-for SLU2 and PPI
   45-    1      HARDWARE ENTRY POINT
   46-    1      DIAGNOSTICS-Continued
   47-    1      BOOTS-Description
   48-    9      BOOTS-RX Controller Definitions
   48-   56      BOOTS-TU58 Definitions and Protocol Equates
   48-  114      BOOTS-RT11 Definitions and Equates
   49-    1      BOOTS-Program entry point
   49-   42      -----> HALT AT PC=172234 INDICATES "Illegal device name"
   49-   51      -----> HALT AT PC=172264 INDICATES "Illegal unit number"
   49-   58      -----> HALT AT PC=172304 INDICATES "No low memory, can't boot"
   49-   92      -----> HALT AT PC=172376 INDICATES "Unexpected timeout during boot"
   50-    1      BOOTS-RX01/RX02 Bootstrap
   51-    1      BOOTS-Distinguishing type of boot block
   51-   23      -----> HALT AT PC=172454 INDICATES "No boot block on volume"
   52-    1      BOOTS-TU58 Bootstrap
   52-   29      -----> HALT AT PC=172542 INDICATES "TU58 initialization error"
   52-   37      -----> HALT AT PC=172562 INDICATES "TU58 block 0 read error"
   53-    1      BOOTS-Stand-alone volume bootstrap
   53-   24      -----> HALT AT PC=172614 INDICATES "Directory read error"
```

TABLE OF CONTENTS


```
   53-  36      -----> HALT AT PC=172652 INDICATES "File not found"
   54-   1      BOOTS-Load Stand-Alone Program File
   54-   8      -----> HALT AT PC=172732 INDICATES "Stand-alone file read error"
   54-  12      -----> HALT AT PC=172750 INDICATES "Illegal transfer address"
   55-   1      173000G ENTRY POINT
   56-   1      BOOTS-Continued
   57-   1      BOOTS-RX01/RX02 Read routines
   57-  36      -----> HALT AT PC=173070 INDICATES "Floppy drive not ready"
   57- 114      -----> HALT AT PC=173262 INDICATES "Floppy read error"
   60-   1      BOOTS-TU58 Read routines
   61-  27      -----> HALT AT PC=173556 INDICATES "TU58 END packet missing"
   61-  37      -----> HALT AT PC=173610 INDICATES "TU58 checksum error"
   63-   1      END STATEMENT
```

```
                                        .TITLE KXT11-A2 1K FIRMWARE
                                        .IDENT  /V1.00/
                                        .ENABL LC

                                ; Place identification number in last ROM location:
  000000                        .ASECT
             173776             .=173776
  173776        000             .BYTE   0
  173777        001             .BYTE   1.
```

```
        .SBTTL COPYRIGHT NOTICE
;
; COPYRIGHT (C) 1980, 1981 BY
; DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASS.
;
; THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND COPIED
; ONLY IN ACCORDANCE WITH THE TERMS OF SUCH LICENSE AND WITH THE
; INCLUSION OF THE ABOVE COPYRIGHT NOTICE.  THIS SOFTWARE, OR ANY OTHER
; COPIES THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY
; OTHER PERSON.  NO TITLE TO AND OWNERSHIP OF THE SOFTWARE IS HEREBY
; TRANSFERRED.
;
; THE INFORMATION IN THIS DOCUMENT IS SUBJECT TO CHANGE WITHOUT NOTICE
; AND SHOULD NOT BE CONSTRUED AS A COMMITMENT BY DIGITAL EQUIPMENT
; CORPORATION.
;
; DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE OR RELIABILITY OF ITS
; SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.
;
; VERSION V1.00
;
; EL 29-SEP-81
```

```
                                        .SBTTL  KXT11-A2 EDIT HISTORY

                                ;EDIT HISTORY:
                                ;
```

```
                                                .SBTTL Equates

                          ; BIT EQUATES

        000001            BIT0     =       1
        000002            BIT1     =       2
        000004            BIT2     =       4
        000010            BIT3     =       10
        000020            BIT4     =       20
        000040            BIT5     =       40
        000100            BIT6     =       100
        000200            BIT7     =       200
        000400            BIT8     =       400
        001000            BIT9     =       1000
        002000            BIT10    =       2000
        004000            BIT11    =       4000
        010000            BIT12    =       10000
        020000            BIT13    =       20000
        040000            BIT14    =       40000
        100000            BIT15    =       100000

                          ; ASCII CHARACTER EQUATES

        000012            LF       =       12                      ;Line feed
        000015            CR       =       15                      ;Carriage return
        000040            SPACE    =       40                      ;Space
```

```
                                        .SBTTL General DLART Equates

                        ; DLART EQUATES

        177560          RCSR$1  =       177560          ;SLU1 Receive CSR
        177562          RBUF$1  =       177562          ;SLU1 Receive buffer
        177564          XCSR$1  =       177564          ;SLU1 Xmit CSR
        177566          XBUF$1  =       177566          ;SLU1 Xmit buffer
        176540          RCSR$2  =       176540          ;SLU2 Receive CSR
        176542          RBUF$2  =       176542          ;SLU2 Receive buffer
        176544          XCSR$2  =       176544          ;SLU2 Xmit CSR
        176546          XBUF$2  =       176546          ;SLU2 Xmit buffer

                        ; DLART RECEIVE CSR BITS

        004000          RC.ACT  =       BIT11           ;Receiver active (R/O).  Set
                                                        ; while character is being
                                                        ;  received.
        000200          RC.DUN  =       BIT7            ;Receiver done (R/O).  A
                                                        ; character has been completely
                                                        ;  received and now resides
                                                        ;   in RBUF.
        000100          RC.IEN  =       BIT6            ;Receiver int. enable (R/W).
                                                        ; When set, enables "keyboard"
                                                        ;  interrupts, using vector
                                                        ;   at 60.

                        ; DLART RECEIVE BUFFER BITS (R/O)

        100000          RB.ERR  =       BIT15           ;Error.  Framing error or
                                                        ; overrun has occurred.
        040000          RB.OVR  =       BIT14           ;Overrun error.  Character was
                                                        ; received before previous one
                                                        ;  was read.
        020000          RB.FRM  =       BIT13           ;Framing error.  No valid stop
                                                        ; bit was detected.
        004000          RB.BRK  =       BIT11           ;Break detect.  Set when break
                                                        ; is detected, reset when next
                                                        ;  start bit arrives.
```

```
                                ; DLART TRANSMIT CSR BITS

        000200                  XC.RDY  =       BIT7                    ;Transmitter ready (R/O).
                                                                        ; When set, indicates that the
                                                                        ;  last character was completely
                                                                        ;   sent and XBUF is ready for
                                                                        ;    a new one.
        000100                  XC.IEN  =       BIT6                    ;Transmit int. enable (R/W).
                                                                        ;When set, enables "console
                                                                        ; printer" interrupts, using
                                                                        ;  vector at 64.

                                ; Programmable baud rate bits

        000010                  PBR0    =       BIT3
        000020                  PBR1    =       BIT4
        000040                  PBR2    =       BIT5

                                ; PBR0-2 set baud rates as follows:

        000000                  BD.003  =       0                       ;Baud rate =   300
        000010                  BD.006  =       PBR0                    ;Baud rate =   600
        000020                  BD.012  =       PBR1                    ;Baud rate =  1200
        000030                  BD.024  =       PBR1!PBR0               ;Baud rate =  2400
        000040                  BD.048  =       PBR2                    ;Baud rate =  4800
        000050                  BD.096  =       PBR2!PBR0               ;Baud rate =  9600
        000060                  BD.192  =       PBR2!PBR1               ;Baud rate = 19200
        000070                  BD.384  =       PBR2!PBR1!PBR0          ;Baud rate = 38400

        000004                  XC.MNT  =       BIT2                    ;Maintenance (R/W).  When set,
                                                                        ; creates an internal "loop-
                                                                        ;  back" between the transmitter
                                                                        ;   and receiver.  Also dis-
                                                                        ;    connects the external
                                                                        ;     serial input.
        000002                  XC.PBE  =       BIT1                    ;Prog. baud rate enable.  When
                                                                        ; set, the baud rate is deter-
                                                                        ;  mined by bits 3-5 as
                                                                        ;   tabulated above.  WHEN
                                                                        ;    CLEAR, BAUD RATE IS DETER-
                                                                        ;     MINED BY VOLTAGES APPLIED
                                                                        ;      TO DLART IC PINS.
        000001                  XC.BRK  =       BIT0                    ;Transmit break (R/W).  When
                                                                        ; set, serial output is a
                                                                        ;  continuous BREAK.
```

```
                                        .SBTTL  General PPI Equates

                                ; PROGRAMMABLE PERIPHERAL INTERFACE (PPI) EQUATES

        176206                  PP.CWR  =       176206                  ;PPI Control Word Register
        176200                  PP.A    =       176200                  ;PPI Port A Register
        176202                  PP.B    =       176202                  ;PPI Port B Register
        176204                  PP.C    =       176204                  ;PPI Port C Register

                                ; PPI MODE-SETTING BITS

                                ; KXT11-AA board configuration does not permit all combinations of
                                ; the mode bits.  Consult the manual before using the PPI.

        000200                  PP.MOD  =       BIT7                    ;This MUST be or'd with other
                                                                        ; bits to set mode.
        000100                  PP.MD2  =       BIT6                    ;Sets mode 2
        000040                  PP.MDA  =       BIT5                    ;If bit 6 is low, determines
                                                                        ; mode of port A
                                                                        ;  (hi=mode 1, lo=mode 0)
        000020                  PP.DRA  =       BIT4                    ;Direction of port A.
                                                                        ; Hi=IN, lo=OUT.
        000010                  PP.CHI  =       BIT3                    ;Direction of port C upper half
                                                                        ; Hi=IN, lo=OUT.
        000004                  PP.MDB  =       BIT2                    ;Mode of port B.
                                                                        ; Hi=mode 1, lo=mode 0.
        000002                  PP.DRB  =       BIT1                    ;Direction of port B.
                                                                        ; Hi=IN, lo=OUT.
        000001                  PP.CLO  =       BIT0                    ;Direction of port C lower half
                                                                        ; Hi=IN, lo=OUT.

                                ; PPI BIT SET/RESET CONTROL BITS

                                ;       When bit 7 is low, writing to the PPI CSR will set or reset
                                ;       individual bits in Port C, depending on the mode and direction
                                ;       of the port's bits, and on the combination of bits you write.

        000016                  PP.BI7  =       BIT3!BIT2!BIT1          ;Use ONE
        000014                  PP.BI6  =       BIT3!BIT2               ;of these
        000012                  PP.BI5  =       BIT3!BIT1               ;to select
        000010                  PP.BI4  =       BIT3                    ;which bit
        000006                  PP.BI3  =       BIT2!BIT1               ;is desired
        000004                  PP.BI2  =       BIT2                    ;to be
        000002                  PP.BI1  =       BIT1                    ;SET or
        000000                  PP.BI0  =       0                       ;CLEARed

        000001                  PP.BIS  =       BIT0                    ;SET specified bit.
        000000                  PP.BIC  =       0                       ;CLEAR specified bit.
```

```
                                .SBTTL  Program-specific Equates

                        ; EQUATES USED TO TURN LED ON AND OFF

        000221          MODE    =       PP.MOD!PP.DRA!PP.CLO    ;Port A = Mode 0 IN
                                                                ;Port B = Mode 0 OUT
                                                                ;Port C upper nibble = OUT
                                                                ;Port C lower nibble = IN

        000017          LEDOFF  =       PP.BIS!PP.BI7           ;Set PC7

                        ; EQUATES USED TO SET UP DLARTS

        000032          BAUDRS  =       BD.024!XC.PBE           ;Initial console baud rate to
                                                                ; be 2400, with prog. baud
                                                                ;  rates enabled.

        000072          TUBAUD  =       BD.384!XC.PBE           ;TU58 Baud rate = 38,400

                        ; MEMORY CONFIGURATION EQUATES

        160010          RAMBOT  =       160010                  ;Bottom address of RAM
        167776          RAMTOP  =       167776                  ;Top address of RAM

                        ; SOFTWARE FLAGS AND MASKS

        000300          PRI6    =       300                     ;PS for priority of 6
        000340          PRI7    =       340                     ;PS for priority of 7

                        ; USED BY ODT MODULE

        000200          RFLAG   =       BIT7                    ;Register flag bit- Indicates
                                                                ; register is being examined
        000020          T.BIT   =       BIT4                    ;Trace bit in PSW
```

```
                        ; RESTART TYPE WORD BITS

        100000          R.HALT  =       BIT15           ;HALT or BREAK occurred
        000200          R.NXM   =       BIT7            ;Accessed non-existent memory
        000001          R.STAK  =       BIT0            ;Double-bus error

                        ; BOOT CONTROL WORD BITS

        100000          NO.LOW  =       BIT15           ;No memory found at 000000-
                                                        ; do not boot
        000200          DEVBIT  =       BIT7            ;1 = RX01/02 floppy
                                                        ;0 = TU58 cassette
        000001          DEVNUM  =       BIT0            ;Unit no. (0 or 1)

                        ; DIAGNOSTIC MESSAGES

        000100          E.EXT   =       100             ;SLU2 loopback test failed
        000010          E.INT   =       10              ;SLU2 internal loopback failed
        000001          E.PAR   =       1               ;Parallel port loopback failed
```

```
                                .SBTTL MACRO DEFINITIONS

        ;
        ;       MACRO DEFINITIONS
        ;

        ;+
        ;
        ; This macro will insert ABORTS into the code which will halt the
        ; program, exit to ODT with the PC printed on the console, and generate
        ; an entry in the table of contents which describes the error condition.
        ;
        ;-

        .MACRO  ABORT   TEXT
                HALT
        .IRP    PC$,\.
        .SBTTL  -----> HALT AT PC='PC$ INDICATES "'TEXT"
        .ENDR
                BR      .-2
        .ENDM
```

```
                ;+
                ; DELAY A,B,N
                ;       where A and B are names of registers that are free (both will
                ;       be clear when through) and N is an integer.
                ;
                ; This macro produces a delay whose duration (when running in KXT11-AA
                ; ROM) is .2399N seconds.
                ; When N<4, it is more efficient to use the following code:
                ;
                ;       CLR     Rn                      ;1W           2.44
                ;       SOB     Rn,.                    ;1W      239861.76
                ;       [SOB    kn,.                    ;1W      239861.76]
                ;       [SOB    Rn,.                    ;1W      239861.76]
                ;
                ; The macro generates code like the following:
                ;
                ;       MOV     #N,Ra                   ;2W           3.66
                ;n$:    CLR     Rb                      ;1W         N*2.44
                ;       SOB     Rb,.                    ;1W    65536N*3.66
                ;       SOB     Ra,n$                   ;1W         N*3.66
                ;-

                        .MACRO  DELAY   A,B,N,?L
                        MOV     #N,A
                L:      CLR     B
                        SOB     B,.
                        SOB     A,L
                        .ENDM
```

```
                                        .SBTTL RAM Definition

                                ; SCRATCH RAM AREA

        167776                  TRAP4   ==      167776          ;Enables trap-to-4 emulation
                                                                ; when non-zero
        167774                  ODTWHY  ==      167774          ;User-readable copy of R.TYPE.
                                                                ; Restart cause.  See R.TYPE
                                                                ;  table in RESTART routine.
        167772                  O.CNTL  ==      167772          ;ODT Control word.  Set Bit 15
                                                                ; to disable T-Bit filter, set
                                                                ;  Bit 7 to disable Priority 7
                                                                ;   filter.
        167770                  B.CNTL  ==      167770          ;Boot control word.
        167766                  R.PC    ==      167766          ;Where restart saves top of stack
        167764                  IN.USR  ==      167764          ;Enables user-caused restart
                                                                ; and BREAK when non-zero
        167762                  R.TYPE  ==      167762          ;Restart cause.  See table in
                                                                ; RESTART routine.
        167760                  USERSP  ==      167760          ;Used by ODT to store the user's
                                                                ; stack pointer.
        167756                  RPOINT  ==      167756          ;Used by ODT to point to the image
                                                                ; of user's R0 in its stack.
        167754                  SAVPS   ==      167754          ;Store halted PS here for ODT
        167752                  SAVPC   ==      167752          ;Store halted PC here for ODT
        167750                  ODTFLG  ==      167750          ;Used by ODT for internal flags.
        167746                  ODTLOC  ==      167746          ;Used by ODT to point to location
                                                                ; currently open.
        167744                  ODTSTK  ==      167744          ;Bottom of ODT's stack
        167644                  $STACK  ==      ODTSTK-100      ;Bottom of default user stack
```

```
        170000                            .=170000

                                                  .SBTTL TRAPS-Trap-handling routines
                                                  .SBTTL TRAPS-LTC Trap-killer
                                                  .SBTTL TRAPS-BREAK handler

                                  ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
                                  ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
                                  ;;;;                                                    ;;;;
                                  ;;;;                 BREAK-HANDLING ROUTINE              ;;;;
                                  ;;;;          AND LINE TIME CLOCK INTERRUPT KILLER       ;;;;
                                  ;;;;                                                    ;;;;
                                  ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
                                  ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

 170000                           $$$BRK::
 170000 005767  177760                    TST     IN.USR                  ;Are we in user mode?
 170004 001001                            BNE     BRKN00                  ;YES-Go to ODT

 170006                           $$$LTC::
 170006 000002                            RTI                             ;NO-Go back to ROM program.
                                                                          ; BREAKS are ignored by ODT,
                                                                          ;  RESTART, POWERUP and the
                                                                          ;   DIAGNOSTICS.  The BOOTS
                                                                          ;    can be interrupted, though.
 170010                           BRKN00::
 170010 012667  177736                    MOV     (SP)+,SAVPC             ;Save context
 170014 012667  177734                    MOV     (SP)+,SAVPS             ;for ODT.
 170020 012767  100000  177734            MOV     #R.HALT,R.TYPE          ;Causes PC to be printed
                                                                          ; upon entry to ODT.
 170026 005067  177732                    CLR     IN.USR                  ;Get out of user mode
 170032 000167  000544                    JMP     ODT
```

```
                .SBTTL RESTART-Introduction

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;;                                                    ;;;;
;;;;                 RESTART MODULE                     ;;;;
;;;;                                                    ;;;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

;+
;
;The purpose of the RESTART routine is to restore the FALCON to a
;known state following those exceptions which cause a RESTART hardware
;action.  This action consists of stacking the current PSW and program
;counter, then setting the PSW to 340 and jumping to the hardwired
;RESTART location.  This location is at the address START+4 where
;START is jumper selectable as 000000, 010000, 020000, 040000, 100000,
;140000, 172000 or 173000 (all in octal).  This program is designed
;for a START location of 172000, thus RESTARTs jump to 172004.
;
;There are several different ways in which RESTART performs its
;function, depending on the value of IN.USR, TRAP4, the contents
;of the location the SP points to, and one bit (R.STAK) in R.TYPE.
;
;R.TYPE, the restart type word, is RESTART's output to ODT.
;
;-

;+
;
;The goal is to maximize PDP-11 software compatibility and to provide
;useful debugging information to the program developer.
;
;-
```

```
;             ----------
;             | R.STRT |   Enter via hardware mechanism,
;             ----------   with (PS)=340
;                  |
;          ----------------
;          | Is the stack |
;          |  flag set?   |
;          |  N        Y  |
;          ----------------
;             |        |
;             |   -------------------
;             |   |    Set R.NXM    |
;             |   | Set IN-ROM mode |
;             |   |    Go to ODT    |
;             |   -------------------
; --------------------------------
; |        Set stack bit         |
; |      Read top of stack       |<---Could time out and cause the
; | Check if too close to "hole" |<-----exit to ODT shown above
; |       Clear stack bit        |
; --------------------------------
;             |
;        ---------------------
;        | Did restart occur |
;        |   in user mode?   |
;        |  N             Y  |
;        ---------------------
;           |             |
; -----------------       |
; |   Is top of   |<------+----- A BREAK does this when
; | stack 000000? |       |      there's no memory in
; |  N         Y  |       |      the vector area.
; -----------------       |
;    |         |          |
;    |   ---------------  |
;    |   |  Pop stack  |<-+----- Only a BREAK while in ODT can
;    |   |  frame and  |  |      get us here, so the RTI takes
;    |   |   return    |  |      us back to ODT.
;    |   ---------------  |
;  ----------------   -------------------
;  |  Set carry   |   | Leave user mode |
;  | in pushed PS |   -------------------
;  |  and return  |       |
;  ----------------    -----------------
;                      |   Is top of   |
;                      | stack 000000? |
; A BREAK does this--->|  Y         N  |
; when there's no      -----------------
; memory in the           |         |
; vector area   ----------------    |
;               |  Pop stack   |<---+------This entry point is in
;               | frame and go |    |      the TRAPS module.  It is
;               |  to BREAK's  |    |      where a BREAK in user mode
;               | SAVE CONTEXT |    |      goes when there IS memory
;               | entry point  |    |      in the vector area.
;               ----------------    |
;                                   |
```

```
;                                         |
;                               ----------------------
;                               | Pop stack frame    |
;                               | if 172004 on top.  |
;                               |  Get pushed PC.    |
;                               |  Set up ODT's PC   |
;                               | and PS locations.  |
;                               ----------------------
;                                         |
;                              -----------------------
;                 COULD---->|  Test word prior     |
;                 TIME----->|  to where pushed     |
;                 OUT------>|     PC points        |
;                              -----------------------
;                                         |
;                               ---------------------
;                               |    Was the word   |
;                               |      a HALT       |
;                               | or did PC-->NXM?  |
;                               |   Y          N    |
;                               ---------------------
;                                   |          |
;                          -----------------   |
;                          | Set HALT flag |   |
;                          |   Go to ODT   |   |
;                          -----------------   |
;                                              |
;                                 ----------------
;                                 | Is trap-to-4 |
;                                 |  emulation   |
;                                 |   enabled?   |
;                                 |  N        Y  |
;                                 ----------------
;                                    |        |
;                        ----------------     |
;                        | Set NXM flag |     |
;                        |  Go to ODT   |     |
;                        ----------------     |
;                                             |
;                                   -------------------
;                                   |  Set user mode  |
;                                   |  Push @#6, @#4  |
;                                   |   onto stack    |
;                                   |     and RTI     |
;                                   -------------------
```

```
;+
;
;Exception-type word (R.TYPE) is passed to ODT and is RESTARTs "best guess"
;as to why a restart happened:
;
;       Note: A user-readable copy of this word is at ODTWHY.
;
;|------|-----|--------|---------------------------------------------
;| EXIT | BIT |  NAME  | CAUSE
;|------|-----|--------|---------------------------------------------
;|------|     |        |
;| ODT  | 15  | R.HALT | HALT instruction in user code-RESTART POPS STACK.
;|      |     |        | note-BREAK also sets this bit (see the TRAPS
;|      |     |        | module).  ODT uses this bit for PDP-11 ODT
;|      |     |        | compatibility.
;|------|     |        |
;| ODT  | 14  |        | Reserved
;| OR   | 13  |        | Reserved
;| TRAP | 12  |        | Reserved
;| TO   | 11  |        | Reserved
;| FOUR | 10  |        | Reserved
;|      |  9  |        | Reserved
;|      |  8  |        | Reserved
;|      |  7  | R.NXM  | Timeout during user access of non-existant
;|      |     |        |   memory
;|      |  6  |        | Reserved
;|      |  5  |        | Reserved
;|      |  4  |        | Reserved
;|      |  3  |        | Reserved
;|      |  2  |        | Reserved
;|      |  1  |        | Reserved
;|------|-----|--------|
;| ODT  |  0  | R.STAK | Indicates that a timeout was caused by RESTART
;|      |     |        |   itself accessing non-existant memory.  This
;|      |     |        |   occurs in conjunction with testing for
;|      |     |        |   validity of the stack pointer.
;|      |     |        |   In PDP-11 parlance, this is a
;|      |     |        |   "double-bus error"
;|------|-----|--------|---------------------------------------------
;
;-
```

```
                                        .SBTTL RESTART-Entry point

                        ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
                        ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
                        ;;;;                                                      ;;;;
                        ;;;;                RESTART ENTRY POINT                   ;;;;
                        ;;;;                                                      ;;;;
                        ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
                        ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

170036                  R.STRT::

                        ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
                        ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
                        ;;;;                                                      ;;;;
                        ;;;;    IF THE RESTART ROUTINE CAUSED THE RESTART         ;;;;
                        ;;;;             GO TO ODT AND PRINT "?"                  ;;;;
                        ;;;;                                                      ;;;;
                        ;;;;     THIS EXCEPTION CAN BE CAUSED BY RESTART'S        ;;;;
                        ;;;;              STACK MANIPULATIONS                     ;;;;
                        ;;;;                                                      ;;;;
                        ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
                        ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

                        ; R.TYPE will have been cleared prior to entering
                        ; any ODT command.  So, if the stack bit is set, only RESTART
                        ; itself could have caused the trap.  Since the stack is always
                        ; valid in in-ROM mode, bad stack means we are in in-USER mode.

                        ;State: X=don't care, U=user, R=in-ROM----
                        ;                                        |
170036  005767  177720          TST     R.TYPE                   ;X|Did the stack test fail?
170042  001406                  BEQ     1$                       ;X|NO- go to next test
170044  052767  000200  177710  BIS     #R.NXM,R.TYPE            ;U|YES- set R.NXM
                                                                 ;U| this forces "?" from ODT
170052  005067  177706          CLR     IN.USR                   ;R|enter in-ROM mode
170056  000476                  BR      8$                       ;R|go to ODT
```

```
                                            .SBTTL  RESTART-See if stack exists
                                ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
                                ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
                                ;;;;                                                        ;;;;
                                ;;;;                STACK VALIDITY TEST                     ;;;;
                                ;;;;                                                        ;;;;
                                ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
                                ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
170060  052767  000001  177674  1$:     BIS     #R.STAK, R.TYPE         ;X|If we timeout, we want RESTART
                                                                        ;X| to know we were diddling SP
170066  005716                          TST     (SP)                    ;X|see if stack is valid
170070  000240                          NOP                             ;X|(in case times out)
170072  005766  000004                  TST     4(SP)                   ;X|see if too close to top of
170076  000240                          NOP                             ;X| valid memory
170100  005067  177656                  CLR     R.TYPE                  ;X|stack is OK
                                            .SBTTL RESTART-Exit if in IN-ROM state
                                ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
                                ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
                                ;;;;                                                        ;;;;
                                ;;;;     RETURN WITH CARRY SET IF IN "IN-ROM" MODE          ;;;;
                                ;;;;                                                        ;;;;
                                ;;;;  OR, GO BACK TO ODT IF A BREAK WITH NO LOW MEMORY      ;;;;
                                ;;;;                                                        ;;;;
                                ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
                                ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
170104  005767  177654                  TST     IN.USR                  ;X|Are we in user mode?
170110  001007                          BNE     3$                      ;U|YES-go to next test
170112  005716                          TST     (SP)                    ;R|NO-see if BREAK brought
                                                                        ;R| us here
170114  001002                          BNE     2$                      ;R|NO-Just a RESTART
170116  022626                          CMP     (SP)+,(SP)+             ;R|YES-Behave like a BREAK that
170120  000002                          RTI                             ;R| happened with RAM
170122  005266  000002          2$:     INC     2(SP)                   ;R|Set carry in pushed PS
                                                                        ;R| UNLESS ALREADY SET
170126  000002                          RTI                             ;R|and return to ROM code that
                                                                        ;R| caused timeout
```

```
                                                     .SBTTL RESTART-Cause determination


                                  ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
                                  ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
                                  ;;;;                                                           ;;;;
                                  ;;;;            DETERMINE HOW USER CAUSED A RESTART            ;;;;
                                  ;;;;                                                           ;;;;
                                  ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
                                  ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

170130  005067  177630            3$:     CLR     IN.USR                  ;U|We were in user mode,
                                                                          ;R| but no longer.
170134  005716                            TST     (SP)                    ;R|See if BREAK brought
                                                                          ;R| us here without low "core".
170136  001003                            BNE     4$                      ;R|NO-Just a RESTART
170140  022626                            CMP     (SP)+,(SP)+             ;R|YES-Behave like a BREAK that
170142  000167  177642                    JMP     BRKNO0                  ;R| happened while in user prog.



                                  ;
                                  ;       If the CPU attempts to fetch an instruction from non-existent
                                  ;       memory, two traps (the first from executing a HALT, the second
                                  ;       from timing out) will occur, the result being that second
                                  ;       trap pushes the restart address and 340 on the stack.
                                  ;       This information is useless and gets popped here.

170146  021627  172004            4$:     CMP     (SP),#RESTAR            ;X|Get rid of double stacking
170152  001001                            BNE     5$                      ;X|caused by EXECUTION of NXM
170154  022626                            CMP     (SP)+,(SP)+             ;X|

                                  ;       Note: Because the contents of the stack is assumed to remain
                                  ;       unchanged following the first instruction below, it is imperative
                                  ;       that interrupts be disabled during the next three instrutions.

170156  012667  177604            5$:     MOV     (SP)+,R.PC              ;R|Get pushed PC
170162  011667  177566                    MOV     (SP),SAVPS              ;R|ODT would like
170166  014667  177560                    MOV     -(SP),SAVPC             ;R|to see these

170172  162767  000002  177566            SUB     #2,R.PC                 ;R|Set pointer to last word fetched
                                                                          ;R| before restart occurred
170200  005777  177562                    TST     @R.PC                   ;R|Is contents of pushed PC - 2
                                                                          ;R| a zero (eg a HALT)?
170204  000240                            NOP                             ;R|Make sure next instruction
                                                                          ;R| won't execute if we time out
170206  001005                            BNE     6$                      ;R|NO- it was an NXM
170210  052767  100000  177544            BIS     #R.HALT,R.TYPE          ;R|YES- Flag a HALT,
170216  022626                            CMP     (SP)+,(SP)+             ;R|pop the non-PDP-11 stack frame
170220  000415                            BR      8$                      ;R|and go to ODT.
```

```
                                                        .SBTTL RESTART-Exits

                                          ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
                                          ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
                                          ;;;;                                                    ;;;;
                                          ;;;;                 EXIT APPROPRIATELY                 ;;;;
                                          ;;;;                                                    ;;;;
                                          ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
                                          ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

   170222  005767  177550          6$:    TST     TRAP4                   ;R|Trap-to-4 emulation enabled?
   170226  001407                         BEQ     7$                      ;R|NO-go to ODT
   170230  005167  177530                 COM     IN.USR                  ;U|YES-Set user mode
   170234  013746  000006                 MOV     @#6,-(SP)               ;U|Emulate a
   170240  013746  000004                 MOV     @#4,-(SP)               ;U|trap to
   170244  000002                         RTI                             ;U|four

   170246  052767  000200  177506  7$:    BIS     #R.NXM, R.TYPE          ;R|flag NXM error
   170254  000167  000322          8$:    JMP     ODT                     ;R|go to ODT
```

```
                .SBTTL POWERUP-Introduction

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;;                                                    ;;;;
;;;;                   POWER-UP MODULE                  ;;;;
;;;;                                                    ;;;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

; This module contains a series of routines which perform
; tests on the on-board RAM and the console DLART.  These
; tests are preceded by the lighting of the LED on the
; KXT11-AA board, and followed by its extinguishing.  Should
; the LED fail to either light or go out, there may be a
; defect in the board or its configuration.

; Following these tests, the on-board RAM is written with the
; default values of certain control words, and, if there is
; memory in the vector region (i.e., near 000000), the BREAK
; and clock vectors are set up.  If not, a bit is set in the
; boot control word to disable the bootstraps.
```

```
                                                        .SBTTL POWERUP-Turn on LED

                                        ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
                                        ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
                                        ;;;;                                                    ;;;;
                                        ;;;;                  TURN ON LED                       ;;;;
                                        ;;;;                                                    ;;;;
                                        ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
                                        ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;


170260                                  PWR$UP::
170260  012706  167644                          MOV     #$$TACK,SP              ;Initialize stack pointer

                                        ; Because a mode-setting command automatically clears all the internal
                                        ; registers in the PPI, and clearing Port C Bit 7 turns on the LED, all
                                        ; we have to do is set the mode, which is port A and lo half of C as
                                        ; input, ports B and hi half of C as output.

170264  012737  000221  176206                  MOV     #MODE,@#PP.CWR          ;Set proper PPI mode

                                                        .SBTTL POWERUP-Test console DLART

                                        ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
                                        ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
                                        ;;;;                                                    ;;;;
                                        ;;;;            CHECK THE CONSOLE DLART                 ;;;;
                                        ;;;;                                                    ;;;;
                                        ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
                                        ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

170272  005037  177564                          CLR     @#XCSR$1                ;Disable XMIT interrupts,
                                                                                ; BRK XMIT, maint. mode
                                                                                ;  Set baud rate to default
170276  005737  177562                          TST     @#RBUF$1                ;Take out the trash.
170302  032737  000300  177560                  BIT     #<RC.IEN!RC.DUN>,@#RCSR$1
                                                                                ;Should be clear.
170310  001377                                  BNE     .                       ;If not, drop dead.
170312  023727  177564  000200                  CMP     @#XCSR$1,#XC.RDY        ;Should be set
170320  001377                                  BNE     .                       ;If not, rest in peace.
```

```
                                            .SBTTL POWERUP-Test and set up I/O-page RAM

                             ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
                             ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
                             ;;;;                                                      ;;;;
                             ;;;;                   I/O PAGE RAM TEST                  ;;;;
                             ;;;;                   AND INITIALIZATION                 ;;;;
                             ;;;;                                                      ;;;;
                             ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
                             ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

                             ; Write the location's address into the location and read it back.
                             ; Do this for all I/O page RAM locations.
                             ; If it fails, enter tight loop.

                             ; In the process, clear all of this RAM.  Note that the default
                             ; value of most of the control and flag words is zero.

170322  012700  160010               MOV     #RAMBOT,R0              ;Lowest address of RAM
170326  010010               1$:     MOV     R0,(R0)                 ;Write the address
170330  020010                       CMP     R0,(R0)                 ;Read it back
170332  001377               2$:     BNE     2$                      ;Tight failure loop
170334  005020                       CLR     (R0)+                   ;Clear and go on to next location
170336  020027  170000               CMP     R0,#RAMTOP+2            ;Until no more to test.
170342  103771                       BLO     1$
```

```
                                          .SBTTL POWERUP-Turn off LED

                                ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
                                ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
                                ;;;;                                                        ;;;;
                                ;;;;  DELAY SO THAT LED IS ON A VISIBLE LENGTH OF TIME      ;;;;
                                ;;;;                                                        ;;;;
                                ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
                                ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

170344  005000                          CLR     R0
170346  077001                  3$:     SOB     R0,3$                   ;This leaves a 0 in R0, which
170350  077001                  4$:     SOB     R0,4$                   ; is essential for testing for
170352  077001                  5$:     SOB     R0,5$                   ;  the presence of memory at
170354  077001                  6$:     SOB     R0,6$                   ;   zero below.

                                ; Under no circumstances can R0 be altered until "low core" test below.

                                ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
                                ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
                                ;;;;                                                        ;;;;
                                ;;;;                    TURN OFF LED                        ;;;;
                                ;;;;                                                        ;;;;
                                ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
                                ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

170356  012737  000017  176206          MOV     #LEDOFF,@#PP.CWR

                                          .SBTTL POWERUP-Test for "low core"

                                ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
                                ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
                                ;;;;                                                        ;;;;
                                ;;;;            TEST FOR MEMORY AT 000000                   ;;;;
                                ;;;;                                                        ;;;;
                                ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
                                ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

                                ; Read memory at 000000, discard result.  If this fails, exit to
                                ; AUTOBAUD rather than continuing with normal powerup sequence.

170364  005710                          TST     (R0)
170366  000240                          NOP                             ;This will execute even if
                                                                        ; last instruction times out
170370  103403                          BCS     7$                      ;Timed out, don't set vectors
170372  004767  000042                  CALL    VECSET                  ;Didn't time out, so go ahead
170376  000403                          BR      8$                      ;Don't set the NO.LOW flag

170400  052767  100000  177362  7$:     BIS     #NO.LOW,B.CNTL          ;Did time out,
                                                                        ; so let the world know.
```

```
                                                  .SBTTL POWERUP-Exit

                                  ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
                                  ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
                                  ;;;;                                                      ;;;;
                                  ;;;;              EXIT FROM POWER-UP SEQUENCE             ;;;;
                                  ;;;;                                                      ;;;;
                                  ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
                                  ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

170406  012767  000300  177340    8$:     MOV     #PR16,SAVPS             ;If P is typed in reponse to
170414  012767  170424  177330            MOV     #FAKOUT,SAVPC           ; ODT prompt before loading R7,
170422  000423                            BR      AUTOBA                  ;  will force yet more ODT.

170424                            FAKOUT:
170424  005067  177334                    CLR     IN.USR                  ;   BUT IN THE RIGHT MODE!
170430  012706  167644                    MOV     #$STACK,SP              ;    And without running out of
170434  000167  000142                    JMP     ODT                     ;     stack, either.

                                                  .SBTTL POWERUP-Subroutine to initialize vectors

                                  ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
                                  ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
                                  ;;;;                                                      ;;;;
                                  ;;;;                  INITIALIZE VECTORS                  ;;;;
                                  ;;;;                                                      ;;;;
                                  ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
                                  ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

                                  ; Note: This subroutine is also used by the bootstrap module, to
                                  ; restore the vector area in the event that an invalid boot block
                                  ; was read into low memory.

170440                            VECSET::
170440  012737  170000  000140            MOV     #$$$BRK,@#140           ;Set up the BREAK-detect
170446  012737  000340  000142            MOV     #PR17,@#142             ; vector.
170454  012737  170006  000100            MOV     #$$$LTC,@#100           ;Set up the line time clock
170462  012737  000340  000102            MOV     #PR17,@#102             ; vector.
170470  000207                            RETURN
```

```
                                        .SBTTL  AUTOBAUD-Synchronize with Console

                                ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
                                ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
                                ;;;;                                                    ;;;;
                                ;;;;                    AUTOBAUD MODULE                 ;;;;
                                ;;;;                                                    ;;;;
                                ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
                                ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

                                ; Description:
                                ;
                                ;       AUTOBAUD allows the FALCON to automatically synchronize its
                                ;       console DLART to the baud rate of the console terminal.
                                ;       On power-up, the user must type a carriage return character.
                                ;       Upon synchronization, AUTOBAUD will proceed to ODT where an '@'
                                ;       character will be displayed on the console.
                                ;
                                ;       Autobaud will loop indefinitely until synchronization is successful.
                                ;
                                ;       The algorithm requires that the console terminal generates a
                                ;       zero (space) for the eighth bit in the carriage return.  This
                                ;       will happen if the terminal is capable of sending eight-bit-
                                ;       no-parity or seven-bit-odd-parity characters.
                                ;
                                ; Environment:
                                ;
                                ;       Interrupts must be disabled for the algorithm to execute correctly
                                ;       since time durations are critical and delays due to long
                                ;       service routines may cause DLART overruns, which this routine
                                ;       ignores but cannot tolerate.
                                ;



                                ; VT103/FALCON configurations leave garbage in the DLART long after the
                                ; powerup sequence has begun.  We must delay a bit before clearing garbage
                                ; out of the DLART, otherwise the garbage would arrive after the clear
                                ; (i.e., while polling for input).  The "garbage" is an X-ON (<CTRL-q>)
                                ; that the VT-100 hardware sends after its power-up diagnostics have
                                ; completed successfully.
                                ;

170472                          AUTOBA::
170472  012737  000032  177564          MOV     #BAUDR$,@#XCSR$1        ;Set 2400 baud
170500  005000                          CLR     R0                      ;Delay
170502  077001                          SOB     R0,.                    ;       .5
170504  077001                          SOB     R0,.                    ;          seconds
```

```

                                        ; AUTOBAUD proper:
 170506  105737  177562                 10$:    TSTB    @#RBUF$1                ; discard any garbage

 170512  105737  177560                 20$:    TSTB    @#RCSR$1                ; wait for input
 170516  100375                                 BPL     20$
 170520  113700  177562                         MOVB    @#RBUF$1, R0            ; R0 = input character
 170524  012701  170550                         MOV     #INBYTE, R1             ; R1 -> scrambled char table
 170530  120021                         30$:    CMPB    R0, (R1)+               ; in the table?
 170532  001411                                 BEQ     HVBAUD                  ; yes
 170534  020127  170556                         CMP     R1, #INBYTS             ; end of table reached?
 170540  001373                                 BNE     30$                     ; not yet
 170542  005000                                 CLR     R0                      ; uh oh, wait for DLART to clear out
 170544  077001                         40$:    SOB     R0, 40$                 ; wait for a while
 170546  000757                                 BR      10$                     ; and try for another character

                                        ; Table of what you would see if an octal 15 were sent at the following
                                        ; baud rates.

 170550                                 INBYTE:
 170550     200                                 .BYTE   200                     ; 300
 170551     170                                 .BYTE   170                     ; 600
 170552     346                                 .BYTE   346                     ; 1200
 170553     015                                 .BYTE   15                      ; 2400
 170554     362                                 .BYTE   362                     ; 4800
 170555     377                                 .BYTE   377                     ; 9600, 19200, 38400
 170556                                 INBYTS:


                                        ; We have a match.  Set baud rate into DLART.
 170556                                 HVBAUD:
 170556  162701  170551                         SUB     #INBYTE+1, R1           ; turn pointer into bit mask
 170562  006301                                 ASL     R1
 170564  006301                                 ASL     R1
 170566  005201                                 INC     R1                      ; turn on XC.PBE
 170570  006301                                 ASL     R1                      ; set the baud rate
 170572  010137  177564                         MOV     R1, @#XCSR$1            ; into CSR
 170576  005000                                 CLR     R0                      ; delay .24 seconds for rest
 170600  077001                                 SOB     R0,.                    ; of char. at slo baud rates

                                        ; Fall into ODT.
```

```
                                        .SBTTL  macroODT-Introduction

                        ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
                        ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
                        ;;;;                                                      ;;;;
                        ;;;;                       macroODT                       ;;;;
                        ;;;;                                                      ;;;;
                        ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
                        ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

                        ; macroODT is the user interface to the functions contained
                        ; in the KXT11-A2 firmware product.  It interprets commands
                        ; entered via the console terminal keyboard (see tables below)
                        ; to permit the user to load a program into memory, execute
                        ; it and debug it.

                        ;   COMMAND
                        ;   1- Slash (/)
                        ;      a-OPEN MEMORY LOCATION
                        ;      b-OPEN GENERAL REGISTER
                        ;      c-OPEN STATUS REGISTER
                        ;   2- Carriage return (<CR>)
                        ;      a-CHANGE AND CLOSE MEMORY LOCATION OR REGISTER
                        ;      b-CLOSE WITHOUT CHANGE
                        ;   3- Line feed (<LF>)
                        ;      a- CHANGE AND CLOSE MEMORY LOCATION AND OPEN NEXT
                        ;      b- CLOSE MEMORY LOCATION WITHOUT CHANGING AND OPEN NEXT
                        ;   4- Go (G)
                        ;   5- Proceed (P)
                        ;   6- Execute I/O diagnostics (X)
                        ;   7- Execute bootstraps (D)
```

```
;SYNTAX OF COMMANDS LISTED ABOVE, SHOWING CONSOLE BEFORE,
;DURING AND AFTER THE TYPING OF THE COMMAND.
;       Key: n-an octal integer typed by the user, only
;               last 6 digits significant
;            x-a single octal digit
;            u-the digits 0 or 1
;            all other characters are literals
;
;   BEFORE        DURING                AFTER
;
;1a @             @n/                   @n/xxxxxx
;1b @             @Rx/                  @Rx/xxxxxx
;1c @             @RS/                  @RS/xxxxxx
;2a @n/xxxxxx     @n/xxxxxx n<CR>       @
;2a @Rx/xxxxxx    @Rx/xxxxxx n<CR>      @
;2a @RS/xxxxxx    @Rx/xxxxxx n<CR>      @
;2a xxxxxx/xxxxxx xxxxxx/xxxxxx n<CR>   @
;2b @n/xxxxxx     @n/xxxxxx <CR>        @
;2b @Rx/xxxxxx    @Rx/xxxxxx <CR>       @
;2b @RS/xxxxxx    @RS/xxxxxx <CR>       @
;2b xxxxxx/xxxxxx xxxxxx/xxxxxx <CR>    @
;3a @n/xxxxxx     @n/xxxxxx n<LF>       xxxxxx/xxxxxx
;3a xxxxxx/xxxxxx xxxxxx/xxxxxx n<LF>   xxxxxx/xxxxxx
;3b @n/xxxxxx     @n/xxxxxx <LF>        xxxxxx/xxxxxx
;3b xxxxxx/xxxxxx xxxxxx/xxxxxx <LF>    xxxxxx/xxxxxx
;4  @             @nG
;5  @             @P
;6  @             @X                    xxxxxx
;                                       @
;7  @             @DDu
;7  @             @DXu
;7  @             @DYu
;7  @             @DD<CR>
;7  @             @DX<CR>
;7  @             @DY<CR>
```

```
                                        .SBTTL macroODT-Save status and print prompt

                                ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
                                ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
                                ;;;;                                                        ;;;;
                                ;;;;      SAVE CONTEXT, PRINT MESSAGES AND PROMPT          ;;;;
                                ;;;;                                                        ;;;;
                                ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
                                ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

170602                          ODT::
170602  105737  177562                  TSTB    @#RBUF$1                ;Clear out console garbage

                                ; Copy the restart type word into user area

170606  016767  177150  177160          MOV     R.TYPE,ODTWHY

                                ; Protect against stack timeouts, but save user's SP first

170614  010667  177140                  MOV     SP,USERSP               ;SAVE USERS STACK POINTER
170620  012706  167744                  MOV     #ODTSTK,SP              ;LOAD NEW SP

                                ; Save rest of user program's context

170624  016716  177130                  MOV     USERSP,(SP)             ;RESERVE LOCATION FOR R6
170630  010546                          MOV     R5,-(SP)                ;SAVE
170632  010446                          MOV     R4,-(SP)                ; ALL
170634  010346                          MOV     R3,-(SP)                ;  OF
170636  010246                          MOV     R2,-(SP)                ;   USER'S
170640  010146                          MOV     R1,-(SP)                ;    GENERAL
170642  010046                          MOV     R0,-(SP)                ;     REGISTERS
170644  010667  177106                  MOV     SP,RPOINT               ;POINTER TO R0

                                ; Determine whether "?" or PC message is appropriate, and print it

170650  005767  177106                  TST     R.TYPE                  ;Did we get a HALT or BREAK?
170654  100004                          BPL     QODT                    ;NO-next question
                                                                        ;YES-PRINT PC
170656  016700  177070                  MOV     SAVPC,R0                ;GET STOPPED PC
170662  004767  000764                  CALL    OCTSTO                  ;TYPE THE PC ON TERMINAL
170666                          QODT:
170666  105767  177070                  TSTB    R.TYPE                  ;SEE IF RESTART OCCURRED
                                                                        ;(NXM ONLY-BIT 7 SET)
170672  100003                          BPL     KBD$                    ;TYPE PROMPT

                                ; Here's where the prompt gets printed, with or without leading "?"

170674                          KBDQ:
170674  012700  171730                  MOV     #MSGQ,R0                ; GET ? ADDRESS
170700  000402                          BR      PRINT                   ;TYPE IN MESSAGE
170702                          KBD$:
170702  012700  171731                  MOV     #MSG$,R0                ;GET PROMPT MESSAGE ADDRESS

170706  005067  177050          PRINT:  CLR     R.TYPE                  ;So reentry gives no error msg.
170712  106427  000300                  MTPS    #PRI6                   ;Allow BREAKs to happen
170716  004767  000620                  CALL    PUTSTR                  ;TYPE THE PROMPT ALREADY
170722  005067  177022                  CLR     ODTFLG                  ;CLEAR FLAG FOR NEW ENTRY
```

```
                                        .SBTTL  macroODT-Get ODT command

                                ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
                                ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
                                ;;;;                                                    ;;;;
                                ;;;;        INTERPRET FIRST CHARACTER OF COMMAND        ;;;;
                                ;;;;                                                    ;;;;
                                ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
                                ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

                                ; Note: Following CALL GETCHR, the character (7 bit ASCII)
                                ;  appears in R2.
                                ; Note: Following CALL GETNUM, if carry is clear, the octal integer
                                ;  was followed by a carriage return.
                                ; Note: On exit to LCSET or falling through to GO routine, R0 contains
                                ;  the address typed in.

170726  004767  000556                  CALL    GETCHR                  ;...INPUT CHARACTERS
170732  120227  000104                  CMPB    R2,#'D                  ;BOOTSTRAPS?
170736  001002                          BNE     1$                      ;NO
170740  000167  001220                  JMP     BOOTS                   ;YES

170744  120227  000130          1$:     CMPB    R2,#'X                  ;DIAGNOSTICS?
170750  001002                          BNE     2$                      ;NO
170752  000167  000776                  JMP     DIAGNO                  ;YES

170756  120227  000120          2$:     CMPB    R2,#'P                  ;PROCEED?
170762  001430                          BEQ     PCMD                    ;YES
170764  120227  000122                  CMPB    R2,#'R                  ;REGISTER?
170770  001465                          BEQ     RCMD                    ;YES
170772  120227  000060                  CMPB    R2,#'0                  ;OCTAL DIGIT?
170776  103736                          BLO     KBDQ                    ;NO,ERROR
171000  120227  000070                  CMPB    R2,#'8                  ;VALID DIGIT?
171004  103333                          BHIS    KBDQ                    ;NO,ERROR
171006  005000                          CLR     R0                      ;ITS A DIGIT
171010  004767  000576                  CALL    GETNUM                  ;GET REST OF THE DIGIT OR CMD
171014  103327                          BCC     KBDQ                    ;CR WAS ISSUED,ERROR

                                ; The last character at the end of the number could be a valid command-
                                ; Let's check:

171016  120227  000057                  CMPB    R2,#'/                  ;EXAMINE LOCATION?
171022  001511                          BEQ     LCSET                   ;YES
171024  120227  000107                  CMPB    R2,#'G                  ;GO TO?
171030  001321                          BNE     KBDQ                    ;NO,ERROR
```

```
; TABLE OF PERMISSABLE STATES
;
;       NO.      STATE             VALID INPUTS          COMMENT
;       1-       prompt @              0-7       --------> digit.
;                                       P        --------> proceed.
;                                       R        --------> register designator.
;                                       X        --------> execute diagnostic
;                                       D        --------> boot from device
;       2-      @175620                0-7       --------> another digit.
;              [input digit]            /        --------> examine loc.
;                                       G        --------> go from loc n.
;       3-      @176000/000002         0-7       --------> input new value.
;                                       LF       --------> display next loc.
;                                       CR       --------> close loc go to prompt.
;       4-      @200/000023 12         0-7       --------> input more digits.
;                                       LF       --------> save data display next.
;                                       CR       --------> save data go to prompt.
;       5-      @R                     0-7       --------> register number.
;                                       S        --------> PSW.
;       6-      @R5                     /        --------> examine.
;       7-      @R5/000024             0-7       --------> input new value.
;                                       CR       --------> close location.
;       8-      @R5/000024 16          0-7       --------> more digits input
;                                       CR       --------> save value go to prompt
;
```

```
                                          .SBTTL  macroODT- Go and Proceed

                                  ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
                                  ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
                                  ;;;;                                                             ;;;;
                                  ;;;;            PROCESS GO AND PROCEED ODT COMMANDS              ;;;;
                                  ;;;;                                                             ;;;;
                                  ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
                                  ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

171032  010067  176714                    MOV     R0,SAVPC              ;PUT SUPPLIED PC IN MEMORY LOCATION

                                  ; Prepare the environment for the Go command

171036  000005                            RESET                         ;BUS INITIALIZE
171040  005067  176710                    CLR     SAVPS                 ;CLEAR PSW

                                  ; Entry point for the Proceed command

                                  ; First, check for valid stack:

171044                            PCMD:

171044  016600  000014                    MOV     14(SP),R0             ;User's stack pointer
171050  005740                            TST     -(R0)                 ;where SAVPS will go (see below)
171052  000240                            NOP                           ; (in case of time out)
171054  103403                            BCS     1$                    ;No good.  Timed out.
171056  005740                            TST     -(R0)                 ;where SAVPC will go
171060  000240                            NOP                           ;
171062  103004                            BCC     2$                    ;Sufficient stack.

                                  ; EITHER Stack no good, so simulate a double bus trap without losing the
                                  ;        user's context as stored in the ODT stack.

171064  012767  000201  176702    1$:     MOV     #R.STAK!R.NXM,ODTWHY  ;Sneaky! (R.TYPE untouched-
                                                                        ; only the user image of it)
171072  000700                            BR      KBDQ                  ;Error prompt.

                                  ; OR     Stack is OK, so restore user's context.

171074  012600                    2$:     MOV     (SP)+,R0              ;RESTORE
171076  012601                            MOV     (SP)+,R1              ; ALL
171100  012602                            MOV     (SP)+,R2              ;  OF
171102  012603                            MOV     (SP)+,R3              ;   USER'S
171104  012604                            MOV     (SP)+,R4              ;    GENERAL
171106  012605                            MOV     (SP)+,R5              ;     REGISTERS

171110  106427  000340                    MTPS    #PRI7                 ;No BREAKS allowed until out of
                                                                        ; ODT!
171114  042716  000001                    BIC     #BIT0,(SP)            ;Odd stacks are too odd for T-11
171120  011606                            MOV     (SP),SP               ;RESTORE USER SP
171122  005167  176636                    COM     IN.USR                ;Set user mode
171126  016746  176622                    MOV     SAVPS,-(SP)           ;RESTORE PC AND PS TO ...
171132  016746  176614                    MOV     SAVPC,-(SP)           ;...STACK WHERE RTT WILL LOOK
171136  000006                            RTT                           ;RETURN TO USERS PROGRAM
171140  000655                    HKBDQ:  BR      KBDQ                  ;HELP IN BR
171142  000657                    HKBD$:  BR      KBD$                  ;HELP IN BR
```

```
                                        .SBTTL  macroODT-Register and PS command

                                ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
                                ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
                                ;;;;                                                    ;;;;
                                ;;;;            PROCESS ODT REGISTER COMMANDS           ;;;;
                                ;;;;                                                    ;;;;
                                ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
                                ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

                                ; Entry point for Rx and RS commands

171144                          RCMD:
171144  052767  000200  176576          BIS     #RFLAG,ODTFLG           ;SET REGISTER FLAG
171152  004767  000420                  CALL    ONENUM                  ;GET REGISTER NUMBER
171156  103246                          BCC     KBDQ                    ;A VALID CMD DID NOT FOLLOW
171160  120227  000123                  CMPB    R2,#'S                  ;IS IT THE RS?
171164  001412                          BEQ     SWCMD                   ;YES,BRANCH
171166  120227  000057                  CMPB    R2,#'/                  ;EXAMINE?
171172  001240                          BNE     KBDQ                    ;NO,ERROR
171174  020027  000007                  CMP     R0,#7                   ;>7?
171200  101235                          BHI     KBDQ                    ;YES,ERROR
171202  001013                          BNE     RCMD1                   ;IS IT EXACTLY SEVEN
171204  012700  167752                  MOV     #SAVPC,R0               ;YES,GET PC ADDRESS
171210  000413                          BR      REGOUT                  ;DISPLAY

                                ; Status register (PS) selected:

171212                          SWCMD:
171212  004767  000272                  CALL    GETCHR                  ;WHAT YOU WANT TO DO WITH RS?
171216  120227  000057                  CMPB    R2,#'/                  ;EXAMINE?
171222  001224                          BNE     KBDQ                    ;NO,ERROR
171224  012700  167754                  MOV     #SAVPS,R0               ;GET ADDRESS WHERE PS IS
171230  000403                          BR      REGOUT                  ;GO AND DISPLAY

171232  006300                  RCMD1:  ASL     R0                      ;SHIFT FOR OFFSET IN MEMORY
171234  066700  176516                  ADD     RPOINT,R0               ;GET EXACT ADDRESS OF REG.
171240  010067  176502          REGOUT: MOV     R0,ODTLOC               ;STORE LOCATION
171244  000402                          BR      LOCDSP                  ;DISPLAY
```

```
                                        .SBTTL macroODT-Examine and Deposit

                                ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
                                ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
                                ;;;;                                                          ;;;;
                                ;;;;  PROCESS ODT MEMORY AND REGISTER EXAMINE/DEPOSIT        ;;;;
                                ;;;;                                                          ;;;;
                                ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
                                ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

                                ; ODTLOC points to register or memory location
                                ; Following CALL GETNUM, if carry is clear, CR followed digit.
                                ; ODTFLG: If register bit set indicates register is being examined

                                ;ENTRY FROM CMD ROUTINE AFTER LOC. VALUE IS GIVEN

171246  010067  176474          LCSET:  MOV     R0,ODTLOC               ;SAVE NEW LOCATION
171252  011000                  LOCDSP: MOV     (R0),R0                 ;GET DATA
171254  000240                          NOP                             ;So next inst. does not execute
                                                                        ;if we time out.
171256  103730                          BCS     HKBDQ                   ;Print "?" if we timed out
171260  004767  000372                  CALL    OCTSTR                  ;PRINT IT
171264  112702  000040                  MOVB    #SPACE,R2               ;Print a space after the data
171270  004767  000226                  CALL    PUTCHR
171274  004767  000210                  CALL    GETCHR                  ;GET NEXT CHARACTER
171300  120227  000015                  CMPB    R2,#CR                  ;FINISH
171304  001716                          BEQ     HKBDS                   ;YES,CLOSE LOCATION
171306  120227  000060                  CMPB    R2,#'0                  ;DEPOSIT?
171312  103450                          BLO     4$                      ;NO,CHECK LF
171314  120227  000070                  CMPB    R2,#'8                  ;MAYBE!
171320  103307                          BHIS    HKBDQ                   ;NO,FORGET IT
171322  005000                          CLR     R0                      ;YES
171324  004767  000262                  CALL    GETNUM                  ;GET REST OF NUMBER
171330  103006                          BCC     1$                      ;CR FOUND, STORE NEW VALUE

171332  120227  000012                  CMPB    R2,#LF                  ;Not CR, must be LF
171336  001300                          BNE     HKBDQ                   ;Print error message
171340  105767  176404                  TSTB    ODTFLG                  ;If LF, cannot be register
171344  100675                          BMI     HKBDQ                   ;(Error exit)

                                ;T-BIT FILTER.  The T-BIT can be set from the keyboard via ODT.
                                ;This can either be useful for debugging or disastrous.  So, you can
                                ;do it only if you first set FILT.T in O.CNTL (BIT 15).

171346  022767  167754  176372  1$:     CMP     #SAVPS,ODTLOC           ;Are we diddling the PS?
171354  001021                          BNE     3$                      ;No, we're not.
171356  042700  177400                  BIC     #^C<377>,R0             ;PS is not a word.
171362  005767  176404                  TST     O.CNTL                  ;Is BIT 15 (FILT.T) SET?
171366  100402                          BMI     2$                      ;Yes, the filter's disabled
171370  042700  000020                  BIC     #T.BIT,R0               ;KILL THE T-BIT

                                ;2$:                                    ;Fall thru to Priority 7
                                                                        ; Filter
```

```
                                       ;Priority-7 filter: unless FILT.7 (BIT 7) in O.CNTL is set, you cannot
                                       ;actually set the PS to priority 7 using ODT from the keyboard.  This
                                       ;protects the ability to break.

171374  105767  176372         2$:     TSTB    O.CNTL                  ;ODT control word
171400  100407                         BMI     3$                      ;Do nothing-filter disabled
171402  105700                         TSTB    R0                      ;Intended new PS
171404  100005                         BPL     3$                      ;Do nothing-Priority < 4
171406  032700  000100                 BIT     #BIT6,R0                ;Check again
171412  001402                         BEQ     3$                      ;Do nothing-Priority < 6
171414  042700  000040                 BIC     #BIT5,R0                ;LOWER THE BOOM
171420  010077  176322         3$:     MOV     R0,@ODTLOC              ;STORE NEW VALUE
171424  120227  000012                 CMPB    R2,#LF                  ;Go on to next location?
171430  001407                         BEQ     5$                      ;Sure, why not.
171432  000643                         BR      HKBDS                   ;GO TO PROMPT

171434  120227  000012         4$:     CMPB    R2,#LF                  ;IS A LF ISSUED
171440  001237                         BNE     HKBDQ                   ;NO,ERROR
171442  105767  176302                 TSTB    ODTFLG                  ;IS REGISTER FLAG SET
171446  100634                         BMI     HKBDQ                   ;YES, LF NOT PERMITTED
171450  112702  000015         5$:     MOVB    #CR,R2                  ;TO LINE UP CURSOR
171454  004767  000042                 CALL    PUTCHR                  ;SEND IT
171460  062767  000002  176260         ADD     #2,ODTLOC               ;GET ADDRESS OF NEXT LOC.
171466  016700  176254                 MOV     ODTLOC,R0               ;GET NEXT ADDRESS VALUE...
171472  004767  000160                 CALL    OCTSTR                  ;...AND PRINT IT
171476  112702  000057                 MOVB    #'/,R2                  ;SEND A SLASH BEFORE...
171502  004767  000014                 CALL    PUTCHR                  ;...SHOWING THE CONTENTS...
171506  000661                         BR      LOCDSP                  ;...OF THE LOCATION
```

```
                                        .SBTTL  macroODT-Get and echo character

                                ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
                                ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
                                ;;;;                                                    ;;;;
                                ;;;;        CHARACTER INPUT AND ECHO SUBROUTINE         ;;;;
                                ;;;;                                                    ;;;;
                                ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
                                ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

                                ; Get a character from the console keyboard and echo it back
                                ; exactly as received including parity bits if any.  Return with
                                ; character in R2, eighth bit (and high byte) zero.

   171510                       GETCHR:
   171510  105737  177560               TSTB    @#RCSR$1                ;CHARACTER READY?
   171514  100375                       BPL     GETCHR                  ;BRANCH IF NOT AND KEEP TRYING
   171516  113702  177562               MOVB    @#RBUF$1,R2             ;TRANSFER CHARACTER
   171522                       PUTCHR:
   171522  105737  177564               TSTB    @#XCSR$1                ;PRINTER READY
   171526  100375                       BPL     PUTCHR                  ;NO, TRY AGAIN
   171530  110237  177566               MOVB    R2,@#XBUF$1             ;YES, XMIT CHARACTER
   171534  042702  177600               BIC     #^C<177>,R2             ;CLEAR PARITY
   171540  000207                       RETURN                          ;CONTINUE
```

```
                                .SBTTL  macroODT-Type ASCII string

                        ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
                        ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
                        ;;;;                                                    ;;;;
                        ;;;;            MESSAGE PRINT SUBROUTINE                ;;;;
                        ;;;;                                                    ;;;;
                        ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
                        ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

                        ; Print message starting with character pointed to by R0 and
                        ; ending with first character with eighth bit set (this character
                        ; is not printed).

171542                  PUTSTR:
171542  112002                  MOVB    (R0)+,R2                ;GET ASCII CHAR
171544  100413                  BMI     DONE                    ;IS IT THE END MARK?
171546  004767  177750          CALL    PUTCHR                  ;NO, PRINT IT
171552  000773                  BR      PUTSTR                  ;MORE

                        ;ENTRY FOR CARRIAGE RETURN

171554                  PUTCLF:
171554  112702  000015          MOVB    #CR,R2                  ;PRINT CR
171560  004767  177736          CALL    PUTCHR                  ;FALL THRU AND PRINT LF

                        ;ENTRY FOR LF

171564  112702  000012  PUTLF:  MOVB    #LF,R2                  ;PRINT LF
171570  004767  177726          CALL    PUTCHR
171574  000207          DONE:   RETURN
```

```
                                        .SBTTL  macroODT-Get octal digits

                                ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
                                ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
                                ;;;;                                                             ;;;;
                                ;;;;                 NUMERIC INPUT ROUTINE                       ;;;;
                                ;;;;                                                             ;;;;
                                ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
                                ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

                                ; On exit, R0 contains the binary representation of the number entered
                                ; If the carry bit is clear, a <CR> followed number
                                ; If the carry bit is set, some other character followed the number,
                                ; possibly a command.

171576                          ONENUM:
171576  005000                          CLR     R0                      ;CLEAR ACCUMULATOR
171600                          NEXNUM:
171600  004767  177704                  CALL    GETCHR                  ;GET DIGIT OR TERMINATOR
171604  120227  000015                  CMPB    R2,#CR                  ;CLEAR CARRY AND RETURN
171610  001412                          BEQ     SRET                    ;IF <CR> WAS TYPED
171612                          GETNUM:
171612  162702  000070                  SUB     #'8,R2                  ;CONVERT TO BINARY...
171616  062702  000010                  ADD     #'8-'0,R2               ;...AND TEST IF OCTAL OR NOT
171622  103007                          BCC     NOCT                    ;NOT VALID DIGIT.
171624  006300                          ASL     R0                      ;MAKE ROOM FOR NEW DIGIT
171626  006300                          ASL     R0                      ;DITTO
171630  006300                          ASL     R0                      ;DITTO
171632  050200                          BIS     R2,R0                   ;PUT IT IN PLACE
171634  000761                          BR      NEXNUM                  ;GET NEXT

171636  000241                  SRET:   CLC                             ;CLEAR CARRY
171640  000207                          RETURN                          ;CONTINUE

171642  062702  000060          NOCT:   ADD     #'0,R2                  ;RESTORE ASCII BECAUSE...
171646  000261                          SEC                             ;...POSSIBLE COMMAND
171650  000207                          RETURN                          ;CONTINUE
```

```
                                  .SBTTL  macroODT-OCTSTR--type binary in R0 as ASCII

                          ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
                          ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
                          ;;;;                                                          ;;;;
                          ;;;;                  NUMERIC OUTPUT ROUTINE                  ;;;;
                          ;;;;                                                          ;;;;
                          ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
                          ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

                          ; Prints, as a 6-digit octal integer, the value of the binary
                          ; number in R0.

171652  004767  177676    OCTSTO: CALL    PUTCLF                  ;NEED CRLF AT ODT ENTRY
171656                    OCTSTR:
171656  010046                    MOV     R0,-(SP)                ;SAVE VALUE
171660  012746  000006            MOV     #6,-(SP)                ;NO. OF CHARACTERS
171664  005002                    CLR     R2                      ;OUTPUT HOLD
171666  006100            5$:     ROL     R0                      ;SHIFT MSB INTO LSB
171670  006102                    ROL     R2                      ;" " " " " " " " "
171672  062702  000060            ADD     #'0,R2                  ;MAKE A DIGIT
171676  004767  177620            CALL    PUTCHR                  ;OUTPUT A CHARACTER
171702  005316                    DEC     (SP)                    ;COUNT
171704  001406                    BEQ     10$                     ;DONE
171706  005002                    CLR     R2                      ;NEXT
171710  006100                    ROL     R0                      ;GET NEXT DIGIT INTO
171712  006102                    ROL     R2                      ;R2
171714  006100                    ROL     R0                      ;FIRST TWO BITS
171716  006102                    ROL     R2                      ;" "  " "" " "
171720  000762                    BR      5$                      ;CONTINUE
171722  005726            10$:    TST     (SP)+                   ;CLEAR COUNT
171724  012600                    MOV     (SP)+,R0                ;ORIGINAL VALUE
171726  000207                    RETURN                          ;
```

```
                                                .SBTTL  macroODT-Output messages

                                        ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
                                        ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
                                        ;;;;                                                    ;;;;
                                        ;;;;                      MESSAGES                      ;;;;
                                        ;;;;                                                    ;;;;
                                        ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
                                        ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

                                                .NLIST  BEX
   171730       077                     MSGQ:   .ASCII  '?'                     ; ERROR MESSAGE
   171731       015     012     100     MSG@:   .ASCII  <CR><LF>'@'<200>        ; PROMPT
                                                .EVEN
                                                .LIST   BEX
```

```
                                        .SBTTL DIAGNOSTICS-for SLU2 and PPI

                                ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
                                ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
                                ;;;;                                                    ;;;;
                                ;;;;                DIAGNOSTIC MODULE                   ;;;;
                                ;;;;                                                    ;;;;
                                ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
                                ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

                                ; Diagnose PPI in mode 0 with loopback connectors installed.
                                ; Diagnose SLU2 internal circuitry (maintenance mode) and
                                ; SLU2 drivers/receivers (with external loopback connector).

                                ; List of error bit definitions to return to user.

171736  000100                          .WORD   E.EXT
171740  000010                          .WORD   E.INT
171742                          ERRBIT:

                                ; List of masks to put in XCSR$2.  Perform internal loopback
                                ; test first, then external loopback test.

171742  000000                          .WORD   0
171744  000002                          .WORD   XC.PBE                  ; 300 baud
171746  000006                          .WORD   XC.PBE ! XC.MNT         ; 300 baud and maintenance
171750                          INITS:

                                ; List of pattern bytes to loop around.
                                ; All bits on, alternating bits, all bits off.
                                ; Note: last byte must be 0.

171750     377     252     000  PATERN: .BYTE   377, 252, 0
                                        .EVEN

                                        .ENABL  LSB
171754                          DIAGNO:
171754  012737  000221  176206          MOV     #MODE,@#PP.CWR          ; set proper PPI mode- LED
171762  012737  000017  176206          MOV     #LEDOFF,@#PP.CWR        ; must immediately be turned
                                                                        ; off as a consequence.

171770  005000                          CLR     R0                      ; assume success

                                ; Perform parallel port diagnostic

171772  005001                          CLR     R1                      ; R1 = loop pattern
171774  000405                          BR      ARGUN2                  ; SKIP OVER THE ENTRY POINT
```

```
      1                                                  .SBTTL HARDWARE ENTRY POINT
      2
      3         172000                           .=172000
      4 172000                           START::
      5 172000  000167  176254                   JMP     PWRSUP
      6 172004                           RESTAR::
      7 172004  000167  176026                   JMP     K.$TRT
```

```
                                          .SBTTL DIAGNOSTICS-Continued
172010                          AROUN2:
172010 110137 176202            1$:     MOVB    R1, @#PP.B              ; send it out port B
172014 123701 176200                    CMPB    @#PP.A, R1              ; check input in port A
172020 001402                           BEQ     2$                      ; branch if same
172022 052700 000001                    BIS     #E.PAR, R0              ; else set error flag
172026 077110                   2$:     SOB     R1, 1$                  ; loop for all values

                                ; Perform SLU 2 diagnostic

172030 012702 171742                    MOV     #ERRBIT, R2             ; R2->error flags
172034 012701 176540                    MOV     #RCSR$2, R1             ; R1 -> SLU2
172040 016146 000002                    MOV     2(R1), -(SP)            ; ignore garbage, make temp
172044 012704 171750                    MOV     #INITS, R4              ; R4->initial XCSR value
172050 014461 000004            3$:     MOV     -(R4), 4(R1)            ; init XCSR
172054 001436                           BEQ     11$                     ; branch if done
172056 005742                           TST     -(R2)                   ; R2->next error flag
172060 012716 000010                    MOV     #8., (SP)               ; (SP)=baud rate counter
172064 012703 171750            4$:     MOV     #PATERN, R3             ; R3->patterns
172070 005005                   5$:     CLR     R5                      ; init timeout counter
172072 105761 000004            6$:     TSTB    4(R1)                   ; loop pattern around
172076 100402                           BMI     7$                      ; branch if ready
172100 077504                           SOB     R5, 6$                  ; else bump timeout counter
172102 000422                           BR      10$                     ; branch if timeout

172104 111361 000006            7$:     MOVB    (R3), 6(R1)
172110 005005                           CLR     R5                      ; initialize timeout counter
172112 105711                   8$:     TSTB    (R1)
172114 100402                           BMI     9$                      ; branch if ready
172116 077503                           SOB     R5, 8$                  ; else bump timeout counter
172120 000413                           BR      10$                     ; branch if timeout

172122 126113 000002            9$:     CMPB    2(R1), (R3)             ; come back OK?
172126 001010                           BNE     10$                     ; no, set error bit & exit
172130 105723                           TSTB    (R3)+                   ; done all bit patterns?
172132 001356                           BNE     5$                      ; no
172134 005316                           DEC     (SP)                    ; yes, done all bauds?
172136 001744                           BEQ     3$                      ; yes
172140 062761 000010 000004             ADD     #10, 4(R1)              ; no, to next baud rate
172146 000746                           BR      4$

172150 051200                   10$:    BIS     (R2),R0                 ; set error bit
172152 005726                   11$:    TST     (SP)+                   ; rid of temp
172154 004767 177472                    CALL    OCTSTO                  ; print error flags
172160 000167 176516                    JMP     KBD$                    ; and just get out.
                                        .DSABL  LSB
```

```
                                        .SBTTL BOOTS-Description

                        ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
                        ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
                        ;;;;                                                           ;;;;
                        ;;;;                    BOOTSTRAP MODULE                       ;;;;
                        ;;;;                                                           ;;;;
                        ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
                        ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

        000000                  .REPT   0

                        This is a short bootstrap program designed to handle floppy disks or TU58
                        tape cassettes in either our standard bootable format or in the stand-alone
                        volume format (RT-11 ".SAV"-structured files).

                        The bootstrap sequence is as follows:

                        1.  Since entry is effected by typing D in response to ODT prompt, get next
                            character (D, X or Y).  Get optional device number next (default is 0).

                        2.  If floppy boot is selected:

                            a.  Attempt to read 512 bytes from specified unit of the floppy
                                disk, starting from logical block zero, into memory locations
                                starting at 0 at the density of the medium present in the
                                drive at the time.

                            b.  If the drive is not ready or does not contain a bootable
                                medium, go back to ODT.

                        3.  If TU58 boot is selected, read the first block from the selected
                            drive into locations starting at 0.

                        4.  If the first byte read into RAM is 240 octal, jump to it.  If the
                            first byte is 260 octal, execute the stand-alone volume loader,
                            using the selected device as input.

                                .ENDR
```

```
                        ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
                        ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
                        ;;;;                                                     ;;;;
                        ;;;;           EQUATES USED ONLY BY BOOTSTRAPS           ;;;;
                        ;;;;                                                     ;;;;
                        ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
                        ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

                                      .SBTTL BOOTS-RX Controller Definitions

                        ; RX01/RX02 (RXV11,RXV21) Register Definitions

        177170          RXCS=   177170                          ;Control and Status
        177172          RXDB=   RXCS+2                          ;Data Buffer

                        ; RX Control and Status Bits

        100000          RX$$ER= 100000                          ;Error
        040000          RX$$IN= 040000                          ;Initialize controller
        030000          RX$$XA= 030000                          ;Extended address bits
        004000          RX$$02= 004000                          ;1 if RX02; 0 if RX01
        003000          RX$$XX= 003000                          ;Unused bits
        000400          RX$$DE= 000400                          ;Density (1=double,0=single)
        000200          RX$$TR= 000200                          ;Transfer function
        000100          RX$$IE= 000100                          ;Interrupt enable
        000040          RX$$DN= 000040                          ;Done
        000020          RX$$UN= 000020                          ;Unit select
        000016          RX$$FN= 000016                          ;Function select
        000001          RX$$GO= 000001                          ;GO

                        ; RX Function Codes (in RX$$FN) with GO bit preset

        000001          RX$FIL= 0*2+RX$$GO                      ;Fill buffer
        000003          RX$EMP= 1*2+RX$$GO                      ;Empty buffer
        000005          RX$WRT= 2*2+RX$$GO                      ;Write sector
        000007          RX$RED= 3*2+RX$$GO                      ;Read sector
        000011          RX$STD= 4*2+RX$$GO                      ;Set media density
        000013          RX$RST= 5*2+RX$$GO                      ;Read status
        000015          RX$WDD= 6*2+RX$$GO                      ;Write sector with deleted data
        000017          RX$REC= 7*2+RX$$GO                      ;Read error code

                        ; RX Error Codes

        000400          RXE$UN= 000400                          ;Unit selected
        000200          RXE$DR= 000200                          ;Drive ready
        000100          RXE$DD= 000100                          ;Deleted data
        000040          RXE$DN= 000040                          ;Drive density
        000020          RXE$DE= 000020                          ;Density error
        000004          RXE$ID= 000004                          ;Initialize done
        000001          RXE$CR= 000001                          ;CRC error

                        ; Miscellaneous Definitions

        000010          RETRY=  8.                              ;Number of retries

                                .SBTTL BOOTS-TU58 Definitions and Protocol Equates
```

```
                                ; Absolute address definitions

        000002                  FILNAM  = 000002                        ;Address of RAD50 filename for
                                                                        ; stand-alone program loading
        001000                  DIRBUF  = 001000                        ;Start of 512. word buffer used
                                                                        ; for RT-11 directory operations
                                                                        ;  in stand-alone loading

                                ; TU58 Address definitions

        176540                  TI$CSR  = RCSR$2                        ;DL receiver control and status
        176542                  TI$BFR  = RBUF$2                        ;DL receiver data buffer
        176544                  TO$CSR  = XCSR$2                        ;DL transmitter control and status
        176546                  TO$BFR  = XBUF$2                        ;DL transmitter data buffer


                                ; TU58 Radial Serial Protocol codes

                                ; Flag Byte Definitions:

        000001                  R$$DAT  = ^B<00001>                     ;Data message flag
        000002                  R$$CTL  = ^B<00010>                     ;Control message flag
        000004                  R$$INT  = ^B<00100>                     ;Initialize flag
        000020                  R$$CON  = ^B<10000>                     ;Continue flag
        000023                  R$$XOF  = ^B<10011>                     ;XOFF

                                ; Control packet operation codes:

        000000                  R$NOP   = 0.                            ;No-operation
        000001                  R$INIT  = 1.                            ;Initialize
        000002                  R$READ  = 2.                            ;Read operation
        000003                  R$WRIT  = 3.                            ;Write operation
        000004                  R$COMP  = 4.                            ;Compare (NOP on TU58)
        000005                  R$POSI  = 5.                            ;Position operation
        000006                  R$ABRT  = 6.                            ;Abort (NOP on TU58)
        000007                  R$DIAG  = 7.                            ;Diagnose
        000010                  R$GETS  = 8.                            ;Get status
        000011                  R$SETS  = 9.                            ;Set status (NOP on TU58)
        000012                  R$GETC  = 10.                           ;Get characteristics
        000013                  R$SETC  = 11.                           ;Set characteristics (NOP on TU58)
        000100                  R$END   = ^B<01000000>                  ;*END message

                                ; END packet success codes:

        000000                  S$NORM  = 0.                            ;Normal success
        000001                  S$RETR  = 1.                            ;Success but with retries
        177776                  S$PART  = -2.                           ;Partial operation (end of medium)
        177770                  S$UNIT  = -8.                           ;Invalid unit number
        177767                  S$CART  = -9.                           ;No cartridge
        177765                  S$WPRT  = -11.                          ;Cartridge write protected
        177757                  S$DCHK  = -17.                          ;Data check error
        177740                  S$SEEK  = -32.                          ;Seek error (block not found)
        177737                  S$MOTR  = -33.                          ;Motor stopped
        177720                  S$OPCD  = -48.                          ;Invalid operation code
        177711                  S$RECN  = -55.                          ;Invalid record number

                                        .SBTTL BOOTS-RT11 Definitions and Equates
```

```

                                ; RT-11 Directory Structure Definitions

        001000                  SEGALO  = DIRBUF                        ;Number of segments allocated
        001002                  NXTSEG  = DIRBUF+2                      ;Number of next logical segment
        001004                  HGHSEG  = DIRBUF+4                      ;Highest segment in use
        001006                  XTRBYT  = DIRBUF+6                      ;Number of extra bytes per entry
        001010                  STRBLK  = DIRBUF+10                     ;Starting blocks for files
                                                                        ; in this segment
        000016                  ENTSIZ  = 7*2                           ;Size of a directory entry
        000010                  D.FLEN  = 10                            ;Offset to file length in entry
        000400                  TENTA$  = 000400                        ;Flag for tentative file entry
        001000                  EMPTY$  = 001000                        ;Flag for empty area entry
        002000                  PERMF$  = 002000                        ;Flag for permanent file
        004000                  ENDSG$  = 004000                        ;Flag for end of segment

                                ; RT-11 System Communications Area Definitions

        000040                  RT$STA  = 000040                        ;Start address for program
        000042                  RT$ISP  = 000042                        ;Initial stack pointer
        000044                  RT$JSW  = 000044                        ;Job status word
        000046                  RT$USR  = 000046                        ;USR load address
        000050                  RT$HGH  = 000050                        ;Job high memory limit
        000052                  RT$EMT  = 000052                        ;(Byte) EMT error code
        000053                  RT$UER  = 000053                        ;(Byte) User error code
        000054                  RT$RMN  = 000054                        ;Base address of resident monitor
        000056                  RT$FCH  = 000056                        ;(Byte) Console fill character
        000057                  RT$FCT  = 000057                        ;(Byte) Console fill count
```

```
                                        .SBTTL BOOTS-Program entry point

                                ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
                                ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
                                ;;;;                                                    ;;;;
                                ;;;;  BOOTSTRAP INITIALIZATION AND COMMAND INTERPRETER  ;;;;
                                ;;;;                                                    ;;;;
                                ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
                                ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

                                ; A 'D' was entered in response to the ODT prompt, so we get
                                ; here and expect 'D','X' or 'Y' next, followed by a CR or a
                                ; unit number.  We set bits up in B.CNTL as follows:
                                ;
                                ;     BIT  7: 0 = TU58
                                ;             1 = RX01/02
                                ;             Used by stand-alone volume loader to select proper
                                ;             read routine.
                                ;     BIT  0: Device number
                                ;
                                ; Note: if no memory was found at 000000, bit 15 of B.CNTL,
                                ;   called "NO.LOW" will be set and the bootstraps will be
                                ;   disabled.

172164                          BOOTS::
172164  012737  000072  176544          MOV     #TUBAUD,@#TO$CSR        ;Set TU58 Baud Rate

                                ; Jump here with ODT if booting TU58's at other than default baud rate

172172                          STTUBD::
172172  010667  175566                  MOV     SP,IN.USR               ;Permit HALTs and BREAKS
                                                                        ; by making IN.USR non-zero
172176  005004                          CLR     R4                      ;Assemble new B.CNTL here
172200  004767  177304                  CALL    GETCHR                  ;Keyboard character in R2
172204  120227  000104                  CMPB    R2,#'D                  ;DD = TU58 cassette
172210  001412                          BEQ     1$                      ;R4 is clear for DD
172212  012704  000200                  MOV     #DEVBIT,R4              ;R4 bit 7 is set for DX, DY
172216  020227  000130                  CMP     R2,#'X                  ;DX = RX01 or RX02
172222  001405                          BEQ     1$                      ;DY = RX01 or RX02, the code's
172224  020227  000131                  CMP     R2,#'Y                  ; the same- it knows both den-
172230  001402                          BEQ     1$                      ;  sities, DMA, non-DMA
172232                                  ABORT   <Illegal device name>

172236  004767  177246          1$:     CALL    GETCHR                  ;Get device number or CR
172242  022702  000015                  CMP     #15,R2                  ;Is it CR?
172246  001410                          BEQ     3$                      ;CR means drive 0
172250  162702  000060                  SUB     #'0,R2                  ;Drive 0?
172254  001405                          BEQ     3$                      ;Yup.
172256  005302                          DEC     R2                      ;Drive 1?
172260  001402                          BEQ     2$                      ;Yes, skip the ABORT
172262                                  ABORT   <Illegal unit number>

172266  005204                  2$:     INC     R4                      ;For unit 1.
172270  110467  175474          3$:     MOVB    R4,B.CNTL               ;Set device, unit information.
172274  005767  175470                  TST     B.CNTL                  ;Test NO.LOW
172300  100002                          BPL     4$                      ;We have low memory
                                                                        ;We don't, so go to ODT
```

```
 172302                                           ABORT   <No low memory, can't boot>

                                         ; Before proceeding, we set up the bus timeout trap vector, enable
                                         ; trap to 4 emulation and reset the bus.  We do a delay (see
                                         ; explanation below) and set up the stack so the stand-alone booter and
                                         ; device primary bootstraps can get the information they need passed
                                         ; to them in R0 and R1 (see CHK240, below).

 172306  012737  172370  000004  4$:      MOV     #BADBOT,@#4              ;If we time out, we want to re-
 172314  012737  000300  000006           MOV     #PRI6,@#6                ;initialize everything.
 172322  000005                           RESET                            ;For now, init. the bus.

                                         ;        Note: the previous instruction also screws up some devices
                                         ;        which perform a long initialization sequence, such as RX02's,
                                         ;        which do an automatic boot from drive 0.  The long delay below
                                         ;        is necessary in order to assure drive 1 is ready if a boot
                                         ;        is desired from it.

 172324                                   DELAY   R0,R1,9.                 ;Delay 2 seconds
 172336  012706  167644                   MOV     #$$STACK,SP              ;Initialize the stack.
 172342  010667  175430                   MOV     SP,TRAP4                 ;Set up trap-to-4 emulation
                                                                           ;by making TRAP4 non-zero
 172346  012716  037776                   MOV     #37776,(SP)              ;Some boots need a memory-top
                                                                           ; address here, so 8k will do
 172352  010402                           MOV     R4,R2                    ;Boot control word here
 172354  042702  177776                   BIC     #^C<DEVNUM>,R2           ;Want only unit no. in R2
 172360  010246                           MOV     R2,-(SP)                 ;And we'll save it too.

 172362  105704                           TSTB    R4                       ;Bit 7 set for RX01/02
 172364  100405                           BMI     RXBOOT                   ;Go to floppy boot
 172366  000436                           BR      TUBOOT                   ;Go to TU58 boot

 172370                          BADBOT:
 172370  012706  167644                   MOV     #$$STACK,SP              ;Restore the stack
 172374                                   ABORT   <Unexpected timeout during boot>
```

```
                                  .SBTTL BOOTS-RX01/RX02 Bootstrap

                          ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
                          ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
                          ;;;;                                                                ;;;;
                          ;;;;                      FLOPPY BOOTSTRAP                          ;;;;
                          ;;;;                                                                ;;;;
                          ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
                          ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

                          ; This routine will bootstrap either floppy drive, at the density of the
                          ; media mounted in that drive.

172400                    RXBOOT:
172400  012746  177170            MOV     #RXCS,-(SP)             ;Need floppy CSR for CHK240
172404  005737  177170            TST     @#RXCS                  ;If not there, time out via 4
172410  000240                    NOP                             ;to ST173 and reset the world
172412  012701  001000            MOV     #512.,R1                ;Byte count
172416  005000                    CLR     R0                      ;Starting block number
172420  005004                    CLR     R4                      ;RAM buffer address = 000000
172422  004767  000402            CALL    DREAD                   ;LOAD IT ALL IN
```

```
                                                  .SBTTL BOOTS-Distinguishing type of boot block

                                    ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
                                    ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
                                    ;;;;                                                              ;;;;
                                    ;;;;     DISTINGUISH STANDARD FROM STAND-ALONE FROM               ;;;;
                                    ;;;;                NON-BOOTABLE VOLUMES.                         ;;;;
                                    ;;;;                                                              ;;;;
                                    ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
                                    ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

                                    ; The CHK240 routine will repeat powerup sequence if location 0 does not
                                    ; contain a valid secondary bootstrap (i.e., does not have a 240 or 260
                                    ; in it).  It starts execution of the booted program if there's a 240,
                                    ; and goes to the stand-alone program loader if there's a 260.

 172426                             CHK240:
 172426  022737  000240  000000              CMP     #240,@#0                ;Did we read a valid bootstrap?
 172434  001410                              BEQ     1$
 172436  022737  000260  000000              CMP     #260,@#0
 172444  001447                              BEQ     STANDB                  ;Stand-alone volumes start with 260
 172446  004767  175766                      CALL    VECSET                  ;Restore wiped-out vectors
 172452                                      ABORT   <No boot block on volume>

 172456  012601                     1$:      MOV     (SP)+,R1                ;Unit CSR address
 172460  012600                              MOV     (SP)+,R0                ;Unit number
 172462  005007                              CLR     PC                      ;Standard secondary boots
```

```
                                .SBTTL BOOTS-TU58 Bootstrap

                        ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
                        ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
                        ;;;;                                                    ;;;;
                        ;;;;            TU58 TAPE CASSETTE BOOTSTRAP            ;;;;
                        ;;;;                                                    ;;;;
                        ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
                        ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

                                .ENABL  LSB
172464                  TUBOOT:
172464  012746  176540          MOV     #TI$CSR,-(SP)           ;CHK240 wants TU58 CSR
172470  012701  176544          MOV     #TO$CSR,R1              ;R1 -> output CSR for TU58 serial line
172474  005003                  CLR     R3                      ;Set R3 = 0 (Two NULLs)
172476  005211                  INC     @R1                     ;Start transmitting BREAK to TU58
172500  004767  001132          CALL    CH$OUT                  ;Send eight NULLS
172504  105711          1$:     TSTB    @R1                     ;Is transmitter ready again yet?
172506  100376                  BPL     1$                      ;If PL no - wait
172510  042711  000001          BIC     #XC.BRK,@R1             ;Else stop sending BREAK now
172514  012703                  MOV     (PC)+,R3                ;Get two INIT commands for TU58
172516     004     004          .BYTE   R$$INT,R$$INT
172520  004715                  CALL    @R5                     ;And transmit them
172522  005741                  TST     -(R1)                   ;Dump any garbage char in TI$BUF
172524  105737  176540  2$:     TSTB    @#TI$CSR                ;Is character available from the TU58?
172530  100375                  BPL     2$                      ;If PL, no - wait in loop
172532  121127  000020          CMPB    @R1,#R$$CON             ;If so, was it a CONTINUE flag?
172536  001402                  BEQ     3$                      ;If EQ, yes- go ahead
172540                          ABORT   <TU58 initialization error>

                        ; TU58 is now initialized.  Prepare to read block #0.

172544  005000          3$:     CLR     R0                      ;Block number = 0
172546  012701  001000          MOV     #512.,R1                ;Byte count = one block
172552  004767  000212          CALL    READZU                  ;Attempt to read the block
172556  100323                  BPL     CHK240                  ;If PL, read was successful
172560                          ABORT   <TU58 block 0 read error>
                                .DSABL  LSB
```

```
                                        .SBTTL BOOTS-Stand-alone volume bootstrap

                                ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
                                ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
                                ;;;;                                                        ;;;;
                                ;;;;            STAND-ALONE-VOLUME BOOTSTRAP                ;;;;
                                ;;;;                                                        ;;;;
                                ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
                                ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

                                ; This routine loads stand-alone programs (assumed to be in RT-11 .SAV
                                ; file format) from an RT-11 file structured TU58 cartridge.  It is
                                ; invoked if the first word in block 0 of the cartridge is a 260.

172564                          STANDB:
172564  012700  000001                  MOV     #1,R0                   ;Set directory segment #1
172570  006300                  1$:     ASL     R0                      ;Two blocks per segment
172572  022020                          CMP     (R0)+,(R0)+             ;Add 4 to R0, as directory starts
                                                                        ; in block#6
172574  012701  002000                  MOV     #1024.,R1               ;Prepare to read two blocks
172600  012704  001000                  MOV     #DIRBUF,R4              ;Into the directory buffer
172604  004767  000162                  CALL    READU                   ;Read the segment
172610  100002                          BPL     2$                      ;If PL, read was successful
172612                                  ABORT   <Directory read error>
172616  012704  001010          2$:     MOV     #STRBLK,R4              ;Else prepare to pick up
                                                                        ; starting block
172622  012400                          MOV     (R4)+,R0                ;R0 = starting block for files
172624  010403                  3$:     MOV     R4,R3                   ;Save pointer to current entry
172626  032724  002000                  BIT     #PERMF$,(R4)+           ;Is this a permanent file?
172632  001010                          BNE     4$                      ;If bit set, yes - check if it matches
172634  022744  004000                  CMP     #ENDSGS,-(R4)           ;Else is this the end-of-segment
                                                                        ; marker?
172640  001015                          BNE     5$                      ;If NE, no - go skip this entry
172642  013700  001002                  MOV     @#NXTSEG,R0             ;Else get number of next segment
172646  001350                          BNE     1$                      ;If NE, there is one - go read it
172650                                  ABORT   <File not found>

172654  012705  000002          4$:     MOV     #FILNAM,R5              ;Point to RAD50 name of desired file
172660  022425                          CMP     (R4)+,(R5)+             ;Check file name, first word
172662  001004                          BNE     5$                      ;If NE not desired file
172664  022425                          CMP     (R4)+,(R5)+             ;...Check second word of filename
172666  001002                          BNE     5$                      ;If NE not desired one
172670  022425                          CMP     (R4)+,(R5)+             ;...Finally, check extension
172672  001410                          BEQ     LOAD                    ;If EQ, got it - go load this
                                                                        ; one into memory
172674  010304                  5$:     MOV     R3,R4                   ;Get entry pointer back
172676  062704  000010                  ADD     #D.FLEN,R4              ;Advance to file size of entry
172702  062400                          ADD     (R4)+,R0                ;Update current file base
172704  022424                          CMP     (R4)+,(R4)+             ;And skip to next file entry
172706  063704  001006                  ADD     @#XTRBYT,R4             ;Plus any extra bytes in each entry
172712  000744                          BR      3$                      ;Continue file search
```

```
                                                         .SBTTL BOOTS-Load Stand-Alone Program File

172714  011401                      LOAD:   MOV     @R4,R1                  ;R1 = size of file in blocks
172716  000301                              SWAB    R1                      ; * 256. = word count
172720  006301                              ASL     R1                      ; * 2 = byte count
172722  004767  000042                      CALL    READZU                  ;Read the program file into memory
172726  100002                              BPL     1$                      ;If MI, error in read-ABORT
172730                                      ABORT   <Stand-alone file read error>
172734  013705  000040              1$:     MOV     @#RT$STA,R5             ;Get program start adrs
172740  032705  000001                      BIT     #1,R5                   ;Is adrs even?
172744  001402                              BEQ     START$                  ;If EQ yes - okay
172746                                      ABORT   <Illegal transfer address>

172752                              START$:
172752  012601                              MOV     (SP)+,R1                ;Pass the CSR address
172754  112600                              MOVB    (SP)+,R0                ;Get unit number booted
172756  013706  000042                      MOV     @#RT$ISP,SP             ;Load program's stack pointer
172762  005067  175010                      CLR     TRAP4                   ;Disable trap to 4 feature
172766  000115                              JMP     @R5                     ;Go start program execution

172770                              READZU:
172770  005004                              CLR     R4                      ;Load at 0
172772  016602  000004              READU:  MOV     4(SP),R2                ;Get unit number
172776  000407                              BR      AROUN3                  ; SKIP OVER THE ENTRY POINT
```

```
                                                  .SBTTL 173000G ENTRY POINT

           173000                          .=173000
   173000                         ST173::
   173000  106427  000340                  MTPS    #PRI7              ;Can't assume anything here.
   173004  000005                          RESET                      ;But PWRSUP usually does.
   173006  005000                          CLR     R0                 ;DELAY for the sake of DLART.
   173010  077001                          SOB     R0,.               ;(Maint. bit cleared by RESET
                                                                      ;just a little too long).
   173012  000167  175242                  JMP     PWRSUP
```

```

  173016                                                      .SBTTL BOOTS-Continued
                                          AROUN3:
  173016  105767  174746                          TSTB    B.CNTL
  173022  100402                                  BMI     DKEAD                         ;Bit 7 set for RX01/RX02
  173024  000167  000370                          JMP     TREAD                         ;Read from tape
```

```
                                       .SBTTL BOOTS-RX01/RX02 Read routines

                               ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
                               ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
                               ;;;;                                                    ;;;;
                               ;;;;            FLOPPY DISK READ ROUTINES               ;;;;
                               ;;;;                                                    ;;;;
                               ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
                               ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

                               ; With registers set up as below, read the appropriate number of
                               ; full sectors from the floppy, at either density, with either
                               ; RXV21 DMA or RXV11 Programmed I/O interface.
                               ;
                               ; R0: Starting block number for transfer.
                               ; R1: Byte count for transfer
                               ; R2: Unit number
                               ; R4: Address of buffer to receive data

                               .ENABL LSB
173030  010446                 DREAD:  MOV     R4,-(SP)                ;Save buffer address
173032  010046                         MOV     R0,-(SP)                ;Save starting LBN
173034  010146                         MOV     R1,-(SP)                ;Save byte count

                               ; Check status and media density of selected drive

173036  012701  177172                 MOV     #RXDB,R1                ;Set up R1 for benefit of RXGO
173042  005000                         CLR     R0                      ;Initialize current unit/density word
173044  006002                         ROR     R2                      ;Bit 0 set = unit 1
173046  103002                         BCC     1$
173050  052700  000020                 BIS     #RX$$UN,R0              ;Set unit 1
173054  004567  000312         1$:     JSR     R5,RXGO                 ;Start a read status operation
173060  000013                         .WORD   RX$RST                  ; to determine status and density
173062  111102                         MOVB    @R1,R2                  ;Pick up low byte of status
173064  100402                         BMI     2$                      ;If PL, drive not ready
173066                                 ABORT   <Floppy drive not ready>
173072  032702  000040         2$:     BIT     #RXE$DN,R2              ;Check media density
173076  001411                         BEQ     3$                      ;If EQ, single density

                               ; Double density.
                               ; Logical sector number = logical block number * 2
                               ; Sector count = byte count/256.

173100  052700  000400                 BIS     #RX$$DE,R0              ;Set double density in command
173104  012602                         MOV     (SP)+,R2                ;Byte count
173106  000302                         SWAB    R2                      ;Divide by 256
173110  012603                         MOV     (SP)+,R3                ;LBN
173112  006303                         ASL     R3                      ;Multiply by 2
173114  012704  000200                 MOV     #128.,R4                ;Words per sector
173120  000410                         BR      4$

                               ; Single density.
                               ; Logical sector number = logical block number * 4
                               ; Sector count = byte count/128.

173122  012602                 3$:     MOV     (SP)+,R2                ;Byte count
173124  000302                         SWAB    R2                      ;Divide by 256
```

```
173126  006302                      ASL     R2                      ;And multiply by 2
173130  012603                      MOV     (SP)+,R3                ;LBN
173132  006303                      ASL     R3
173134  006303                      ASL     R3                      ;Multiply by 4
173136  012704  000100              MOV     #64.,R4                 ;Words per sector

                            ; Set up stack as follows:
                            ;   0(SP) = Logical Sector Number
                            ;   2(SP) = Sector count
                            ;   4(SP) = Words per sector
                            ;   6(SP) = Buffer address

173142  010446              4$:     MOV     R4,-(SP)                ;Words per sector
173144  010246                      MOV     R2,-(SP)                ;Sector count
173146  010346                      MOV     R3,-(SP)                ;Logical Sector Number

                            ; Start the read operation.
                            ; This is the top of the loop.

173150  004567  000216      5$:     JSR     R5,RXGO                 ;Start a sector read
173154  000007                      .WORD   RX$RED

                            ; Convert Logical Sector Numbers to Physical tracks and sectors.

173156  011603                      MOV     @SP,R3                  ;Get Logical Sector Number
173160  012702  000010              MOV     #8.,R2                  ;Loop count
173164  022703  006400      6$:     CMP     #26.*200,R3             ;Does 26 go into dividend?
173170  101002                      BHI     7$                      ;Branch if not, C clear (BHI => BCC)
173172  062703  171400              ADD     #-26.*200,R3            ;Subtract 26 from dividend (C set)
173176  006103              7$:     ROL     R3                      ;Shift dividend and quotient
173200  005302                      DEC     R2                      ;Decrement loop count
173202  003370                      BGT     6$                      ;Branch till divide done
173204  110302                      MOVB    R3,R2                   ;Copy track number
173206  105003                      CLRB    R3                      ;Remove track number from remainder
173210  000303                      SWAB    R3                      ;Get remainder
173212  022703  000014              CMP     #12.,R3                 ;C=1 if 13<=R3<=25, else C=0
173216  006103                      ROL     R3                      ;Sector*2 (2:1 interleave)
                                                                    ;[+1 (C) if sector 13-25]
173220  006302                      ASL     R2                      ;Double the track number
173222  060203                      ADD     R2,R3                   ;Skew the sector
173224  060203                      ADD     R2,R3                   ; by adding in
173226  060203                      ADD     R2,R3                   ;  6 * track number
173230  006202                      ASR     R2                      ;Undouble the track number
173232  005202                      INC     R2                      ; and make it 1-76 (Skip track 0
                                                                    ;  for ANSI)
173234  162703  000032      8$:     SUB     #26.,R3                 ;Put sector
173240  002375                      BGE     8$                      ; into range
173242  062703  000033              ADD     #27.,R3                 ;  1-26

                            ; Read the sector

173246  010311                      MOV     R3,@R1                  ;Set sector number
173250  004514                      JSR     R5,@R4                  ;
173252  010211                      MOV     R2,@R1                  ;Set track number
173254  004514                      JSR     R5,@R4                  ;Perform a sector read
173256  100002                      BPL     9$                      ;If MI, error
173260                              ABORT   <Floppy read error>
```

```
                                   ; Empty RXV11/RXV21 buffer into RAM

173264  004567  000102             9$:     JSR     R5,RXGO              ;Start empty buffer function
173270  000003                             .WORD   RX$EMP               ; and wait for TR
173272  032737  004000  177170             BIT     #RX$$02,@#RXCS       ;Is DMA available?
173300  001407                             BEQ     10$                  ;If EQ no - handle as RX01

                                   ; RX02 DMA Operation

173302  016611  000004                     MOV     4(SP),@R1            ;Else load word count
173306  004514                             JSR     R5,@R4               ;Wait for TR
173310  016611  000006                     MOV     6(SP),@R1            ;And load current bus address
173314  004514                             JSR     R5,@R4               ;Wait for DONE
173316  000410                             BR      12$

                                   ; RX01 Programmed I/O Operation

173320  016603  000004             10$:    MOV     4(SP),R3             ;Get word count
173324  006303                             ASL     R3                   ;Turn word count into byte count
173326  016602  000006                     MOV     6(SP),R2             ;Get starting bus address
173332  111122                     11$:    MOVB    @R1,(R2)+            ;Move one byte from buffer to memory
173334  004514                             JSR     R5,@R4               ;Wait for TR or DONE
173336  077303                             SOB     R3,11$               ;Loop for all bytes in first sector

                                   ; Loop back if not yet finished

173340  016603  000004             12$:    MOV     4(SP),R3             ;Get word count
173344  006303                             ASL     R3                   ;Turn into byte count
173346  060366  000006                     ADD     R3,6(SP)             ;Update bus address
173352  005216                             INC     @SP                  ;Update Logical Sector Number
173354  005366  000002                     DEC     2(SP)                ;Decrement Sector Count
173360  001273                             BNE     5$                   ;Read another sector
173362  062706  000010                     ADD     #8.,SP               ;Pop the stack
173366  000257                             CCC                          ;Clear condition codes
                                                                        ; to show success.
173370  000207                             RETURN                       ;All done
                                   .DSABL LSB
```

```
                                ; The main subroutine for sending disk commands and waiting for
                                ; their completion.
                                ;
                                ; Register usage:
                                ;       R0 = density bit ! unit select bit (proto for commands)
                                ;       R1 = RXDB address
                                ;       R4 = RXGO TR/DONE test routine pointer
                                ;

173372  012504                  RXGO:   MOV     (R5)+,R4                ;Copy command word to use
173374  050004                          BIS     R0,R4                   ;Set unit # and density
173376  010437  177170                  MOV     R4,@#RXCS               ;Start operation
173402  010704                          MOV     PC,R4                   ;Copy adrs for later calls
173404  005741                          TST     -(R1)                   ;R1 -> RXCS
173406  032711  000240          1$:     BIT     #RXS$TR!RXS$DN,@R1      ;Wait for TR or DONE
173412  001775                          BEQ     1$                      ;If EQ, neither are true yet
173414  005721                          TST     (R1)+                   ;Reset R1 -> RXDB and check for errors
173416  000205                          RTS     R5                      ;Return to caller

```

```
                                                .SBTTL BOOTS-TU58 Read routines

                                ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
                                ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
                                ;;;;                                                         ;;;;
                                ;;;;            TU58 DECtape II READ ROUTINES                ;;;;
                                ;;;;                                                         ;;;;
                                ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
                                ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

                                ; Starts a read operation on the TU58 by transmitting a command packet
                                ;
                                ; Inputs:
                                ;       R0 = starting block # for transfer
                                ;       R1 = byte count for transfer
                                ;       R2 = unit number
                                ;       R4 = address of buffer to receive data
                                ; Outputs:
                                ;       R0, R1, R2 unchanged
                                ; Destroys:
                                ;       R3, R4, R5

                                .ENABL LSB
173420  010446                  TREAD:  MOV     R4,-(SP)                ;Save buffer address
173422  005004                          CLR     R4                      ;Init checksum
173424  012703  005002                  MOV     #10.*400+R$$CTL,R3      ;Set command flag and length
173430  004767  000206                  CALL    CH2OUT                  ;Output two chars and set R5
173434  012703  000002                  MOV     #R$READ,R3              ;Send read command and modifier=0
173440  004715                          CALL    @R5
173442  010203                          MOV     R2,R3                   ;Then unit number and switches=0
173444  004715                          CALL    @R5
173446  005003                          CLR     R3                      ;Plus a zero sequence number
173450  004715                          CALL    @R5
173452  010103                          MOV     R1,R3                   ;Followed by the byte count
173454  004715                          CALL    @R5
173456  010003                          MOV     R0,R3                   ;And the block number
173460  004715                          CALL    @R5
173462  010403                          MOV     R4,R3                   ;Finally, transmit the checksum
173464  004715                          CALL    @R5
```

```
                                        ; Now ready to accept data messages from the TU58

  173466  012600                                MOV     (SP)+,R0                ;R0 -> data buffer
                                        ;       CLC                             ;(CH2OUT leaves C clear)
  173470  006001                                ROR     R1                      ;R1 = word count for transfer
  173472  004767  000116                1$:     CALL    7$                      ;Get first word of packet
  173476  122703  000001                        CMPB    #R$$DAT,R3              ;Is this indeed a data message?
  173502  001017                                BNE     3$                      ;If NE no - may be END message
  173504  105003                                CLRB    R3                      ;Else clear flags
  173506  000303                                SWAB    R3                      ;Move packet byte count to low byte
  173510  106003                                RORB    R3                      ;And convert to word count
  173512  160301                                SUB     R3,R1                   ;Remove from transfer count
  173514  010305                                MOV     R3,R5                   ;And copy for loop counter
  173516  004767  000102                2$:     CALL    9$                      ;Get next two words
  173522  010320                                MOV     R3,(R0)+                ;Store in buffer
  173524  077504                                SOB     R5,2$                   ;Loop for entire data message
  173526  004767  000044                        CALL    5$                      ;Get checksum and compare
  173532  005701                                TST     R1                      ;Have all data records been
                                                                                ; transferred?
  173534  001356                                BNE     1$                      ;If NE no
  173536  004767  000052                        CALL    7$                      ;And get prospective
                                                                                ; END packet start
  173542  004767  000056                3$:     CALL    9$                      ;Get opcode/success bytes
                                                                                ; of END packet
  173546  122703  000100                        CMPB    #R$END,R3               ;Is this an END packet?
  173552  001402                                BEQ     4$                      ;If NE no - ABORT
  173554                                        ABORT   <TU58 END packet missing>
  173560  010300                        4$:     MOV     R3,R0                   ;Save success code in R0
  173562  004767  000032                        CALL    8$                      ;Read remainder of END packet
  173566  004767  000004                        CALL    5$                      ;And check its checksum
  173572  000300                                SWAB    R0                      ;Set CC's on success code of transfer
  173574  000207                                RETURN                          ;Return to caller

  173576  004767  000064                5$:     CALL    CH2IN                   ;Get two checksum bytes
  173602  020403                                CMP     R4,R3                   ;Does it match calculated value?
  173604  001402                                BEQ     6$                      ;If NE no - ERROR
  173606                                        ABORT   <TU58 checksum error>

  173612  000207                        6$:     RETURN                          ;Else return with success

  173614  005004                        7$:     CLR     R4                      ;Init checksum
  173616  000402                                BR      9$                      ;And get the first word

  173620  004717                        8$:     CALL    @PC                     ;Read 4 words
  173622  004717                                CALL    @PC
  173624  004767  000036                9$:     CALL    CH2IN                   ;Read next two bytes
  173630  060304                                ADD     R3,R4                   ;Add into checksum
  173632  005504                                ADC     R4                      ; with end-around carry
  173634  000207                                RETURN                          ;And back to caller
                                        .DSABL LSB
```

```
                                        ; CH2OUT -- Write two bytes to the TU58
                                        ;
                                        ; Writes two bytes to interface and updates checksum.
                                        ;
                                        ; Inputs:
                                        ;       R3 = two bytes to be output; low byte first
                                        ;       R4 = current checksum word
                                        ; Outputs:
                                        ;       R3 unchanged
                                        ;       R4 updated to new checksum
                                        ;       R5 pointing to CH2OUT routine for easier future CALLs

 173636                                 CH8OUT:
 173636  004717                                 CALL    @PC                     ;Entry point to output 8 characters
 173640  004717                                 CALL    @PC
 173642                                 CH2OUT:
 173642  010705                                 MOV     PC,R5                   ;Set R5 to following routine adrs
 173644  060304                                 ADD     R3,R4                   ;Update checksum word
 173646  005504                                 ADC     R4                      ; with end-around carry
 173650  004717                                 CALL    @PC                     ;Repeat for both characters
 173652  105737  176544                 1$:     TSTB    @#TO$CSR                ;Is interface ready for output?
 173656  100375                                 BPL     1$                      ;If PL no - wait
 173660  110337  176546                         MOVB    R3,@#TO$BFR             ;Else transmit character to TU58
 173664  000407                                 BR      CHRET                   ;Merge with other routine to return

                                        ; CH2IN -- Read two bytes from the TU58
                                        ; CHIN  -- Read a single byte from the TU58
                                        ;
                                        ; Inputs:
                                        ;       none.
                                        ; Outputs:
                                        ;       R3 = character(s) read

 173666  004717                         CH2IN:  CALL    @PC                     ;Read two, not one
 173670  105003                         CHIN:   CLRB    R3                      ;And zero out space for new one
 173672  105737  176540                 1$:     TSTB    @#TI$CSR                ;Is a character available?
 173676  100375                                 BPL     1$                      ;If PL no
 173700  153703  176542                         BISB    @#TI$BFR,R3             ;Else set into register
 173704  000303                         CHRET:  SWAB    R3                      ;Move current character over
 173706  000207                                 RETURN                          ;And return to caller
```

```
      1                                              .SBTTL END STATEMENT
      2        000001              .END
```

SYMBOL TABLE

```
AROUN2   172010      E.PAR  = 000001      PP.BI6= 000014      RT$USR= 000046      R.PC   = 167766 G
AROUN3   173016      FAKOUT   170424      PP.BI7= 000016      RXBOOT   172400     R.STAK= 000001
AUTOBA   170472 G    FILNAM= 000002       PP.C   = 176204     RXCS   = 177170     R.TYPE= 167762 G
BADBOT   172370      GETCHR   171510      PP.CHI= 000010      RXDB   = 177172     R.$TRT   170036 G
BAUDR$= 000032       GETNUM   171612      PP.CLO= 000001      RXE$CR= 000001      SAVPC  = 167752 G
BD.003= 000000       HGHSEG= 001004       PP.CWR= 176206      RXE$DD= 000100      SAVPS  = 167754 G
BD.006= 000010       HKBDQ    171140      PP.DRA= 000020      RXE$DE= 000020      SEGALO= 001000
BD.012= 000020       HKBD$    171142      PP.DRB= 000002      RXE$DN= 000040      SPACE  = 000040
BD.024= 000030       HVBAUD   170556      PP.MDA= 000040      RXE$DR= 000200      SRET     171636
BD.048= 000040       INBYTE   170550      PP.MDB= 000004      RXE$ID= 000004      STANDB   172564
BD.096= 000050       INBYT$   170556      PP.MD2= 000100      RXE$UN= 000400      START    172000 G
BD.192= 000060       INITS    171750      PP.MOD= 000200      RXGO     173372     START$   172752
BD.384= 000070       IN.USR= 167764 G     PRINT    170706     RX$EMP= 000003      STRBLK= 001010
BIT0   = 000001      KBDQ     170674      PRI6   = 000300     RX$FIL= 000001      STTUBD   172172 G
BIT1   = 000002      KBD$     170702      PRI7   = 000340     RX$REC= 000017      ST173    173000 G
BIT10  = 002000      LCSET    171246      PUTCHR   171522     RX$RED= 000007      SWCMD    171212
BIT11  = 004000      LEDOFF= 000017       PUTCLF   171554     RX$RST= 000013      S$CART= 177767
BIT12  = 010000      LF     = 000012      PUTLF    171564     RX$STD= 000011      S$DCHK= 177757
BIT13  = 020000      LOAD     172714      PUTSTR   171542     RX$WDD= 000015      S$MOTR= 177737
BIT14  = 040000      LOCDSP   171252      PWR$UP   170260 G   RX$WRT= 000005      S$NORM= 000000
BIT15  = 100000      MODE   = 000221      QODT     170666     RX$$DE= 000400      S$OPCD= 177720
BIT2   = 000004      MSGQ     171730      RAMBOT= 160010      RX$$DN= 000040      S$PART= 177776
BIT3   = 000010      MSG$     171731      RAMTOP= 167776      RX$$ER= 100000      S$RECN= 177711
BIT4   = 000020      NEXNUM   171600      RBUF$1= 177562      RX$$FN= 000016      S$RETR= 000001
BIT5   = 000040      NOCT     171642      RBUF$2= 176542      RX$$GO= 000001      S$SEEK= 177740
BIT6   = 000100      NO.LOW= 100000       RB.BRK= 004000      RX$$IE= 000100      S$UNIT= 177770
BIT7   = 000200      NXTSEG= 001002       RB.ERR= 100000      RX$$IN= 040000      S$WPRT= 177765
BIT8   = 000400      OCTSTR   171656      RB.FRM= 020000      RX$$TR= 000200      TENTA$= 000400
BIT9   = 001000      OCTSTO   171652      RB.OVR= 040000      RX$$UN= 000020      TI$BFR= 176542
BOOTS    172164 G    ODT      170602 G    RCMD     171144     RX$$XA= 030000      TI$CSR= 176540
BRKNOO   170010 G    ODTFLG= 167750 G     RCMD1    171232     RX$$XX= 003000      TO$BFR= 176546
B.CNTL= 167770 G     ODTLOC= 167746 G     RCSR$1= 177560      RX$$02= 004000      TO$CSR= 176544
CHIN     173670      ODTSTK= 167744 G     RCSR$2= 176540      R$ABRT= 000006      TRAP4  = 167776 G
CHK240   172426      ODTWHY= 167774 G     RC.ACT= 004000      R$COMP= 000004      TREAD    173420
CHRET    173704      ONENUM   171576      RC.DUN= 000200      R$DIAG= 000007      TUBAUD= 000072
CH2IN    173666      O.CNTL= 167772 G     RC.IEN= 000100      R$END  = 000100     TUBOOT   172464
CH2OUT   173642      PATERN   171750      READU    172772     R$GETC= 000012      T.BIT  = 000020
CH8OUT   173636      PBR0   = 000010      READZU   172770     R$GETS= 000010      USERSP= 167760 G
CR     = 000015      PBR1   = 000020      REGOUT   171240     R$INIT= 000001      VECSET   170440 G
DEVBIT= 000200       PBR2   = 000040      RESTAR   172004 G   R$NOP  = 000000     XBUF$1= 177566
DEVNUM= 000001       PCMD     171044      RETRY  = 000010     R$POSI= 000005      XBUF$2= 176546
DIAGNO   171754      PERMF$= 002000       RFLAG  = 000200     R$READ= 000002      XCSR$1= 177564
DIRBUF= 001000       PP.A   = 176200      RPOINT= 167756 G    R$SETC= 000013      XCSR$2= 176544
DONE     171574      PP.B   = 176202      RT$EMT= 000052      R$SETS= 000011      XC.BRK= 000001
DREAD    173030      PP.BIC= 000000       RT$FCH= 000056      R$WRIT= 000003      XC.IEN= 000100
D.FLEN= 000010       PP.BIS= 000001       RT$FCT= 000057      R$$CON= 000020      XC.MNT= 000004
EMPTY$= 001000       PP.BI0= 000000       RT$HGH= 000050      R$$CTL= 000002      XC.PBE= 000002
ENDSG$= 004000       PP.BI1= 000002       RT$ISP= 000042      R$$DAT= 000001      XC.RDY= 000200
ENTSIZ= 000016       PP.BI2= 000004       RT$JSW= 000044      R$$INT= 000004      XTRBIT= 001006
ERRBIT   171742      PP.BI3= 000006       RT$RMN= 000054      R$$XOF= 000023      $STACK= 167644 G
E.EXT  = 000100      PP.BI4= 000010       RT$STA= 000040      R.HALT= 100000      $$$BRK   170000 G
E.INT  = 000010      PP.BI5= 000012       RT$UER= 000053      R.NXM  = 000200     $$$LTC   170006 G


. ABS.   174000        000
         000000        001
ERRORS DETECTED:  0
```

```
VIRTUAL MEMORY USED:  9216 WORDS  ( 36 PAGES)
DYNAMIC MEMORY AVAILABLE FOR  46 PAGES
,FALCON/C=FALCON
```

CROSS REFERENCE TABLE (CREF V04.00)

```
$$$BRK   14-16#   27-35
$$$LTC   14-20#   27-37
$$TACK   13-30#   24-13    27-17    49-77    49-91
AROUN2   44-47    46-2#
AROUN3   54-24    56-2#
AUTOBA   27-13    28-43#
B.CNTL   13-14#   26-49*   49-54*   49-55    56-3
BADBOT   49-66    49-90#
BAUDR$    9-14#   28-44
BD.003    7-21#
BD.006    7-22#
BD.012    7-23#
BD.024    7-24#    9-14
BD.048    7-25#
BD.096    7-26#
BD.192    7-27#
BD.384    7-28#    9-18
BIT0      5-5#     7-43     8-29     8-47    10-5     10-13    35-50
BIT1      5-6#     7-36     8-27     8-38     8-40     8-42     8-44
BIT10     5-15#
BIT11     5-16#    6-16     6-37
BIT12     5-17#
BIT13     5-18#    6-35
BIT14     5-19#    6-32
BIT15     5-20#    6-30    10-3     10-9
BIT2      5-7#     7-30     8-25     8-38     8-39     8-42     8-43
BIT3      5-8#     7-15     8-23     8-38     8-39     8-40     8-41
BIT4      5-9#     7-16     8-21     9-34
BIT5      5-10#    7-17     8-18    38-11
BIT6      5-11#    6-23     7-8      8-17    38-9
BIT7      5-12#    6-19     7-3      8-15     9-32    10-4     10-11
BIT8      5-13#
BIT9      5-14#
BOOTS    33-21    49-25#
BRKNOO   14-18    14-26#   21-17
CH2IN    61-34    61-46    62-34#
CH2OUT   60-27    62-16#
CH8OUT   52-17    62-13#
CHIN     62-35#
CHK240   51-17#   52-36
CHRET    62-24    62-39#
CR        5-25#   37-26    38-21    40-24    41-20    43-13
D.FLEN   48-125#  53-47
DEVBIT   10-11#   49-37
DEVNUM   10-13#   49-83
DIAGNO   33-25    44-37#
DIRBUF   48-62#   48-118   48-119   48-120   48-121   48-122   53-21
DONE     40-17    40-31#
DREAD    50-21    56-4     57-21#
E.EXT    10-17#   44-17
E.INT    10-18#   44-18
E.PAR    10-19#   46-6
EMPTY$   48-127#
ENDSG$   48-129#  53-31
ENTSIZ   48-124#
ERRBIT   44-19#   46-11
```

```
FAKOUT  27-12     27-15#
FILNAM  48-60#    53-38
GETCHR  33-18     36-30     37-25     39-15#    39-17     41-19     49-34     49-44
GETNUM  33-36     37-33     41-22#
HGHSEG  48-120#
HKBD$   35-57#    37-27     38-15
HKBDQ   35-56#    37-21     37-31     37-37     37-39     38-18     38-20
HVBAUD  29-10     29-31#
IN.USR  13-16#    14-17     14-31*    19-36*    20-31     21-11*    22-13*    27-16*    35-52*    49-31*
INBYT$  29-11     29-27#
INBYTE  29-8      29-20#    29-32
INITS   44-27#    46-14
KBD$    32-44     32-51#    35-57     46-45
KBDQ    32-48#    33-32     33-34     33-37     33-45     35-37     35-56     36-16     36-20     36-22     36-32
LCSET   33-43     37-17#
LEDOFF   9-10#    26-27     44-39
LF       5-24#    37-36     38-13     38-17     40-29     43-13
LOAD    53-44     54-3#
LOCDSP  36-39     37-18#    38-28
MODE     9-5#     24-20     44-38
MSG$    32-52     43-13#
MSGQ    32-49     43-12#
NEXNUM  41-18#    41-30
NO.LOW  10-9#     26-49
NOCT    41-25     41-35#
NXTSEG  48-119#   53-34
O.CNTL  13-10#    37-46     38-5
OCTSTO  32-40     42-14#    46-44
OCTSTR  37-22     38-25     42-15#
ODT     14-32     22-19     27-18     32-11#
ODTFLG  13-26#    32-57*    36-14*    37-38     38-19
ODTLOC  13-27#    36-38*    37-17*    37-45     38-12*    38-23*    38-24
ODTSTK  13-29#    13-30     32-21
ODTWHY  13-7#     32-16*    35-35*
ONENUM  36-15     41-16#
PATERN  44-33#    46-19
PBR0     7-15#     7-22      7-24      7-26      7-28
PBR1     7-16#     7-23      7-24      7-27      7-28
PBR2     7-17#     7-25      7-26      7-27      7-28
PCMD    33-28     35-22#
PERMF$  48-128#   53-29
PP.A     8-6#     46-4
PP.B     8-7#     46-3*
PP.BI0   8-45#
PP.BI1   8-44#
PP.BI2   8-43#
PP.BI3   8-42#
PP.BI4   8-41#
PP.BI5   8-40#
PP.BI6   8-39#
PP.BI7   8-38#     9-10
PP.BIC   8-48#
PP.BIS   8-47#     9-10
PP.C     8-8#
PP.CHI   8-23#
PP.CLO   8-29#     9-5
```

```
PP.CWR   8-5#    24-20*   26-27*   44-38*   44-39*
PP.DRA   8-21#    9-5
PP.DRB   8-27#
PP.MD2   8-17#
PP.MDA   8-18#
PP.MDB   8-25#
PP.MOD   8-15#    9-5
PRI6     9-27#   27-11    32-55    49-67
PRI7     9-28#   27-36    27-38    35-48    55-5
PRINT   32-50    32-54#
PUTCHR  37-24    38-22    38-27    39-19#   39-21    40-18    40-25    40-30    42-22
PUTCLF  40-23#   42-14
PUTLF   40-29#
PUTSTR  32-56    40-15#   40-19
PWR$UP  24-12#   45-5     55-10
QODT    32-37    32-41#
R$$CON  48-81#   52-27
R$$CTL  48-79#   60-26
R$$DAT  48-78#   61-7
R$$INT  48-80#   52-22    52-22
R$$XOF  48-82#
R$ABRT  48-92#
R$COMP  48-90#
R$DIAG  48-93#
R$END   48-98#   61-25
R$GETC  48-96#
R$GETS  48-94#
R$INIT  48-87#
R$NOP   48-86#
R$POSI  48-91#
R$READ  48-88#   60-28
R$SETC  48-97#
R$SETS  48-95#
R$WRIT  48-89#
R.$TRT  19-11#   45-7
R.HALT  10-3#    14-29    21-46
R.NXM   10-4#    19-34    22-18    35-35
R.PC    13-15#   21-35*   21-39*   21-41
R.STAK  10-5#    20-11    35-35
R.TYPE  13-18#   14-29*   19-32    19-34*   20-11*   20-17*   21-46*   22-18*   32-16    32-36    32-42    32-54*
RAMBOT   9-22#   25-19
RAMTOP   9-23#   25-24
RB.BRK   6-37#
RB.ERR   6-30#
RB.FRM   6-35#
RB.OVR   6-32#
RBUF$1   6-6#    24-35    29-3     29-7     32-12    39-18
RBUF$2   6-10#   48-69
RC.ACT   6-16#
RC.DUN   6-19#   24-36
RC.IEN   6-23#   24-36
RCMD    33-30    36-13#
RCMD1   36-23    36-36#
RCSR$1   6-5#    24-36    29-5     39-16
RCSR$2   6-9#    46-12    48-68
READU   53-22    54-23#
```

```
READZU  52-35    54-6     54-21#
REGOUT  36-25    36-34    36-38#
RESTAR  21-27    45-6#
RETRY   48-54#
RFLAG    9-32#   36-14
RPOINT  13-22#   32-32*   36-37
RT$EMT  48-138#
RT$FCH  48-141#
RT$FCT  48-142#
RT$HGH  48-137#
RT$ISP  48-134#  54-17
RT$JSW  48-135#
RT$RMN  48-140#
RT$STA  48-133#  54-9
RT$UER  48-139#
RT$USR  48-136#
RX$$02  48-21#   58-5
RX$$DE  48-23#   57-44
RX$$DN  48-26#   59-15
RX$$ER  48-18#
RX$$FN  48-28#
RX$$GO  48-29#   48-33    48-34    48-35    48-36    48-37    48-38    48-39    48-40
RX$$IE  48-25#
RX$$IN  48-19#
RX$$TR  48-24#   59-15
RX$$UN  48-27#   57-31
RX$$XA  48-20#
RX$$XX  48-22#
RX$EMP  48-34#   58-4
RX$FIL  48-33#
RX$REC  48-40#
RX$RED  48-36#   57-78
RX$RST  48-38#   57-33
RX$STD  48-37#
RX$WDD  48-39#
RX$WRT  48-35#
RXBOOT  49-87    50-14#
RXCS    48-13#   48-14    50-15    50-16    58-5     59-12*
RXDB    48-14#   57-27
RXE$CR  48-50#
RXE$DD  48-46#
RXE$DE  48-48#
RXE$DN  48-47#   57-37
RXE$DR  48-45#
RXE$ID  48-49#
RXE$UN  48-44#
RXGO    57-32    57-77    58-3     59-10#
S$CART  48-106#
S$DCHK  48-108#
S$MOTR  48-110#
S$NORM  48-102#
S$OPCD  48-111#
S$PART  48-104#
S$RECN  48-112#
S$RETR  48-103#
S$SEEK  48-109#
```

```
S$UNIT   48-105#
S$WPRT   48-107#
SAVPC    13-25#   14-27*   21-37*   27-12*   32-39    35-11*   35-54    36-24
SAVPS    13-24#   14-28*   21-36*   27-11*   35-16*   35-53    36-33    37-45
SEGALO   48-118#
SPACE     5-26#   37-23
SRET     41-21    41-32#
ST173    55-4#
STANDB   51-21    53-15#
START    45-4#
START$   54-11    54-14#
STRBLK   48-122#  53-25
STTUBD   49-30#
SWCMD    36-18    36-29#
T.BIT     9-34#   37-50
TENTA$   48-126#
TI$BFR   48-69#   62-38
TI$CSR   48-68#   52-13    52-25    62-36
TO$BFR   48-71#   62-23*
TO$CSR   48-70#   49-26*   52-14    62-21
TRAP4    13-5#    22-11    49-78*   54-18*
TREAD    56-5     60-24#
TUBAUD    9-18#   49-26
TUBOOT   49-8#    52-12#
USERSP   13-20#   32-20*   32-25
VECSET   26-46    27-34#   51-22
XBUF$1    6-8#    39-22*
XBUF$2    6-12#   48-71
XC.BRK    7-43#   52-20
XC.IEN    7-8#
XC.MNT    7-30#   44-26
XC.PBE    7-36#    9-14     9-18    44-25    44-26
XC.RDY    7-3#    24-39
XCSR$1    6-7#    24-32*   24-39    28-44*   29-37*   39-20
XCSR$2    6-11#   48-70
XTRBYT   48-121#  53-50
```

```
ABORT   11-15#   49-42    49-51    49-58    49-92    51-23    52-29    52-37    53-24    53-36    54-8     54-12    57-36    57-114
        61-27    61-37
DELAY   12-23#   49-76
```

APPENDIX E
MACRO-ODT ROM LISTING
FOR KXT11-A5 OPTION

Appendix E provides the user with the program listing of the KXT11-A5 Macro-ODT firmware option.

TABLE OF CONTENTS


```
     3-   13    COPYRIGHT NOTICE
     4-   35    KXT11-A2 EDIT HISTORY
     5-   40    EQUATES
     6-   68    GENERAL DLART EQUATES
     8-  154    GENERAL PPI EQUATES
     9-  203    PROGRAM-SPECIFIC EQUATES
    11-  256    MACRO DEFINITIONS
    13-  310    RAM DEFINITION
    14-  342    MSCP BUFFER AREA
    15-  364    TRAPS-TRAP-HANDLING ROUTINES
    15-  365    TRAPS-LTC TRAP-KILLER
    15-  366    TRAPS-BREAK HANDLER
    16-  395    RESTART-INTRODUCTION
    20-  577    RESTART-ENTRY POINT
    21-  615    RESTART-SEE IF STACK EXISTS
    21-  633    RESTART-EXIT IF IN IN-ROM STATE
    22-  658    RESTART-CAUSE DETERMINATION
    23-  708    RESTART-EXITS
    24-  728    POWERUP-INTRODUCTION
    25-  751    POWERUP-TURN ON LED
    25-  772    POWERUP-TEST CONSOLE DLART
    26-  792    POWERUP-TEST AND SET UP I/O-PAGE RAM
    27-  818    POWERUP-TURN OFF LED
    27-  846    POWERUP-TEST FOR "LOW CORE"
    28-  869    POWERUP-EXIT
    28-  888    POWERUP-SUBROUTINE TO INITIALIZE VECTORS
    29-  913    AUTOBAUD-SYNCHRONIZE WITH CONSOLE
    31- 1003    MACROODT-INTRODUCTION
    33- 1071    MACROODT-SAVE STATUS AND PRINT PROMPT
    34- 1129    MACROODT-GET ODT COMMAND
    36- 1201    MACROODT- GO AND PROCEED
    37- 1259    MACROODT-REGISTER AND PS COMMAND
    38- 1299    MACROODT-EXAMINE AND DEPOSIT
    40- 1382    MACROODT-GET AND ECHO CHARACTER
    41- 1407    MACROODT-TYPE ASCII STRING
    42- 1439    MACROODT-GET OCTAL DIGITS
    43- 1477    MACROODT-OCTSTR--TYPE BINARY IN R0 AS ASCII
    44- 1511    MACROODT-OUTPUT MESSAGES
    45- 1527    DIAGNOSTICS-FOR SLU2 AND PPI
    46- 1567    HARDWARE ENTRY POINT
    47- 1575    DIAGNOSTICS-CONTINUED
    48- 1633    BOOTS-DESCRIPTION
    49- 1690    BOOTS-RX CONTROLLER DEFINITIONS
    49- 1737    BOOTS-TU58 DEFINITIONS AND PROTOCOL EQUATES
    49- 1795    BOOTS-RT11 DEFINITIONS AND EQUATES
    50- 1826    BOOTS-RD/RX CONTROLLER DEFINITIONS
    51- 1865    BOOTS-RX01/RX02 BOOTSTRAP
    52- 1887    BOOTS-DISTINGUISHING TYPE OF BOOT BLOCK
    52- 1909    -----> HALT AT PC=172260 INDICATES "NO BOOT BLOCK ON VOLUME"
    53- 1915    BOOTS-TU58 BOOTSTRAP
    53- 1943    -----> HALT AT PC=172346 INDICATES "TU58 INITIALIZATION ERROR"
    53- 1951    -----> HALT AT PC=172366 INDICATES "TU58 BLOCK 0 READ ERROR"
    54- 1954    BOOTS-STAND-ALONE VOLUME BOOTSTRAP
    54- 1977    -----> HALT AT PC=172420 INDICATES "DIRECTORY READ ERROR"
    54- 1989    -----> HALT AT PC=172456 INDICATES "FILE NOT FOUND"
    54- 2012    -----> HALT AT PC=172536 INDICATES "STAND-ALONE FILE READ ERROR"
    54- 2016    -----> HALT AT PC=172554 INDICATES "ILLEGAL TRANSFER ADDRESS"
```

TABLE OF CONTENTS


```
    54- 2033    173000G ENTRY POINT
    56- 2044    BOOTS-CONTINUED
    57- 2048    BOOTS-RX01/RX02 READ ROUTINES
    57- 2083    =====> HALT AT PC=173056 INDICATES "FLOPPY DRIVE NOT READY"
    57- 2161    =====> HALT AT PC=173250 INDICATES "FLOPPY READ ERROR"
    60- 2221    BOOTS-TU58 READ ROUTINES
    61- 2287    =====> HALT AT PC=173544 INDICATES "TU58 END PACKET MISSING"
    61- 2297    =====> HALT AT PC=173576 INDICATES "TU58 CHECKSUM ERROR"
    63- 2359    BOOTS-PROGRAM ENTRY POINT
    63- 2409    =====> HALT AT PC=164076 INDICATES "ILLEGAL DEVICE NAME"
    63- 2441    =====> HALT AT PC=164222 INDICATES "ILLEGAL UNIT NUMBER"
    63- 2448    =====> HALT AT PC=164242 INDICATES "NO LOW MEMORY, CAN'T BOOT"
    63- 2494    =====> HALT AT PC=164370 INDICATES "UNEXPECTED TIMEOUT DURING BOOT"
    64- 2497    BOOTS-RD/RX READ ROUTINE
    64- 2580    =====> HALT AT PC=164572 INDICATES "INVALID BOOT BLOCK"
    66- 2629    LOCAL DATA
    67- 2643    RLV-11 BOOTSTRAP
    72- 3083    END STATEMENT
```

```
      1
```

```
                                          .TITLE KXT11-A5 1K FIRMWARE
                                          .IDENT  /V1.00/
                                          .ENABL LC
  000000                                  .ASECT
              173776                      .=173776
  173776        000                       .BYTE 0
  173777        001                       .BYTE 1.
```

```
                .SBTTL COPYRIGHT NOTICE
        ;
        ; COPYRIGHT (C) 1980, 1981 BY
        ; DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASS.
        ;
        ; THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND COPIED
        ; ONLY IN ACCORDANCE WITH THE TERMS OF SUCH LICENSE AND WITH THE
        ; INCLUSION OF THE ABOVE COPYRIGHT NOTICE.  THIS SOFTWARE, OR ANY OTHER
        ; COPIES THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY
        ; OTHER PERSON.  NO TITLE TO AND OWNERSHIP OF THE SOFTWARE IS HEREBY
        ; TRANSFERRED.
        ;
        ; THE INFORMATION IN THIS DOCUMENT IS SUBJECT TO CHANGE WITHOUT NOTICE
        ; AND SHOULD NOT BE CONSTRUED AS A COMMITMENT BY DIGITAL EQUIPMENT
        ; CORPORATION.
        ;
        ; DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE OR RELIABILITY OF ITS
        ; SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.
        ;
        ; VERSION V1.00
        ;
        ;
```

```
                        .SBTTL  KXT11-A2 EDIT HISTORY

                ;EDIT HISTORY;
                ;
```

```
                                                   .SBTTL EQUATES

                                ; BIT EQUATES

        000001                  BIT0    =       1
        000002                  BIT1    =       2
        000004                  BIT2    =       4
        000010                  BIT3    =       10
        000020                  BIT4    =       20
        000040                  BIT5    =       40
        000100                  BIT6    =       100
        000200                  BIT7    =       200
        000400                  BIT8    =       400
        001000                  BIT9    =       1000
        002000                  BIT10   =       2000
        004000                  BIT11   =       4000
        010000                  BIT12   =       10000
        020000                  BIT13   =       20000
        040000                  BIT14   =       40000
        100000                  BIT15   =       100000

                                ; ASCII CHARACTER EQUATES

        000012                  LF      =       12              ;LINE FEED
        000015                  CR      =       15              ;CARRIAGE RETURN
        000040                  SPACE   =       40              ;SPACE

```

```
                                .SBTTL GENERAL DLART EQUATES

                        ; DLART EQUATES

        177560          RCSR$1  =       177560                  ;SLU1 RECEIVE CSR
        177562          RBUF$1  =       177562                  ;SLU1 RECEIVE BUFFER
        177564          XCSR$1  =       177564                  ;SLU1 XMIT CSR
        177566          XBUF$1  =       177566                  ;SLU1 XMIT BUFFER
        176540          RCSR$2  =       176540                  ;SLU2 RECEIVE CSR
        176542          RBUF$2  =       176542                  ;SLU2 RECEIVE BUFFER
        176544          XCSR$2  =       176544                  ;SLU2 XMIT CSR
        176546          XBUF$2  =       176546                  ;SLU2 XMIT BUFFER

                        ; DLART RECEIVE CSR BITS

        004000          RC.ACT  =       BIT11                   ;RECEIVER ACTIVE (R/O).  SET
                                                                ; WHILE CHARACTER IS BEING
                                                                ;  RECEIVED.
        000200          RC.DUN  =       BIT7                    ;RECEIVER DONE (R/O).  A
                                                                ; CHARACTER HAS BEEN COMPLETELY
                                                                ;  RECEIVED AND NOW RESIDES
                                                                ;   IN RBUF.
        000100          RC.IEN  =       BIT6                    ;RECEIVER INT. ENABLE (R/W).
                                                                ; WHEN SET, ENABLES "KEYBOARD"
                                                                ;  INTERRUPTS, USING VECTOR
                                                                ;   AT 60.

                        ; DLART RECEIVE BUFFER BITS (R/O)

        100000          RB.ERR  =       BIT15                   ;ERROR.  FRAMING ERROR OR
                                                                ; OVERRUN HAS OCCURRED.
        040000          RB.OVR  =       BIT14                   ;OVERRUN ERROR.  CHARACTER WAS
                                                                ; RECEIVED BEFORE PREVIOUS ONE
                                                                ;  WAS READ.
        020000          RB.FRM  =       BIT13                   ;FRAMING ERROR.  NO VALID STOP
                                                                ; BIT WAS DETECTED.
        004000          RB.BRK  =       BIT11                   ;BREAK DETECT.  SET WHEN BREAK
                                                                ; IS DETECTED, RESET WHEN NEXT
                                                                ;  START BIT ARRIVES.
```

```
                                ; DLART TRANSMIT CSR BITS

        000200                  XC.RDY  =       BIT7            ;TRANSMITTER READY (R/O).
                                                                ; WHEN SET, INDICATES THAT THE
                                                                ;  LAST CHARACTER WAS COMPLETELY
                                                                ;   SENT AND XBUF IS READY FOR
                                                                ;    A NEW ONE.
        000100                  XC.IEN  =       BIT6            ;TRANSMIT INT. ENABLE (R/W).
                                                                ;WHEN SET, ENABLES "CONSOLE
                                                                ; PRINTER" INTERRUPTS, USING
                                                                ;  VECTOR AT 64.

                                ; PROGRAMMABLE BAUD RATE BITS

        000010                  PBR0    =       BIT3
        000020                  PBR1    =       BIT4
        000040                  PBR2    =       BIT5

                                ; PBR0-2 SET BAUD RATES AS FOLLOWS:

        000000                  BD.003  =       0               ;BAUD RATE =   300
        000010                  BD.006  =       PBR0            ;BAUD RATE =   600
        000020                  BD.012  =       PBR1            ;BAUD RATE =  1200
        000030                  BD.024  =       PBR1!PBR0       ;BAUD RATE =  2400
        000040                  BD.048  =       PBR2            ;BAUD RATE =  4800
        000050                  BD.096  =       PBR2!PBR0       ;BAUD RATE =  9600
        000060                  BD.192  =       PBR2!PBR1       ;BAUD RATE = 19200
        000070                  BD.384  =       PBR2!PBR1!PBR0  ;BAUD RATE = 38400

        000004                  XC.MNT  =       BIT2            ;MAINTENANCE (R/W).  WHEN SET,
                                                                ; CREATES AN INTERNAL "LOOP-
                                                                ;  BACK" BETWEEN THE TRANSMITTER
                                                                ;   AND RECEIVER.  ALSO DIS-
                                                                ;    CONNECTS THE EXTERNAL
                                                                ;     SERIAL INPUT.
        000002                  XC.PBE  =       BIT1            ;PROG. BAUD RATE ENABLE.  WHEN
                                                                ; SET, THE BAUD RATE IS DETER-
                                                                ;  MINED BY BITS 3-5 AS
                                                                ;   TABULATED ABOVE.  WHEN
                                                                ;    CLEAR, BAUD RATE IS DETER-
                                                                ;     MINED BY VOLTAGES APPLIED
                                                                ;      TO DLART IC PINS.
        000001                  XC.BRK  =       BIT0            ;TRANSMIT BREAK (R/W).  WHEN
                                                                ; SET, SERIAL OUTPUT IS A
                                                                ;  CONTINUOUS BREAK.
```

```
                                   .SBTTL  GENERAL PPI EQUATES

                        ; PROGRAMMABLE PERIPHERAL INTERFACE (PPI) EQUATES

        176206          PP.CWR  =       176206                  ;PPI CONTROL WORD REGISTER
        176200          PP.A    =       176200                  ;PPI PORT A REGISTER
        176202          PP.B    =       176202                  ;PPI PORT B REGISTER
        176204          PP.C    =       176204                  ;PPI PORT C REGISTER

                        ; PPI MODE-SETTING BITS

                        ; KXT11-AA BOARD CONFIGURATION DOES NOT PERMIT ALL COMBINATIONS OF
                        ; THE MODE BITS.  CONSULT THE MANUAL BEFORE USING THE PPI.

        000200          PP.MOD  =       BIT7                    ;THIS MUST BE OR'D WITH OTHER
                                                                ; BITS TO SET MODE.
        000100          PP.MD2  =       BIT6                    ;SETS MODE 2
        000040          PP.MDA  =       BIT5                    ;IF BIT 6 IS LOW, DETERMINES
                                                                ; MODE OF PORT A
                                                                ;  (HI=MODE 1, LO=MODE 0)
        000020          PP.DRA  =       BIT4                    ;DIRECTION OF PORT A.
                                                                ; HI=IN, LO=OUT.
        000010          PP.CHI  =       BIT3                    ;DIRECTION OF PORT C UPPER HALF
                                                                ; HI=IN, LO=OUT.
        000004          PP.MDB  =       BIT2                    ;MODE OF PORT B.
                                                                ; HI=MODE 1, LO=MODE 0.
        000002          PP.DRB  =       BIT1                    ;DIRECTION OF PORT B.
                                                                ; HI=IN, LO=OUT.
        000001          PP.CLO  =       BIT0                    ;DIRECTION OF PORT C LOWER HALF
                                                                ; HI=IN, LO=OUT.

                        ; PPI BIT SET/RESET CONTROL BITS

                        ;       WHEN BIT 7 IS LOW, WRITING TO THE PPI CSR WILL SET OR RESET
                        ;       INDIVIDUAL BITS IN PORT C, DEPENDING ON THE MODE AND DIRECTION
                        ;       OF THE PORT'S BITS, AND ON THE COMBINATION OF BITS YOU WRITE.

        000016          PP.BI7  =       BIT3!BIT2!BIT1          ;USE ONE
        000014          PP.BI6  =       BIT3!BIT2               ;OF THESE
        000012          PP.BI5  =       BIT3!BIT1               ;TO SELECT
        000010          PP.BI4  =       BIT3                    ;WHICH BIT
        000006          PP.BI3  =       BIT2!BIT1               ;IS DESIRED
        000004          PP.BI2  =       BIT2                    ;TO BE
        000002          PP.BI1  =       BIT1                    ;SET OR
        000000          PP.BI0  =       0                       ;CLEARED

        000001          PP.BIS  =       BIT0                    ;SET SPECIFIED BIT.
        000000          PP.BIC  =       0                       ;CLEAR SPECIFIED BIT.
```

```
                                .SBTTL  PROGRAM-SPECIFIC EQUATES

                        ; EQUATES USED TO TURN LED ON AND OFF

        000221          MODE    =       PP.MOD!PP.ORA!PP.CLO    ;PORT A = MODE 0 IN
                                                                ;PORT B = MODE 0 OUT
                                                                ;PORT C UPPER NIBBLE = OUT
                                                                ;PORT C LOWER NIBBLE = IN

        000017          LEDOFF  =       PP.BIS!PP.BI7           ;SET PC7

                        ; EQUATES USED TO SET UP DLARTS

        000032          BAUDRS  =       BD.024!XC.PBE           ;INITIAL CONSOLE BAUD RATE TO
                                                                ; BE 2400, WITH PROG. BAUD
                                                                ;  RATES ENABLED.

        000072          TUBAUD  =       BD.384!XC.PBE           ;TU58 BAUD RATE = 38,400

                        ; MEMORY CONFIGURATION EQUATES

        177600          RAMBOT  =       177600                  ;BOTTOM ADDRESS OF RAM
        177776          RAMTOP  =       177776                  ;TOP ADDRESS OF RAM

                        ; SOFTWARE FLAGS AND MASKS

        000300          PRI6    =       300                     ;PS FOR PRIORITY OF 6
        000340          PRI7    =       340                     ;PS FOR PRIORITY OF 7

                        ; USED BY ODT MODULE

        000200          RFLAG   =       BIT7                    ;REGISTER FLAG BIT- INDICATES
                                                                ; REGISTER IS BEING EXAMINED
        000020          T.BIT   =       BIT4                    ;TRACE BIT IN PSW
```

```

                        ; RESTART TYPE WORD BITS

        100000          R.HALT  =       BIT15           ;HALT OR BREAK OCCURRED
        000200          R.NXM   =       BIT7            ;ACCESSED NON-EXISTENT MEMORY
        000001          R.STAK  =       BIT0            ;DOUBLE-BUS ERROR

                        ; BOOT CONTROL WORD BITS

        100000          NO.LOW  =       BIT15           ;NO MEMORY FOUND AT 000000-
                                                        ; DO NOT BOOT
        000200          DEVBIT  =       BIT7            ;1 = RX01/02 FLOPPY
                                                        ;0 = TU58 CASSETTE
        000001          DEVNUM  =       BIT0            ;UNIT NO. (0 OR 1)

                        ; DIAGNOSTIC MESSAGES

        000100          E.EXT   =       100             ;SLU2 LOOPBACK TEST FAILED
        000010          E.INT   =       10              ;SLU2 INTERNAL LOOPBACK FAILED
        000001          E.PAR   =       1               ;PARALLEL PORT LOOPBACK FAILED
```

```
                                        .SBTTL MACRO DEFINITIONS

        ;
        ;       MACRO DEFINITIONS
        ;

        ;+
        ;
        ; THIS MACRO WILL INSERT ABORTS INTO THE CODE WHICH WILL HALT THE
        ; PROGRAM, EXIT TO ODT WITH THE PC PRINTED ON THE CONSOLE, AND GENERATE
        ; AN ENTRY IN THE TABLE OF CONTENTS WHICH DESCRIBES THE ERROR CONDITION.
        ;
        ;-

        .MACRO  ABORT   TEXT
                HALT
        .IRP    PCS,\.
        .SBTTL  -----> HALT AT PC='PCS INDICATES "'TEXT"
        .ENDR
                BR      .-2
        .ENDM
```

```
;+
; DELAY A,B,N
;       WHERE A AND B ARE NAMES OF REGISTERS THAT ARE FREE (BOTH WILL
;       BE CLEAR WHEN THROUGH) AND N IS AN INTEGER.
;
; THIS MACRO PRODUCES A DELAY WHOSE DURATION (WHEN RUNNING IN KXT11-AA
; ROM) IS .2399N SECONDS.
; WHEN N<4, IT IS MORE EFFICIENT TO USE THE FOLLOWING CODE:
;
;       CLR     RN                          ;1W           2.44
;       SOB     RN,.                        ;1W      239861.76
;       [SOB    RN,.                        ;1W      239861.76]
;       [SOB    RN,.                        ;1W      239861.76]
;
; THE MACRO GENERATES CODE LIKE THE FOLLOWING:
;
;       MOV     #N,RA                       ;2W           3.66
;NS:    CLR     RB                          ;1W        N*2.44
;       SOB     RB,.                        ;1W   65536N*3.66
;       SOB     RA,NS                       ;1W        N*3.66
;-

        .MACRO  DELAY   A,B,N,?L
        MOV     #N,A
L:      CLR     B
        SOB     B,.
        SOB     A,L
        .ENDM
```

```
                                .SBTTL  RAM DEFINITION

                        ; SCRATCH RAM AREA

        177776          TRAP4   ==      177776          ;ENABLES TRAP-TO-4 EMULATION
                                                        ; WHEN NON-ZERO
        177774          ODTWHY  ==      177774          ;USER-READABLE COPY OF R.TYPE.
                                                        ; RESTART CAUSE.  SEE R.TYPE
                                                        ;  TABLE IN RESTART ROUTINE.
        177772          O.CNTL  ==      177772          ;ODT CONTROL WORD.  SET BIT 15
                                                        ; TO DISABLE T-BIT FILTER, SET
                                                        ;  BIT 7 TO DISABLE PRIORITY 7
                                                        ;   FILTER.
        177770          B.CNTL  ==      177770          ;BOOT CONTROL WORD.
        177766          R.PC    ==      177766          ;WHERE RESTART SAVES TOP OF STACK
        177764          IN.USR  ==      177764          ;ENABLES USER-CAUSED RESTART
                                                        ; AND BREAK WHEN NON-ZERO
        177762          R.TYPE  ==      177762          ;RESTART CAUSE.  SEE TABLE IN
                                                        ; RESTART ROUTINE.
        177760          USERSP  ==      177760          ;USED BY ODT TO STORE THE USER'S
                                                        ; STACK POINTER.
        177756          RPOINT  ==      177756          ;USED BY ODT TO POINT TO THE IMAGE
                                                        ; OF USER'S R0 IN ITS STACK.
        177754          SAVPS   ==      177754          ;STORE HALTED PS HERE FOR ODT
        177752          SAVPC   ==      177752          ;STORE HALTED PC HERE FOR ODT
        177750          ODTFLG  ==      177750          ;USED BY ODT FOR INTERNAL FLAGS.
        177746          ODTLOC  ==      177746          ;USED BY ODT TO POINT TO LOCATION
                                                        ; CURRENTLY OPEN.
        177744          ODTSTK  ==      177744          ;BOTTOM OF ODT'S STACK
        177644          $STACK  ==      ODTSTK-100      ;BOTTOM OF DEFAULT USER STACK
```

```
                                          .SBTTL MSCP BUFFER AREA
        001000                 .=1000


001000               BUFFER: .BLKW   256.                          ;BUFFER FOR READ DATA

002000                       .BLKW   2.                            ;PACKET ENVELOPE

002004               RMSG:   .BLKB   48.                           ;BUFFER FOR RESPONSE PACKET FROM UDA

002064                       .BLKW   2.                            ;PACKET ENVELOPE

002070               RCMD2:  .BLKB   48.                           ;BUFFER FOR COMMAND PACKET TO UDA


002150                       .BLKW   2.                            ;INTERRUPT FLAGS
002154               RING:   .BLKW   4.                            ;HOST COMMUNICATION AREA
```

```
         170000                              .=170000

                                                     .SBTTL TRAPS-TRAP-HANDLING ROUTINES
                                                     .SBTTL TRAPS-LTC TRAP-KILLER
                                                     .SBTTL TRAPS-BREAK HANDLER

                                        ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
                                        ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
                                        ;;;;                                                    ;;;;
                                        ;;;;               BREAK-HANDLING ROUTINE               ;;;;
                                        ;;;;        AND LINE TIME CLOCK INTERRUPT KILLER        ;;;;
                                        ;;;;                                                    ;;;;
                                        ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
                                        ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

170000                                  $$$BRK::
170000  005767  007760                          TST     IN.USR                  ;ARE WE IN USER MODE?
170004  001001                                  BNE     BRKNOP                  ;YES-GO TO ODT

170006                                  $$$LTC::
170006  000002                                  RTI                             ;NO-GO BACK TO ROM PROGRAM,
                                                                                ; BREAKS ARE IGNORED BY ODT,
                                                                                ;  RESTART, POWERUP AND THE
                                                                                ;   DIAGNOSTICS.  THE BOOTS
                                                                                ;    CAN BE INTERRUPTED, THOUGH.
170010                                  BRKNOP::
170010  012667  007736                          MOV     (SP)+,SAVPC             ;SAVE CONTEXT
170014  012667  007734                          MOV     (SP)+,SAVPS             ;FOR ODT.
170020  012767  100000  007734                  MOV     #R.HALT,R.TYPE          ;CAUSES PC TO BE PRINTED
                                                                                ; UPON ENTRY TO ODT.
170026  005067  007732                          CLR     IN.USR                  ;GET OUT OF USER MODE
170032  000167  000564                          JMP     ODT
```

```
                .SBTTL RESTART-INTRODUCTION

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;;                                                        ;;;;
;;;;                    RESTART MODULE                      ;;;;
;;;;                                                        ;;;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

;+
;
;THE PURPOSE OF THE RESTART ROUTINE IS TO RESTORE THE FALCON TO A
;KNOWN STATE FOLLOWING THOSE EXCEPTIONS WHICH CAUSE A RESTART HARDWARE
;ACTION.  THIS ACTION CONSISTS OF STACKING THE CURRENT PSW AND PROGRAM
;COUNTER, THEN SETTING THE PSW TO 340 AND JUMPING TO THE HARDWIRED
;RESTART LOCATION.  THIS LOCATION IS AT THE ADDRESS START+4 WHERE
;START IS JUMPER SELECTABLE AS 000000, 010000, 020000, 040000, 100000,
;140000, 172000 OR 173000 (ALL IN OCTAL).  THIS PROGRAM IS DESIGNED
;FOR A START LOCATION OF 172000, THUS RESTARTS JUMP TO 172004.
;
;THERE ARE SEVERAL DIFFERENT WAYS IN WHICH RESTART PERFORMS ITS
;FUNCTION, DEPENDING ON THE VALUE OF IN.USR, TRAP4, THE CONTENTS
;OF THE LOCATION THE SP POINTS TO, AND ONE BIT (R.STAK) IN R.TYPE.
;
;R.TYPE, THE RESTART TYPE WORD, IS RESTART'S OUTPUT TO ODT.
;
;-

;+
;
;THE GOAL IS TO MAXIMIZE PDP-11 SOFTWARE COMPATIBILITY AND TO PROVIDE
;USEFUL DEBUGGING INFORMATION TO THE PROGRAM DEVELOPER.
;
;-
```

```
;             ----------
;             ! R.STRT !   ENTER VIA HARDWARE MECHANISM,
;             ----------   WITH (PS)=340
;                  !
;          ----------------
;          ! IS THE STACK !
;          !  FLAG SET?   !
;          ! N         Y  !
;          ----------------
;             !         !
;             !   -------------------
;             !   !     SET R.NXM   !
;             !   ! SET IN-ROM MODE !
;             !   !    GO TO ODT    !
;             !   -------------------
; --------------------------------
; !        SET STACK BIT         !
; !      READ TOP OF STACK       !<---COULD TIME OUT AND CAUSE THE
; ! CHECK IF TOO CLOSE TO "HOLE" !<-----EXIT TO ODT SHOWN ABOVE
; !       CLEAR STACK BIT        !
; --------------------------------
;             !
;        ---------------------
;        ! DID RESTART OCCUR !
;        !  IN USER MODE?    !
;        ! N               Y !
;        ---------------------
;           !              !
; -----------------        !
; !  IS TOP OF    !<-------+------ A BREAK DOES THIS WHEN
; ! STACK 000000? !        !       THERE'S NO MEMORY IN
; ! N           Y !        !       THE VECTOR AREA.
; -----------------        !
;    !          !          !
;    !    ---------------  !
;    !    ! POP STACK   !<-+------ ONLY A BREAK WHILE IN ODT CAN
;    !    ! FRAME AND   !  !       GET US HERE, SO THE RTI TAKES
;    !    !  RETURN     !  !       US BACK TO ODT.
;    !    ---------------  !
;  ----------------   -------------------
;  ! SET CARRY    !   ! LEAVE USER MODE !
;  ! IN PUSHED PS !   -------------------
;  ! AND RETURN   !            !
;  ----------------    -----------------
;                      !   IS TOP OF   !
;                      ! STACK 000000? !
; A BREAK DOES THIS--->! Y           N !
; WHEN THERE'S NO      -----------------
; MEMORY IN THE            !           !
; VECTOR AREA     -----------------    !
;                 !   POP STACK   !<---+-------THIS ENTRY POINT IS IN
;                 ! FRAME AND GO  !    !       THE TRAPS MODULE.  IT IS
;                 !  TO BREAK'S   !    !       WHERE A BREAK IN USER MODE
;                 ! SAVE CONTEXT  !    !       GOES WHEN THERE IS MEMORY
;                 ! ENTRY POINT   !    !       IN THE VECTOR AREA.
;                 -----------------    !
;                                      !
```

```
;                                          !
;                               ---------------------
;                               ! POP STACK FRAME   !
;                               ! IF 172004 ON TOP, !
;                               !  GET PUSHED PC,   !
;                               !  SET UP ODT'S PC  !
;                               ! AND PS LOCATIONS. !
;                               ---------------------
;                                          !
;                               ---------------------
;                   COULD---->! TEST WORD PRIOR   !
;                   TIME----->! TO WHERE PUSHED   !
;                   OUT------>!    PC POINTS      !
;                               ---------------------
;                                          !
;                               --------------------
;                               !   WAS THE WORD   !
;                               !      A HALT      !
;                               ! OR DID PC-->NXM? !
;                               !   Y          N   !
;                               --------------------
;                                   !          !
;                        ----------------      !
;                        ! SET HALT FLAG !     !
;                        !  GO TO ODT    !     !
;                        ----------------      !
;                                              !
;                                 ----------------
;                                 ! IS TRAP-TO-4 !
;                                 !  EMULATION   !
;                                 !   ENABLED?   !
;                                 !  N        Y  !
;                                 ----------------
;                                    !        !
;                      ---------------        !
;                      ! SET NXM FLAG !       !
;                      !  GO TO ODT   !       !
;                      ---------------        !
;                                             !
;                                  ------------------
;                                  ! SET USER MODE  !
;                                  ! PUSH @#6, @#4  !
;                                  !  ONTO STACK    !
;                                  !    AND RTI     !
;                                  ------------------
```

```
;+
;
;EXCEPTION-TYPE WORD (R.TYPE) IS PASSED TO ODT AND IS RESTARTS "BEST GUESS"
;AS TO WHY A RESTART HAPPENED:
;
;        NOTE: A USER-READABLE COPY OF THIS WORD IS AT ODTWHY.
;
;|------|-----|--------|------------------------------------------------
;| EXIT | BIT |  NAME  | CAUSE
;|------|-----|--------|------------------------------------------------
;|------|     |        |
;| ODT  | 15  | R.HALT | HALT INSTRUCTION IN USER CODE-RESTART POPS STACK.
;|      |     |        | NOTE-BREAK ALSO SETS THIS BIT (SEE THE TRAPS
;|      |     |        | MODULE).  ODT USES THIS BIT FOR PDP-11 ODT
;|      |     |        | COMPATIBILITY.
;|------|     |        |
;| ODT  | 14  |        | RESERVED
;| OR   | 13  |        | RESERVED
;| TRAP | 12  |        | RESERVED
;| TO   | 11  |        | RESERVED
;| FOUR | 10  |        | RESERVED
;|      |  9  |        | RESERVED
;|      |  8  |        | RESERVED
;|      |  7  | R.NXM  | TIMEOUT DURING USER ACCESS OF NON-EXISTANT
;|      |     |        |   MEMORY
;|      |  6  |        | RESERVED
;|      |  5  |        | RESERVED
;|      |  4  |        | RESERVED
;|      |  3  |        | RESERVED
;|      |  2  |        | RESERVED
;|      |  1  |        | RESERVED
;|------|-----|--------|
;| ODT  |  0  | R.STAK | INDICATES THAT A TIMEOUT WAS CAUSED BY RESTART
;|      |     |        |   ITSELF ACCESSING NON-EXISTANT MEMORY.  THIS
;|      |     |        |   OCCURS IN CONJUNCTION WITH TESTING FOR
;|      |     |        |   VALIDITY OF THE STACK POINTER.
;|      |     |        |   IN PDP-11 PARLANCE, THIS IS A
;|      |     |        |   "DOUBLE-BUS ERROR"
;|------|-----|--------|------------------------------------------------
;
;-
```

```
                                             .SBTTL RESTART-ENTRY POINT

                                  ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
                                  ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
                                  ;;;;                                                        ;;;;
                                  ;;;;                  RESTART ENTRY POINT                   ;;;;
                                  ;;;;                                                        ;;;;
                                  ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
                                  ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

170036                            R.STRT::

                                  ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
                                  ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
                                  ;;;;                                                        ;;;;
                                  ;;;;    IF THE RESTART ROUTINE CAUSED THE RESTART           ;;;;
                                  ;;;;            GO TO ODT AND PRINT "?"                     ;;;;
                                  ;;;;                                                        ;;;;
                                  ;;;;     THIS EXCEPTION CAN BE CAUSED BY RESTART'S          ;;;;
                                  ;;;;               STACK MANIPULATIONS                      ;;;;
                                  ;;;;                                                        ;;;;
                                  ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
                                  ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

                                  ; R.TYPE WILL HAVE BEEN CLEARED PRIOR TO ENTERING
                                  ; ANY ODT COMMAND.  SO, IF THE STACK BIT IS SET, ONLY RESTART
                                  ; ITSELF COULD HAVE CAUSED THE TRAP.  SINCE THE STACK IS ALWAYS
                                  ; VALID IN IN-ROM MODE, BAD STACK MEANS WE ARE IN IN-USER MODE.

                                  ;STATE: X=DON'T CARE, U=USER, R=IN-ROM----
                                  ;                                        |
170036  005767  007720                    TST     R.TYPE                  ;X|DID THE STACK TEST FAIL?
170042  001406                            BEQ     1$                      ;X|NO- GO TO NEXT TEST
170044  052767  000200  007710            BIS     #R.NXM,R.TYPE           ;U|YES- SET R.NXM
                                                                          ;U| THIS FORCES "?" FROM ODT
170052  005067  007706                    CLR     IN.USR                  ;R|ENTER IN-ROM MODE
170056  000476                            BR      8$                      ;R|GO TO ODT
```

```
                                                    .SBTTL  RESTART-SEE IF STACK EXISTS

                                        ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
                                        ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
                                        ;;;;                                                      ;;;;
                                        ;;;;                  STACK VALIDITY TEST                 ;;;;
                                        ;;;;                                                      ;;;;
                                        ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
                                        ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

170060  052767  000001  007674  1$:     BIS     #R.STAK, R.TYPE         ;X|IF WE TIMEOUT, WE WANT RESTART
                                                                        ;X| TO KNOW WE WERE DIDDLING SP
170066  005716                          TST     (SP)                    ;X|SEE IF STACK IS VALID
170070  000240                          NOP                             ;X|(IN CASE TIMES OUT)
170072  005766  000004                  TST     4(SP)                   ;X|SEE IF TOO CLOSE TO TOP OF
170076  000240                          NOP                             ;X| VALID MEMORY
170100  005067  007656                  CLR     R.TYPE                  ;X|STACK IS OK

                                                    .SBTTL RESTART-EXIT IF IN IN-ROM STATE

                                        ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
                                        ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
                                        ;;;;                                                      ;;;;
                                        ;;;;      RETURN WITH CARRY SET IF IN "IN-ROM" MODE       ;;;;
                                        ;;;;                                                      ;;;;
                                        ;;;;  OR, GO BACK TO ODT IF A BREAK WITH NO LOW MEMORY    ;;;;
                                        ;;;;                                                      ;;;;
                                        ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
                                        ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

170104  005767  007654                  TST     IN.USR                  ;X|ARE WE IN USER MODE?
170110  001007                          BNE     3$                      ;U|YES-GO TO NEXT TEST
170112  005716                          TST     (SP)                    ;R|NO-SEE IF BREAK BROUGHT
                                                                        ;R| US HERE
170114  001002                          BNE     2$                      ;R|NO-JUST A RESTART
170116  022626                          CMP     (SP)+,(SP)+             ;R|YES-BEHAVE LIKE A BREAK THAT
170120  000002                          RTI                             ;R| HAPPENED WITH RAM

170122  005266  000002          2$:     INC     2(SP)                   ;R|SET CARRY IN PUSHED PS
                                                                        ;R| UNLESS ALREADY SET
170126  000002                          RTI                             ;R|AND RETURN TO ROM CODE THAT
                                                                        ;R| CAUSED TIMEOUT
```

```
                                           .SBTTL RESTART-CAUSE DETERMINATION

                                ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
                                ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
                                ;;;;                                                                ;;;;
                                ;;;;        DETERMINE HOW USER CAUSED A RESTART                     ;;;;
                                ;;;;                                                                ;;;;
                                ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
                                ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

170130  005067  007630          3$:     CLR     IN.USR                  ;U|WE WERE IN USER MODE,
                                                                        ;R| BUT NO LONGER.
170134  005716                          TST     (SP)                    ;R|SEE IF BREAK BROUGHT
                                                                        ;R| US HERE WITHOUT LOW "CORE".
170136  001003                          BNE     4$                      ;R|NO-JUST A RESTART
170140  022626                          CMP     (SP)+,(SP)+             ;R|YES-BEHAVE LIKE A BREAK THAT
170142  000167  177642                  JMP     BRKN00                  ;R| HAPPENED WHILE IN USER PROG.


                                ;
                                ;       IF THE CPU ATTEMPTS TO FETCH AN INSTRUCTION FROM NON-EXISTENT
                                ;       MEMORY, TWO TRAPS (THE FIRST FROM EXECUTING A HALT, THE SECOND
                                ;       FROM TIMING OUT) WILL OCCUR, THE RESULT BEING THAT SECOND
                                ;       TRAP PUSHES THE RESTART ADDRESS AND 340 ON THE STACK.
                                ;       THIS INFORMATION IS USELESS AND GETS POPPED HERE.

170146  021627  172004          4$:     CMP     (SP),#RESTAR            ;X|GET RID OF DOUBLE STACKING
170152  001001                          BNE     5$                      ;X|CAUSED BY EXECUTION OF NXM
170154  022626                          CMP     (SP)+,(SP)+             ;X|

                                ;       NOTE: BECAUSE THE CONTENTS OF THE STACK IS ASSUMED TO REMAIN
                                ;       UNCHANGED FOLLOWING THE FIRST INSTRUCTION BELOW, IT IS IMPERATIVE
                                ;       THAT INTERRUPTS BE DISABLED DURING THE NEXT THREE INSTRUTIONS.

170156  012667  007604          5$:     MOV     (SP)+,R.PC              ;R|GET PUSHED PC
170162  011667  007566                  MOV     (SP),SAVPS              ;R|ODT WOULD LIKE
170166  014667  007560                  MOV     -(SP),SAVPC             ;R|TO SEE THESE

170172  162767  000002  007566          SUB     #2,R.PC                 ;R|SET POINTER TO LAST WORD FETCHED
                                                                        ;R| BEFORE RESTART OCCURRED
170200  005777  007562                  TST     @R.PC                   ;R|IS CONTENTS OF PUSHED PC - 2
                                                                        ;R| A ZERO (EG A HALT)?
170204  000240                          NOP                             ;R|MAKE SURE NEXT INSTRUCTION
                                                                        ;R| WON'T EXECUTE IF WE TIME OUT
170206  001005                          BNE     6$                      ;R|NO- IT WAS AN NXM
170210  052767  100000  007544          BIS     #R.HALT,R.TYPE          ;R|YES- FLAG A HALT,
170216  022626                          CMP     (SP)+,(SP)+             ;R|POP THE NON-PDP-11 STACK FRAME
170220  000415                          BR      8$                      ;R|AND GO TO ODT.
```

```
                                                .SBTTL RESTART-EXITS

                              ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
                              ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
                              ;;;;                                                    ;;;;
                              ;;;;                 EXIT APPROPRIATELY                 ;;;;
                              ;;;;                                                    ;;;;
                              ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
                              ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

170222  005767  007550        6$:     TST     TRAP4                   ;R|TRAP-TO-4 EMULATION ENABLED?
170226  001407                        BEQ     7$                      ;R|NO-GO TO ODT
170230  005167  007530                COM     IN.USR                  ;U|YES-SET USER MODE
170234  013746  000006                MOV     @#6,-(SP)               ;U|EMULATE A
170240  013746  000004                MOV     @#4,-(SP)               ;U|TRAP TO
170244  000002                        RTI                             ;U|FOUR

170246  052767  000200  007506  7$:   BIS     #R.NXM, R.TYPE          ;R|FLAG NXM ERROR
170254  000167  000342        8$:     JMP     ODT                     ;R|GO TO ODT
```

```
                                .SBTTL POWERUP-INTRODUCTION

                        ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
                        ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
                        ;;;;                                                      ;;;;
                        ;;;;                  POWER-UP MODULE                     ;;;;
                        ;;;;                                                      ;;;;
                        ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
                        ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

                        ; THIS MODULE CONTAINS A SERIES OF ROUTINES WHICH PERFORM
                        ; TESTS ON THE ON-BOARD RAM AND THE CONSOLE DLART.  THESE
                        ; TESTS ARE PRECEDED BY THE LIGHTING OF THE LED ON THE
                        ; KXT11-AA BOARD, AND FOLLOWED BY ITS EXTINGUISHING.  SHOULD
                        ; THE LED FAIL TO EITHER LIGHT OR GO OUT, THERE MAY BE A
                        ; DEFECT IN THE BOARD OR ITS CONFIGURATION.

                        ; FOLLOWING THESE TESTS, THE ON-BOARD RAM IS WRITTEN WITH THE
                        ; DEFAULT VALUES OF CERTAIN CONTROL WORDS, AND, IF THERE IS
                        ; MEMORY IN THE VECTOR REGION (I.E., NEAR 000000), THE BREAK
                        ; AND CLOCK VECTORS ARE SET UP.  IF NOT, A BIT IS SET IN THE
                        ; BOOT CONTROL WORD TO DISABLE THE BOOTSTRAPS.
```

```
                                          .SBTTL POWERUP-TURN ON LED

                                  ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
                                  ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
                                  ;;;;                                                        ;;;;
                                  ;;;;                   TURN ON LED                          ;;;;
                                  ;;;;                                                        ;;;;
                                  ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
                                  ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

170260                            PWRSUP::
170260  012706  177644                    MOV     #$STACK,SP              ;INITIALIZE STACK POINTER

                                  ; BECAUSE A MODE-SETTING COMMAND AUTOMATICALLY CLEARS ALL THE INTERNAL
                                  ; REGISTERS IN THE PPI, AND CLEARING PORT C BIT 7 TURNS ON THE LED, ALL
                                  ; WE HAVE TO DO IS SET THE MODE, WHICH IS PORT A AND LO HALF OF C AS
                                  ; INPUT, PORTS B AND HI HALF OF C AS OUTPUT.

170264  012737  000221  176206            MOV     #MODE,@#PP.CWR          ;SET PROPER PPI MODE

                                          .SBTTL POWERUP-TEST CONSOLE DLART

                                  ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
                                  ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
                                  ;;;;                                                        ;;;;
                                  ;;;;            CHECK THE CONSOLE DLART                     ;;;;
                                  ;;;;                                                        ;;;;
                                  ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
                                  ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

170272  005037  177564                    CLR     @#XCSR$1                ;DISABLE XMIT INTERRUPTS,
                                                                          ; BRK XMIT, MAINT, MODE
                                                                          ;  SET BAUD RATE TO DEFAULT
170276  005737  177562                    TST     @#RBUF$1                ;TAKE OUT THE TRASH.
170302  032737  000300  177560            BIT     #<RC.IEN!RC.DUN>,@#RCSR$1
                                                                          ;SHOULD BE CLEAR.
170310  001377                            BNE     .                       ;IF NOT, DROP DEAD.
170312  023727  177564  000200            CMP     @#XCSR$1,#XC.RDY        ;SHOULD BE SET
170320  001377                            BNE     .                       ;IF NOT, REST IN PEACE.
```

```
                                          .SBTTL POWERUP-TEST AND SET UP I/O-PAGE RAM

                                  ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
                                  ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
                                  ;;;;                                                            ;;;;
                                  ;;;;                    I/O PAGE RAM TEST                       ;;;;
                                  ;;;;                    AND INITIALIZATION                      ;;;;
                                  ;;;;                                                            ;;;;
                                  ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
                                  ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

                                  ; WRITE THE LOCATION'S ADDRESS INTO THE LOCATION AND READ IT BACK.
                                  ; DO THIS FOR ALL I/O PAGE RAM LOCATIONS.
                                  ; IF IT FAILS, ENTER TIGHT LOOP.

                                  ; IN THE PROCESS, CLEAR ALL OF THIS RAM.  NOTE THAT THE DEFAULT
                                  ; VALUE OF MOST OF THE CONTROL AND FLAG WORDS IS ZERO.

170322  012700  177600                    MOV     #RAMBOT,R0              ;LOWEST ADDRESS OF RAM
170326  010010                    1$:     MOV     R0,(R0)                 ;WRITE THE ADDRESS
170330  020010                            CMP     R0,(R0)                 ;READ IT BACK
170332  001377                    2$:     BNE     2$                      ;TIGHT FAILURE LOOP
170334  005020                            CLR     (R0)+                   ;CLEAR AND GO ON TO NEXT LOCATION
170336  020027  000000                    CMP     R0,#RAMTOP+2            ;UNTIL NO MORE TO TEST.
170342  001371                            BNE     1$
```

```
                                                .SBTTL POWERUP-TURN OFF LED

                                ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
                                ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
                                ;;;;                                                        ;;;;
                                ;;;;  DELAY SO THAT LED IS ON A VISIBLE LENGTH OF TIME      ;;;;
                                ;;;;                                                        ;;;;
                                ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
                                ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

170344  005000                          CLR     R0
170346  077001                  3$:     SOB     R0,3$                   ;THIS LEAVES A 0 IN R0, WHICH
170350  077001                  4$:     SOB     R0,4$                   ; IS ESSENTIAL FOR TESTING FOR
170352  077001                  5$:     SOB     R0,5$                   ;  THE PRESENCE OF MEMORY AT
170354  077001                  6$:     SOB     R0,6$                   ;    ZERO BELOW.

                                ; UNDER NO CIRCUMSTANCES CAN R0 BE ALTERED UNTIL "LOW CORE" TEST BELOW.

                                ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
                                ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
                                ;;;;                                                        ;;;;
                                ;;;;                    TURN OFF LED                        ;;;;
                                ;;;;                                                        ;;;;
                                ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
                                ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

170356  012737  000017  176206          MOV     #LEDOFF,@#PP.CWR

                                                .SBTTL POWERUP-TEST FOR "LOW CORE"

                                ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
                                ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
                                ;;;;                                                        ;;;;
                                ;;;;             TEST FOR MEMORY AT 000000                  ;;;;
                                ;;;;                                                        ;;;;
                                ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
                                ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

                                ; READ MEMORY AT 000000, DISCARD RESULT.  IF THIS FAILS, EXIT TO
                                ; AUTOBAUD RATHER THAN CONTINUING WITH NORMAL POWERUP SEQUENCE.

170364  005710                          TST     (R0)
170366  000240                          NOP                             ;THIS WILL EXECUTE EVEN IF
                                                                        ; LAST INSTRUCTION TIMES OUT
170370  103403                          BCS     7$                      ;TIMED OUT, DON'T SET VECTORS
170372  004767  000042                  CALL    VECSET                  ;DIDN'T TIME OUT, SO GO AHEAD
170376  000403                          BR      8$                      ;DON'T SET THE NO.LOW FLAG

170400  052767  100000  007362  7$:     BIS     #NO.LOW,B.CNTL          ;DID TIME OUT,
                                                                        ; SO LET THE WORLD KNOW.
```

```
                                                      .SBTTL POWERUP-EXIT

                                         ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
                                         ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
                                         ;;;;                                                       ;;;;
                                         ;;;;             EXIT FROM POWER-UP SEQUENCE               ;;;;
                                         ;;;;                                                       ;;;;
                                         ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
                                         ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

170406  012767  000300  007340   8S:     MOV     #PRI6,SAVPS             ;IF P IS TYPED IN REPONSE TO
170414  012767  170424  007330           MOV     #FAKOUT,SAVPC           ; ODT PROMPT BEFORE LOADING R7,
170422  000433                           BR      AUTOBA                  ;  WILL FORCE YET MORE ODT.

170424                           FAKOUT:
170424  005067  007334                   CLR     IN.USR                  ;   BUT IN THE RIGHT MODE!
170430  012706  177644                   MOV     #$STACK,SP              ;    AND WITHOUT RUNNING OUT OF
170434  000167  000162                   JMP     ODT                     ;     STACK, EITHER.

                                                 .SBTTL POWERUP-SUBROUTINE TO INITIALIZE VECTORS

                                         ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
                                         ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
                                         ;;;;                                                       ;;;;
                                         ;;;;                 INITIALIZE VECTORS                    ;;;;
                                         ;;;;                                                       ;;;;
                                         ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
                                         ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

                                 ; NOTE: THIS SUBROUTINE IS ALSO USED BY THE BOOTSTRAP MODULE, TO
                                 ; RESTORE THE VECTOR AREA IN THE EVENT THAT AN INVALID BOOT BLOCK
                                 ; WAS READ INTO LOW MEMORY.

170440                           VECSET::
170440  022737  000340  000142           CMP     #340,@#142              ;VECTOR 140 SET UP
170446  001406                           BEQ     V102                    ;IF YES
170450  012737  170000  000140           MOV     #$$SBRK,@#140           ;SET UP THE BREAK-DETECT
170456  012737  000340  000142           MOV     #PRI7,@#142             ; VECTOR.
170464  022737  000340  000102   V102:   CMP     #340,@#102              ;VECTOR 100 SET UP ?
170472  001406                           BEQ     OUT                     ;IF EQ YES
170474  012737  170006  000100           MOV     #$$SLTC,@#100           ;SET UP THE LINE TIME CLOCK
170502  012737  000340  000102           MOV     #PRI7,@#102             ; VECTOR.
170510  000207                   OUT:    RETURN
```

```
                                        .SBTTL  AUTOBAUD-SYNCHRONIZE WITH CONSOLE

                                ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
                                ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
                                ;;;;                                                        ;;;;
                                ;;;;                   AUTOBAUD MODULE                      ;;;;
                                ;;;;                                                        ;;;;
                                ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
                                ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

                                ; DESCRIPTION:
                                ;
                                ;       AUTOBAUD ALLOWS THE FALCON TO AUTOMATICALLY SYNCHRONIZE ITS
                                ;       CONSOLE DLART TO THE BAUD RATE OF THE CONSOLE TERMINAL.
                                ;       ON POWER-UP, THE USER MUST TYPE A CARRIAGE RETURN CHARACTER.
                                ;       UPON SYNCHRONIZATION, AUTOBAUD WILL PROCEED TO ODT WHERE AN '@'
                                ;       CHARACTER WILL BE DISPLAYED ON THE CONSOLE.
                                ;
                                ;       AUTOBAUD WILL LOOP INDEFINITELY UNTIL SYNCHRONIZATION IS SUCCESSFUL.
                                ;
                                ;       THE ALGORITHM REQUIRES THAT THE CONSOLE TERMINAL GENERATES A
                                ;       ZERO (SPACE) FOR THE EIGHTH BIT IN THE CARRIAGE RETURN.  THIS
                                ;       WILL HAPPEN IF THE TERMINAL IS CAPABLE OF SENDING EIGHT-BIT-
                                ;       NO-PARITY OR SEVEN-BIT-ODD-PARITY CHARACTERS.
                                ;
                                ; ENVIRONMENT:
                                ;
                                ;       INTERRUPTS MUST BE DISABLED FOR THE ALGORITHM TO EXECUTE CORRECTLY
                                ;       SINCE TIME DURATIONS ARE CRITICAL AND DELAYS DUE TO LONG
                                ;       SERVICE ROUTINES MAY CAUSE DLART OVERRUNS, WHICH THIS ROUTINE
                                ;       IGNORES BUT CANNOT TOLERATE.
                                ;


                                ; VT103/FALCON CONFIGURATIONS LEAVE GARBAGE IN THE DLART LONG AFTER THE
                                ; POWERUP SEQUENCE HAS BEGUN.  WE MUST DELAY A BIT BEFORE CLEARING GARBAGE
                                ; OUT OF THE DLART, OTHERWISE THE GARBAGE WOULD ARRIVE AFTER THE CLEAR
                                ; (I.E., WHILE POLLING FOR INPUT).  THE "GARBAGE" IS AN X-ON (<CTRL-Q>)
                                ; THAT THE VT-100 HARDWARE SENDS AFTER ITS POWER-UP DIAGNOSTICS HAVE
                                ; COMPLETED SUCCESSFULLY.
                                ;

170512                          AUTOBA::
170512  012737  000032  177564          MOV     #BAUDRS,@#XCSR$1        ;SET 2400 BAUD
170520  005000                          CLR     R0                      ;DELAY
170522  077001                          SOB     R0,.                    ;       .5
170524  077001                          SOB     R0,.                    ;               SECONDS
```

```
                                 ; AUTOBAUD PROPER:

170526  105737  177562           10$:    TSTB    @#RBUFS1                ; DISCARD ANY GARBAGE

170532  105737  177560           20$:    TSTB    @#RCSRS1                ; WAIT FOR INPUT
170536  100375                           BPL     20$
170540  113700  177562                   MOVB    @#RBUFS1, R0            ; R0 = INPUT CHARACTER
170544  012701  170570                   MOV     #INBYTE, R1             ; R1 -> SCRAMBLED CHAR TABLE
170550  120021                   30$:    CMPB    R0, (R1)+               ; IN THE TABLE?
170552  001411                           BEQ     HVBAUD                  ; YES
170554  020127  170576                   CMP     R1, #INBYTS             ; END OF TABLE REACHED?
170560  001373                           BNE     30$                     ; NOT YET
170562  005000                           CLR     R0                      ; UH OH, WAIT FOR DLART TO CLEAR OUT
170564  077001                   40$:    SOB     R0, 40$                 ; WAIT FOR A WHILE
170566  000757                           BR      10$                     ; AND TRY FOR ANOTHER CHARACTER

                                 ; TABLE OF WHAT YOU WOULD SEE IF AN OCTAL 15 WERE SENT AT THE FOLLOWING
                                 ; BAUD RATES.

170570                           INBYTE:
170570     200                           .BYTE   200                     ; 300
170571     170                           .BYTE   170                     ; 600
170572     346                           .BYTE   346                     ; 1200
170573     015                           .BYTE   15                      ; 2400
170574     362                           .BYTE   362                     ; 4800
170575     377                           .BYTE   377                     ; 9600, 19200, 38400
170576                           INBYTS:

                                 ; WE HAVE A MATCH.  SET BAUD RATE INTO DLART.

170576                           HVBAUD:
170576  162701  170571                   SUB     #INBYTE+1, R1           ; TURN POINTER INTO BIT MASK
170602  006301                           ASL     R1
170604  006301                           ASL     R1
170606  005201                           INC     R1                      ; TURN ON XC.PBE
170610  006301                           ASL     R1                      ; SET THE BAUD RATE
170612  010137  177564                   MOV     R1, @#XCSRS1            ; INTO CSR
170616  005000                           CLR     R0                      ; DELAY .24 SECONDS FOR REST
170620  077001                           SOB     R0,.                    ; OF CHAR. AT SLO BAUD RATES

                                 ; FALL INTO ODT.
```

```
                                .SBTTL  MACROODT-INTRODUCTION

        ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
        ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
        ;;;;                                                        ;;;;
        ;;;;                        MACROODT                        ;;;;
        ;;;;                                                        ;;;;
        ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
        ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

        ; MACROODT IS THE USER INTERFACE TO THE FUNCTIONS CONTAINED
        ; IN THE KXT11-A2 FIRMWARE PRODUCT.  IT INTERPRETS COMMANDS
        ; ENTERED VIA THE CONSOLE TERMINAL KEYBOARD (SEE TABLES BELOW)
        ; TO PERMIT THE USER TO LOAD A PROGRAM INTO MEMORY, EXECUTE
        ; IT AND DEBUG IT.

        ;   COMMAND
        ;   1- SLASH (/)
        ;      A-OPEN MEMORY LOCATION
        ;      B-OPEN GENERAL REGISTER
        ;      C-OPEN STATUS REGISTER
        ;   2- CARRIAGE RETURN (<CR>)
        ;      A-CHANGE AND CLOSE MEMORY LOCATION OR REGISTER
        ;      B-CLOSE WITHOUT CHANGE
        ;   3- LINE FEED (<LF>)
        ;      A- CHANGE AND CLOSE MEMORY LOCATION AND OPEN NEXT
        ;      B- CLOSE MEMORY LOCATION WITHOUT CHANGING AND OPEN NEXT
        ;   4- GO (G)
        ;   5- PROCEED (P)
        ;   6- EXECUTE I/O DIAGNOSTICS (X)
        ;   7- EXECUTE BOOTSTRAPS (D)
```

```
;SYNTAX OF COMMANDS LISTED ABOVE, SHOWING CONSOLE BEFORE,
;DURING AND AFTER THE TYPING OF THE COMMAND.
;       KEY: N-AN OCTAL INTEGER TYPED BY THE USER, ONLY
;               LAST 6 DIGITS SIGNIFICANT
;            X-A SINGLE OCTAL DIGIT
;            U-THE DIGITS 0 OR 1
;            ALL OTHER CHARACTERS ARE LITERALS
;
;    BEFORE         DURING              AFTER
;
;1A @               @N/                 @N/XXXXXX
;1B @               @RX/                @RX/XXXXXX
;1C @               @RS/                @RS/XXXXXX
;2A @N/XXXXXX       @N/XXXXXX N<CR>     @
;2A @RX/XXXXXX      @RX/XXXXXX N<CR>    @
;2A @RS/XXXXXX      @RX/XXXXXX N<CR>    @
;2A XXXXXX/XXXXXX XXXXXX/XXXXXX N<CR> @
;2B @N/XXXXXX       @N/XXXXXX <CR>      @
;2B @RX/XXXXXX      @RX/XXXXXX <CR>     @
;2B @RS/XXXXXX      @RS/XXXXXX <CR>     @
;2B XXXXXX/XXXXXX XXXXXX/XXXXXX <CR>  @
;3A @N/XXXXXX       @N/XXXXXX N<LF>     XXXXXX/XXXXXX
;3A XXXXXX/XXXXXX XXXXXX/XXXXXX N<LF> XXXXXX/XXXXXX
;3B @N/XXXXXX       @N/XXXXXX <LF>      XXXXXX/XXXXXX
;3B XXXXXX/XXXXXX XXXXXX/XXXXXX <LF>  XXXXXX/XXXXXX
;4  @               @NG
;5  @               @P
;6  @               @X                  XXXXXX
;                                       @
;7  @               @DDU
;7  @               @DXU
;7  @               @DYU
;7  @               @DD<CR>
;7  @               @DX<CR>
;7  @               @DY<CR>
```

```
                                        .SBTTL MACRODDT-SAVE STATUS AND PRINT PROMPT

                                ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
                                ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
                                ;;;;                                                    ;;;;
                                ;;;;      SAVE CONTEXT, PRINT MESSAGES AND PROMPT       ;;;;
                                ;;;;                                                    ;;;;
                                ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
                                ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

170622                          ODT::
170622  105737  177562                  TSTB    @#RBUFS1                ;CLEAR OUT CONSOLE GARBAGE

                                ; COPY THE RESTART TYPE WORD INTO USER AREA

170626  016767  007130  007140          MOV     R.TYPE,ODTWHY

                                ; PROTECT AGAINST STACK TIMEOUTS, BUT SAVE USER'S SP FIRST

170634  010667  007120                  MOV     SP,USERSP               ;SAVE USERS STACK POINTER
170640  012706  177744                  MOV     #ODTSTK,SP              ;LOAD NEW SP

                                ; SAVE REST OF USER PROGRAM'S CONTEXT

170644  016716  007110                  MOV     USERSP,(SP)             ;RESERVE LOCATION FOR R6
170650  010546                          MOV     R5,-(SP)                ;SAVE
170652  010446                          MOV     R4,-(SP)                ; ALL
170654  010346                          MOV     R3,-(SP)                ;  OF
170656  010246                          MOV     R2,-(SP)                ;   USER'S
170660  010146                          MOV     R1,-(SP)                ;    GENERAL
170662  010046                          MOV     R0,-(SP)                ;     REGISTERS
170664  010667  007066                  MOV     SP,RPOINT               ;POINTER TO R0

                                ; DETERMINE WHETHER "?" OR PC MESSAGE IS APPROPRIATE, AND PRINT IT

170670  005767  007066                  TST     R.TYPE                  ;DID WE GET A HALT OR BREAK?
170674  100004                          BPL     QODT                    ;NO-NEXT QUESTION
                                                                        ;YES-PRINT PC
170676  016700  007050                  MOV     SAVPC,R0                ;GET STOPPED PC
170702  004767  000764                  CALL    OCTST0                  ;TYPE THE PC ON TERMINAL
170706                          QODT:
170706  105767  007050                  TSTB    R.TYPE                  ;SEE IF RESTART OCCURRED
                                                                        ;(NXM ONLY-BIT 7 SET)
170712  100003                          BPL     KBDS                    ;TYPE PROMPT

                                ; HERE'S WHERE THE PROMPT GETS PRINTED, WITH OR WITHOUT LEADING "?"

170714                          KBDQ:
170714  012700  171750                  MOV     #MSGQ,R0                ; GET ? ADDRESS
170720  000402                          BR      PRINT                   ;TYPE IN MESSAGE
170722                          KBDS:
170722  012700  171751                  MOV     #MSGS,R0                ;GET PROMPT MESSAGE ADDRESS

170726  005067  007030          PRINT:  CLR     R.TYPE                  ;SO REENTRY GIVES NO ERROR MSG.
170732  106427  000300                  MTPS    #PRI6                   ;ALLOW BREAKS TO HAPPEN
170736  004767  000620                  CALL    PUTSTR                  ;TYPE THE PROMPT ALREADY
170742  005067  007002                  CLR     ODTFLG                  ;CLEAR FLAG FOR NEW ENTRY
```

```
                                          .SBTTL  MACROODT-GET ODT COMMAND

                                  ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
                                  ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
                                  ;;;;                                                      ;;;;
                                  ;;;;        INTERPRET FIRST CHARACTER OF COMMAND          ;;;;
                                  ;;;;                                                      ;;;;
                                  ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
                                  ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

                                  ; NOTE: FOLLOWING CALL GETCHR, THE CHARACTER (7 BIT ASCII)
                                  ;  APPEARS IN R2.
                                  ; NOTE: FOLLOWING CALL GETNUM, IF CARRY IS CLEAR, THE OCTAL INTEGER
                                  ;  WAS FOLLOWED BY A CARRIAGE RETURN.
                                  ; NOTE: ON EXIT TO LCSET OR FALLING THROUGH TO GO ROUTINE, R0 CONTAINS
                                  ;  THE ADDRESS TYPED IN.

170746  004767  000556                    CALL    GETCHR                  ;...INPUT CHARACTERS
170752  120227  000104                    CMPB    R2,#'D                  ;BOOTSTRAPS?
170756  001002                            BNE     1$                      ;NO
170760  000167  173014                    JMP     BOOTS                   ;YES

170764  120227  000130            1$:     CMPB    R2,#'X                  ;DIAGNOSTICS?
170770  001002                            BNE     2$                      ;NO
170772  000167  001012                    JMP     DIAGNO                  ;YES

170776  120227  000120            2$:     CMPB    R2,#'P                  ;PROCEED?
171002  001430                            BEQ     PCMD                    ;YES
171004  120227  000122                    CMPB    R2,#'R                  ;REGISTER?
171010  001465                            BEQ     RCMD                    ;YES
171012  120227  000060                    CMPB    R2,#'0                  ;OCTAL DIGIT?
171016  103736                            BLO     KBDQ                    ;NO,ERROR
171020  120227  000070                    CMPB    R2,#'8                  ;VALID DIGIT?
171024  103333                            BHIS    KBDQ                    ;NO,ERROR
171026  005000                            CLR     R0                      ;ITS A DIGIT
171030  004767  000576                    CALL    GETNUM                  ;GET REST OF THE DIGIT OR CMD
171034  103327                            BCC     KBDQ                    ;CR WAS ISSUED,ERROR

                                  ; THE LAST CHARACTER AT THE END OF THE NUMBER COULD BE A VALID COMMAND-
                                  ; LET'S CHECK:

171036  120227  000057                    CMPB    R2,#'/                  ;EXAMINE LOCATION?
171042  001511                            BEQ     LCSET                   ;YES
171044  120227  000107                    CMPB    R2,#'G                  ;GO TO?
171050  001321                            BNE     KBDQ                    ;NO,ERROR
```

```
; TABLE OF PERMISSABLE STATES
;
;       NO.     STATE              VALID INPUTS           COMMENT
;       1-      PROMPT @               0-7      -------->  DIGIT.
;                                      P        -------->  PROCEED.
;                                      R        -------->  REGISTER DESIGNATOR.
;                                      X        -------->  EXECUTE DIAGNOSTIC
;                                      D        -------->  BOOT FROM DEVICE
;       2-      @175620                0-7      -------->  ANOTHER DIGIT.
;              [INPUT DIGIT]           /        -------->  EXAMINE LOC.
;                                      G        -------->  GO FROM LOC N.
;       3-      @176000/000002         0-7      -------->  INPUT NEW VALUE.
;                                      LF       -------->  DISPLAY NEXT LOC.
;                                      CR       -------->  CLOSE LOC GO TO PROMPT.
;       4-      @200/000023 12         0-7      -------->  INPUT MORE DIGITS.
;                                      LF       -------->  SAVE DATA DISPLAY NEXT.
;                                      CR       -------->  SAVE DATA GO TO PROMPT.
;       5-      @R                     0-7      -------->  REGISTER NUMBER.
;                                      S        -------->  PSW.
;       6-      @R5                    /        -------->  EXAMINE.
;       7-      @R5/000024             0-7      -------->  INPUT NEW VALUE.
;                                      CR       -------->  CLOSE LOCATION.
;       8-      @R5/000024 16          0-7      -------->  MORE DIGITS INPUT
;                                      CR       -------->  SAVE VALUE GO TO PROMPT
;
```

```
                                                .SBTTL  MACRODDT- GO AND PROCEED

                                        ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
                                        ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
                                        ;;;;                                                      ;;;;
                                        ;;;;            PROCESS GO AND PROCEED ODT COMMANDS       ;;;;
                                        ;;;;                                                      ;;;;
                                        ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
                                        ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

171052  010067  006674                          MOV     R0,SAVPC                ;PUT SUPPLIED PC IN MEMORY LOCATION

                                        ; PREPARE THE ENVIRONMENT FOR THE GO COMMAND

171056  000005                                  RESET                           ;BUS INITIALIZE
171060  005067  006670                          CLR     SAVPS                   ;CLEAR PSW

                                        ; ENTRY POINT FOR THE PROCEED COMMAND

                                        ; FIRST, CHECK FOR VALID STACK:

171064                                  PCMD:

171064  016600  000014                          MOV     14(SP),R0               ;USER'S STACK POINTER
171070  005740                                  TST     -(R0)                   ;WHERE SAVPS WILL GO (SEE BELOW)
171072  000240                                  NOP                             ; (IN CASE OF TIME OUT)
171074  103403                                  BCS     1$                      ;NO GOOD.  TIMED OUT.
171076  005740                                  TST     -(R0)                   ;WHERE SAVPC WILL GO
171100  000240                                  NOP                             ;
171102  103004                                  BCC     2$                      ;SUFFICIENT STACK.

                                        ; EITHER STACK NO GOOD, SO SIMULATE A DOUBLE BUS TRAP WITHOUT LOSING THE
                                        ;       USER'S CONTEXT AS STORED IN THE ODT STACK.

171104  012767  000201  006662  1$:     MOV     #R.STAKIR.NXM,ODTWHY    ;SNEAKY! (R.TYPE UNTOUCHED-
                                                                        ; ONLY THE USER IMAGE OF IT)
171112  000700                          BR      KBDQ                    ;ERROR PROMPT.

                                        ; OR    STACK IS OK, SO RESTORE USER'S CONTEXT.

171114  012600                          2$:     MOV     (SP)+,R0                ;RESTORE
171116  012601                                  MOV     (SP)+,R1                ; ALL
171120  012602                                  MOV     (SP)+,R2                ;  OF
171122  012603                                  MOV     (SP)+,R3                ;   USER'S
171124  012604                                  MOV     (SP)+,R4                ;    GENERAL
171126  012605                                  MOV     (SP)+,R5                ;     REGISTERS

171130  106427  000340                          MTPS    #PRI7                   ;NO BREAKS ALLOWED UNTIL OUT OF
                                                                                ; ODT!
171134  042716  000001                          BIC     #BIT0,(SP)              ;ODD STACKS ARE TOO ODD FOR T-11
171140  011606                                  MOV     (SP),SP                 ;RESTORE USER SP
171142  005167  006616                          COM     IN.USR                  ;SET USER MODE
171146  016746  006602                          MOV     SAVPS,-(SP)             ;RESTORE PC AND PS TO ...
171152  016746  006574                          MOV     SAVPC,-(SP)             ;...STACK WHERE RTT WILL LOOK
171156  000006                                  RTT                             ;RETURN TO USERS PROGRAM
171160  000655                          HKBDQ:  BR      KBDQ                    ;HELP IN BR
171162  000657                          HKBDS:  BR      KBDS                    ;HELP IN BR
```

```
                                                  .SBTTL  MACROODT-REGISTER AND PS COMMAND

                                          ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
                                          ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
                                          ;;;;                                                      ;;;;
                                          ;;;;              PROCESS ODT REGISTER COMMANDS           ;;;;
                                          ;;;;                                                      ;;;;
                                          ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
                                          ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

                                          ; ENTRY POINT FOR RX AND RS COMMANDS

171164                                    RCMD:
171164  052767  000200  006556                    BIS     #RFLAG,ODTFLG           ;SET REGISTER FLAG
171172  004767  000420                            CALL    ONENUM                  ;GET REGISTER NUMBER
171176  103246                                    BCC     KBDQ                    ;A VALID CMD DID NOT FOLLOW
171200  120227  000123                            CMPB    R2,#'S                  ;IS IT THE RS?
171204  001412                                    BEQ     SWCMD                   ;YES,BRANCH
171206  120227  000057                            CMPB    R2,#'/                  ;EXAMINE?
171212  001240                                    BNE     KBDQ                    ;NO,ERROR
171214  020027  000007                            CMP     R0,#7                   ;>7?
171220  101235                                    BHI     KBDQ                    ;YES,ERROR
171222  001013                                    BNE     RCMD1                   ;IS IT EXACTLY SEVEN
171224  012700  177752                            MOV     #SAVPC,R0               ;YES,GET PC ADDRESS
171230  000413                                    BR      REGOUT                  ;DISPLAY

                                          ; STATUS REGISTER (PS) SELECTED:

171232                                    SWCMD:
171232  004767  000272                            CALL    GETCHR                  ;WHAT YOU WANT TO DO WITH RS?
171236  120227  000057                            CMPB    R2,#'/                  ;EXAMINE?
171242  001224                                    BNE     KBDQ                    ;NO,ERROR
171244  012700  177754                            MOV     #SAVPS,R0               ;GET ADDRESS WHERE PS IS
171250  000403                                    BR      REGOUT                  ;GO AND DISPLAY

171252  006300                            RCMD1:  ASL     R0                      ;SHIFT FOR OFFSET IN MEMORY
171254  066700  006476                            ADD     RPOINT,R0               ;GET EXACT ADDRESS OF REG.
171260  010067  006462                    REGOUT: MOV     R0,ODTLOC               ;STORE LOCATION
171264  000402                                    BR      LOCDSP                  ;DISPLAY
```

```
                                          .SBTTL MACROODT-EXAMINE AND DEPOSIT

                                  ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
                                  ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
                                  ;;;;                                                          ;;;;
                                  ;;;;  PROCESS ODT MEMORY AND REGISTER EXAMINE/DEPOSIT         ;;;;
                                  ;;;;                                                          ;;;;
                                  ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
                                  ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

                                  ; ODTLOC POINTS TO REGISTER OR MEMORY LOCATION
                                  ; FOLLOWING CALL GETNUM, IF CARRY IS CLEAR, CR FOLLOWED DIGIT,
                                  ; ODTFLG: IF REGISTER BIT SET INDICATES REGISTER IS BEING EXAMINED

                                  ;ENTRY FROM CMD ROUTINE AFTER LOC, VALUE IS GIVEN

171266  010067  006454            LCSET:  MOV     R0,ODTLOC               ;SAVE NEW LOCATION
171272  011000                    LOCDSP: MOV     (R0),R0                 ;GET DATA
171274  000240                            NOP                             ;SO NEXT INST, DOES NOT EXECUTE
                                                                          ;IF WE TIME OUT,
171276  103730                            BCS     HKBDQ                   ;PRINT "?" IF WE TIMED OUT
171300  004767  000372                    CALL    OCTSTR                  ;PRINT IT
171304  112702  000040                    MOVB    #SPACE,R2               ;PRINT A SPACE AFTER THE DATA
171310  004767  000226                    CALL    PUTCHR
171314  004767  000210                    CALL    GETCHR                  ;GET NEXT CHARACTER
171320  120227  000015                    CMPB    R2,#CR                  ;FINISH
171324  001716                            BEQ     HKBDS                   ;YES,CLOSE LOCATION
171326  120227  000060                    CMPB    R2,#'0                  ;DEPOSIT?
171332  103450                            BLO     4$                      ;NO,CHECK LF
171334  120227  000070                    CMPB    R2,#'8                  ;MAYBE!
171340  103307                            BHIS    HKBDQ                   ;NO,FORGET IT
171342  005000                            CLR     R0                      ;YES
171344  004767  000262                    CALL    GETNUM                  ;GET REST OF NUMBER
171350  103006                            BCC     1$                      ;CR FOUND, STORE NEW VALUE

171352  120227  000012                    CMPB    R2,#LF                  ;NOT CR, MUST BE LF
171356  001300                            BNE     HKBDQ                   ;PRINT ERROR MESSAGE
171360  105767  006364                    TSTB    ODTFLG                  ;IF LF, CANNOT BE REGISTER
171364  100675                            BMI     HKBDQ                   ;(ERROR EXIT)

                                  ;T-BIT FILTER,  THE T-BIT CAN BE SET FROM THE KEYBOARD VIA ODT,
                                  ;THIS CAN EITHER BE USEFUL FOR DEBUGGING OR DISASTROUS.  SO, YOU CAN
                                  ;DO IT ONLY IF YOU FIRST SET FILT.T IN O.CNTL (BIT 15).

171366  022767  177754  006352    1$:     CMP     #SAVPS,ODTLOC           ;ARE WE DIDDLING THE PS?
171374  001021                            BNE     3$                      ;NO, WE'RE NOT.
171376  042700  177400                    BIC     #^C<377>,R0             ;PS IS NOT A WORD.
171402  005767  006364                    TST     O.CNTL                  ;IS BIT 15 (FILT.T) SET?
171406  100402                            BMI     2$                      ;YES, THE FILTER'S DISABLED
171410  042700  000020                    BIC     #T.BIT,R0               ;KILL THE T-BIT

                                  ;2$:                                    ;FALL THRU TO PRIORITY 7
                                                                          ; FILTER
```

```
                                        ;PRIORITY-7 FILTER: UNLESS FILT.7 (BIT 7) IN O.CNTL IS SET, YOU CANNOT
                                        ;ACTUALLY SET THE PS TO PRIORITY 7 USING ODT FROM THE KEYBOARD.  THIS
                                        ;PROTECTS THE ABILITY TO BREAK.

171414  105767  006352          2$:     TSTB    O.CNTL                  ;ODT CONTROL WORD
171420  100407                          BMI     3$                      ;DO NOTHING-FILTER DISABLED
171422  105700                          TSTB    R0                      ;INTENDED NEW PS
171424  100005                          BPL     3$                      ;DO NOTHING-PRIORITY < 4
171426  032700  000100                  BIT     #BIT6,R0                ;CHECK AGAIN
171432  001402                          BEQ     3$                      ;DO NOTHING-PRIORITY < 6
171434  042700  000040                  BIC     #BIT5,R0                ;LOWER THE BOOM
171440  010077  006302          3$:     MOV     R0,@ODTLOC              ;STORE NEW VALUE
171444  120227  000012                  CMPB    R2,#LF                  ;GO ON TO NEXT LOCATION?
171450  001407                          BEQ     5$                      ;SURE, WHY NOT.
171452  000643                          BR      HKBDS                   ;GO TO PROMPT

171454  120227  000012          4$:     CMPB    R2,#LF                  ;IS A LF ISSUED
171460  001237                          BNE     HKBDQ                   ;NO,ERROR
171462  105767  006262                  TSTB    ODTFLG                  ;IS REGISTER FLAG SET
171466  100634                          BMI     HKBDQ                   ;YES, LF NOT PERMITTED
171470  112702  000015          5$:     MOVB    #CR,R2                  ;TO LINE UP CURSOR
171474  004767  000042                  CALL    PUTCHR                  ;SEND IT
171500  062767  000002  006240          ADD     #2,ODTLOC               ;GET ADDRESS OF NEXT LOC.
171506  016700  006234                  MOV     ODTLOC,R0               ;GET NEXT ADDRESS VALUE...
171512  004767  000160                  CALL    OCTSTR                  ;...AND PRINT IT
171516  112702  000057                  MOVB    #'/,R2                  ;SEND A SLASH BEFORE...
171522  004767  000014                  CALL    PUTCHR                  ;...SHOWING THE CONTENTS...
171526  000661                          BR      LOCDSP                  ;...OF THE LOCATION
```

```
                                          .SBTTL  MACROODT-GET AND ECHO CHARACTER

                                  ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
                                  ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
                                  ;;;;                                                        ;;;;
                                  ;;;;         CHARACTER INPUT AND ECHO SUBROUTINE            ;;;;
                                  ;;;;                                                        ;;;;
                                  ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
                                  ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

                                  ; GET A CHARACTER FROM THE CONSOLE KEYBOARD AND ECHO IT BACK
                                  ; EXACTLY AS RECEIVED INCLUDING PARITY BITS IF ANY.  RETURN WITH
                                  ; CHARACTER IN R2, EIGHTH BIT (AND HIGH BYTE) ZERO.

171530                            GETCHR:
171530  105737  177560                    TSTB    @#RCSR$1                ;CHARACTER READY?
171534  100375                            BPL     GETCHR                  ;BRANCH IF NOT AND KEEP TRYING
171536  113702  177562                    MOVB    @#RBUF$1,R2             ;TRANSFER CHARACTER
171542                            PUTCHR:
171542  105737  177564                    TSTB    @#XCSR$1                ;PRINTER READY
171546  100375                            BPL     PUTCHR                  ;NO, TRY AGAIN
171550  110237  177566                    MOVB    R2,@#XBUF$1             ;YES, XMIT CHARACTER
171554  042702  177600                    BIC     #^C<177>,R2             ;CLEAR PARITY
171560  000207                            RETURN                          ;CONTINUE
```

```
                                         .SBTTL  MACRODDT-TYPE ASCII STRING

                                ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
                                ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
                                ;;;;                                                    ;;;;
                                ;;;;            MESSAGE PRINT SUBROUTINE                ;;;;
                                ;;;;                                                    ;;;;
                                ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
                                ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

                                ; PRINT MESSAGE STARTING WITH CHARACTER POINTED TO BY R0 AND
                                ; ENDING WITH FIRST CHARACTER WITH EIGHTH BIT SET (THIS CHARACTER
                                ; IS NOT PRINTED).

171562                          PUTSTR:
171562  112002                          MOVB    (R0)+,R2                ;GET ASCII CHAR
171564  100413                          BMI     DONE                    ;IS IT THE END MARK?
171566  004767  177750                  CALL    PUTCHR                  ;NO, PRINT IT
171572  000773                          BR      PUTSTR                  ;MORE

                                ;ENTRY FOR CARRIAGE RETURN

171574                          PUTCLF:
171574  112702  000015                  MOVB    #CR,R2                  ;PRINT CR
171600  004767  177736                  CALL    PUTCHR                  ;FALL THRU AND PRINT LF

                                ;ENTRY FOR LF

171604  112702  000012          PUTLF:  MOVB    #LF,R2                  ;PRINT LF
171610  004767  177726                  CALL    PUTCHR
171614  000207                  DONE:   RETURN
```

```

                                      .SBTTL  MACRODDT-GET OCTAL DIGITS

                              ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
                              ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
                              ;;;;                                                        ;;;;
                              ;;;;                 NUMERIC INPUT ROUTINE                  ;;;;
                              ;;;;                                                        ;;;;
                              ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
                              ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

                              ; ON EXIT, R0 CONTAINS THE BINARY REPRESENTATION OF THE NUMBER ENTERED
                              ; IF THE CARRY BIT IS CLEAR, A <CR> FOLLOWED NUMBER
                              ; IF THE CARRY BIT IS SET, SOME OTHER CHARACTER FOLLOWED THE NUMBER,
                              ; POSSIBLY A COMMAND.

171616                        ONENUM:
171616  005000                        CLR     R0                      ;CLEAR ACCUMULATOR
171620                        NEXNUM:
171620  004767  177704                CALL    GETCHR                  ;GET DIGIT OR TERMINATOR
171624  120227  000015                CMPB    R2,#CR                  ;CLEAR CARRY AND RETURN
171630  001412                        BEQ     SRET                    ;IF <CR> WAS TYPED
171632                        GETNUM:
171632  162702  000070                SUB     #'8,R2                  ;CONVERT TO BINARY,..
171636  062702  000010                ADD     #'8-'0,R2               ;...AND TEST IF OCTAL OR NOT
171642  103007                        BCC     NOCT                    ;NOT VALID DIGIT.
171644  006300                        ASL     R0                      ;MAKE ROOM FOR NEW DIGIT
171646  006300                        ASL     R0                      ;DITTO
171650  006300                        ASL     R0                      ;DITTO
171652  050200                        BIS     R2,R0                   ;PUT IT IN PLACE
171654  000761                        BR      NEXNUM                  ;GET NEXT

171656  000241                SRET:   CLC                             ;CLEAR CARRY
171660  000207                        RETURN                          ;CONTINUE

171662  062702  000060        NOCT:   ADD     #'0,R2                  ;RESTORE ASCII BECAUSE,..
171666  000261                        SEC                             ;...POSSIBLE COMMAND
171670  000207                        RETURN                          ;CONTINUE
```

```
                                        .SBTTL  MACRODDT-OCTSTR--TYPE BINARY IN R0 AS ASCII

                                ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
                                ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
                                ;;;;                                                            ;;;;
                                ;;;;                    NUMERIC OUTPUT ROUTINE                  ;;;;
                                ;;;;                                                            ;;;;
                                ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
                                ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

                                ; PRINTS, AS A 6-DIGIT OCTAL INTEGER, THE VALUE OF THE BINARY
                                ; NUMBER IN R0.

171672  004767  177676          OCTST0: CALL    PUTCLF                  ;NEED CRLF AT ODT ENTRY
171676                          OCTSTR:
171676  010046                          MOV     R0,-(SP)                ;SAVE VALUE
171700  012746  000006                  MOV     #6,-(SP)                ;NO. OF CHARACTERS
171704  005002                          CLR     R2                      ;OUTPUT HOLD
171706  006100                  5$:     ROL     R0                      ;SHIFT MSB INTO LSB
171710  006102                          ROL     R2                      ;" " " " " " " " "
171712  062702  000060                  ADD     #'0,R2                  ;MAKE A DIGIT
171716  004767  177620                  CALL    PUTCHR                  ;OUTPUT A CHARACTER
171722  005316                          DEC     (SP)                    ;COUNT
171724  001406                          BEQ     10$                     ;DONE
171726  005002                          CLR     R2                      ;NEXT
171730  006100                          ROL     R0                      ;GET NEXT DIGIT INTO
171732  006102                          ROL     R2                      ;R2
171734  006100                          ROL     R0                      ;FIRST TWO BITS
171736  006102                          ROL     R2                      ;" "  " "" " "
171740  000762                          BR      5$                      ;CONTINUE
171742  005726                  10$:    TST     (SP)+                   ;CLEAR COUNT
171744  012600                          MOV     (SP)+,R0                ;ORIGINAL VALUE
171746  000207                          RETURN                          ;
```

```
                                                .SBTTL  MACRODDT-OUTPUT MESSAGES

                                  ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
                                  ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
                                  ;;;;                                                        ;;;;
                                  ;;;;                       MESSAGES                         ;;;;
                                  ;;;;                                                        ;;;;
                                  ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
                                  ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
                                        .NLIST  BEX
171750      077                   MSGQ: .ASCII  '?'                    ; ERROR MESSAGE
171751      015     012     100   MSGS: .ASCII  <CR><LF>'@'<200>       ; PROMPT
                                        .EVEN
                                        .LIST   BEX
```

```
                                                .SBTTL DIAGNOSTICS-FOR SLU2 AND PPI

                                ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
                                ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
                                ;;;;                                                        ;;;;
                                ;;;;                  DIAGNOSTIC MODULE                     ;;;;
                                ;;;;                                                        ;;;;
                                ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
                                ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

                                ; DIAGNOSE PPI IN MODE 0 WITH LOOPBACK CONNECTORS INSTALLED.
                                ; DIAGNOSE SLU2 INTERNAL CIRCUITRY (MAINTENANCE MODE) AND
                                ; SLU2 DRIVERS/RECEIVERS (WITH EXTERNAL LOOPBACK CONNECTOR).

                                ; LIST OF ERROR BIT DEFINITIONS TO RETURN TO USER.

 171756  000100                         .WORD   E.EXT
 171760  000010                         .WORD   E.INT
 171762                         ERRBIT:

                                ; LIST OF MASKS TO PUT IN XCSR$2.  PERFORM INTERNAL LOOPBACK
                                ; TEST FIRST, THEN EXTERNAL LOOPBACK TEST.

 171762  000000                         .WORD   0
 171764  000002                         .WORD   XC.PBE                  ; 300 BAUD
 171766  000006                         .WORD   XC.PBE ! XC.MNT         ; 300 BAUD AND MAINTENANCE

 171770                         INITS:

                                ; LIST OF PATTERN BYTES TO LOOP AROUND.
                                ; ALL BITS ON, ALTERNATING BITS, ALL BITS OFF.
                                ; NOTE: LAST BYTE MUST BE 0.

 171770     377     252     000 PATERN: .BYTE   377, 252, 0
                                        .EVEN

                                        .ENABL  LSB

 171774  000405                         BR      AROUN2                  ;SKIP OVER ENTRY POINT
```

```
                                        .SBTTL HARDWARE ENTRY POINT

                172000                  .=172000
     172000                     START::
     172000     000167  176254          JMP     PWRSUP
     172004                     RESTAR::
     172004     000167  176026          JMP     R.STRT
```

```
                                                .SBTTL DIAGNOSTICS-CONTINUED
172010                               AROUN2:

172010                               DIAGNO:
172010  012737  000221  176206               MOV     #MODE,@#PP.CWR          ; SET PROPER PPI MODE- LED
172016  012737  000017  176206               MOV     #LEDOFF,@#PP.CWR        ; MUST IMMEDIATELY BE TURNED
                                                                             ; OFF AS A CONSEQUENCE.
172024  005000                               CLR     R0                      ; ASSUME SUCCESS

                                     ; PERFORM PARALLEL PORT DIAGNOSTIC

172026  005001                               CLR     R1                      ; R1 = LOOP PATTERN
172030  110137  176202               1$:     MOVB    R1, @#PP.B              ; SEND IT OUT PORT B
172034  123701  176200                       CMPB    @#PP.A, R1              ; CHECK INPUT IN PORT A
172040  001402                               BEQ     2$                      ; BRANCH IF SAME
172042  052700  000001                       BIS     #E.PAR, R0              ; ELSE SET ERROR FLAG
172046  077110                       2$:     SOB     R1, 1$                  ; LOOP FOR ALL VALUES

                                     ; PERFORM SLU 2 DIAGNOSTIC

172050  012702  171762                       MOV     #ERRBIT, R2             ; R2->ERROR FLAGS
172054  012701  176540                       MOV     #RCSRS2, R1             ; R1 => SLU2
172060  016146  000002                       MOV     2(R1), -(SP)            ; IGNORE GARBAGE, MAKE TEMP
172064  012704  171770                       MOV     #INITS, R4              ; R4->INITIAL XCSR VALUE
172070  014461  000004               3$:     MOV     -(R4), 4(R1)            ; INIT XCSR
172074  001436                               BEQ     11$                     ; BRANCH IF DONE
172076  005742                               TST     -(R2)                   ; R2->NEXT ERROR FLAG
172100  012716  000010                       MOV     #8., (SP)               ; (SP)=BAUD RATE COUNTER
172104  012703  171770               4$:     MOV     #PATERN, R3             ; R3->PATTERNS
172110  005005                       5$:     CLR     R5                      ; INIT TIMEOUT COUNTER
172112  105761  000004               6$:     TSTB    4(R1)                   ; LOOP PATTERN AROUND
172116  100402                               BMI     7$                      ; BRANCH IF READY
172120  077504                               SOB     R5, 6$                  ; ELSE BUMP TIMEOUT COUNTER
172122  000422                               BR      10$                     ; BRANCH IF TIMEOUT

172124  111361  000006               7$:     MOVB    (R3), 6(R1)
172130  005005                               CLR     R5                      ; INITIALIZE TIMEOUT COUNTER
172132  105711                       8$:     TSTB    (R1)
172134  100402                               BMI     9$                      ; BRANCH IF READY
172136  077503                               SOB     R5, 8$                  ; ELSE BUMP TIMEOUT COUNTER
172140  000413                               BR      10$                     ; BRANCH IF TIMEOUT

172142  126113  000002               9$:     CMPB    2(R1), (R3)             ; COME BACK OK?
172146  001010                               BNE     10$                     ; NO, SET ERROR BIT & EXIT
172150  105723                               TSTB    (R3)+                   ; DONE ALL BIT PATTERNS?
172152  001356                               BNE     5$                      ; NO
172154  005316                               DEC     (SP)                    ; YES, DONE ALL BAUDS?
172156  001744                               BEQ     3$                      ; YES
172160  062761  000010  000004               ADD     #10, 4(R1)              ; NO, TO NEXT BAUD RATE
172166  000746                               BR      4$

172170  051200                       10$:    BIS     (R2),R0                 ; SET ERROR BIT
172172  005726                       11$:    TST     (SP)+                   ; RID OF TEMP
172174  004767  177472                       CALL    OCTST0                  ; PRINT ERROR FLAGS
172200  000167  176516                       JMP     KBD$                    ; AND JUST GET OUT.
                                             .DSABL  LSB
```

```
                                .SBTTL BOOTS-DESCRIPTION
                ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
                ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
                ;;;;                                                      ;;;;
                ;;;;                  BOOTSTRAP MODULE                    ;;;;
                ;;;;                                                      ;;;;
                ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
                ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

        000000          .REPT   0

                THIS IS A SHORT BOOTSTRAP PROGRAM DESIGNED TO HANDLE FLOPPY DISKS OR TU58
                TAPE CASSETTES IN EITHER OUR STANDARD BOOTABLE FORMAT OR IN THE STAND-ALONE
                VOLUME FORMAT (RT-11 ".SAV"-STRUCTURED FILES). THE BOOTSTRAP ALSO HANDLES
                RD51 AND RX50 WHICH CONFORM TO THE MSCP FORMAT. RL01 AND RL02 DRIVES ARE
                ALSO MADE BOOTABLE.

                THE BOOTSTRAP SEQUENCE IS AS FOLLOWS:

                1.  SINCE ENTRY IS EFFECTED BY TYPING D IN RESPONSE TO ODT PROMPT, GET NEXT
                    CHARACTER (D,X,Y,U,L).  GET OPTIONAL DEVICE NUMBER NEXT (DEFAULT IS 0).

                2.  IF FLOPPY BOOT IS SELECTED:

                    A.  ATTEMPT TO READ 512 BYTES FROM SPECIFIED UNIT OF THE FLOPPY
                        DISK, STARTING FROM LOGICAL BLOCK ZERO, INTO MEMORY LOCATIONS
                        STARTING AT 0 AT THE DENSITY OF THE MEDIUM PRESENT IN THE
                        DRIVE AT THE TIME.

                    B.  IF THE DRIVE IS NOT READY OR DOES NOT CONTAIN A BOOTABLE
                        MEDIUM, GO BACK TO ODT.

                3.  IF TU58 BOOT IS SELECTED, READ THE FIRST BLOCK FROM THE SELECTED
                    DRIVE INTO LOCATIONS STARTING AT 0.

                4.  IF THE FIRST BYTE READ INTO RAM IS 240 OCTAL, JUMP TO IT.  IF THE
                    FIRST BYTE IS 260 OCTAL, EXECUTE THE STAND-ALONE VOLUME LOADER,
                    USING THE SELECTED DEVICE AS INPUT.

                5.  IF RD/RX BOOT IS SELECTED, READ THE FIRST BLOCK FROM THE SELECTED
                    DRIVE INTO LOCATIONS STARTING AT 0. THE BOOT PROGRAM IS THEN
                    RUN FROM RAM LOCATION 0.

                6.  IF THE RL BOOT IS SELECTED, READ THE FIRST BLOCK FROM THE SELECTED
                    DRIVE INTO LOCATION 0. THE BOOT PROGRAM IS THEN RUN FROM RAM
                    LOCATION 0.

                        .ENDR
```

```
                    ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
                    ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
                    ;;;;                                                    ;;;;
                    ;;;;          EQUATES USED ONLY BY BOOTSTRAPS           ;;;;
                    ;;;;                                                    ;;;;
                    ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
                    ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

                            .SBTTL BOOTS-RX CONTROLLER DEFINITIONS

                    ; RX01/RX02 (RXV11,RXV21) REGISTER DEFINITIONS

        177170      RXCS=   177170                  ;CONTROL AND STATUS
        177172      RXDB=   RXCS+2                  ;DATA BUFFER

                    ; RX CONTROL AND STATUS BITS

        100000      RX$$ER= 100000                  ;ERROR
        040000      RX$$IN= 040000                  ;INITIALIZE CONTROLLER
        030000      RX$$XA= 030000                  ;EXTENDED ADDRESS BITS
        004000      RX$$02= 004000                  ;1 IF RX02; 0 IF RX01
        003000      RX$$XX= 003000                  ;UNUSED BITS
        000400      RX$$DE= 000400                  ;DENSITY (1=DOUBLE,0=SINGLE)
        000200      RX$$TR= 000200                  ;TRANSFER FUNCTION
        000100      RX$$IE= 000100                  ;INTERRUPT ENABLE
        000040      RX$$DN= 000040                  ;DONE
        000020      RX$$UN= 000020                  ;UNIT SELECT
        000016      RX$$FN= 000016                  ;FUNCTION SELECT
        000001      RX$$GO= 000001                  ;GO

                    ; RX FUNCTION CODES (IN RX$$FN) WITH GO BIT PRESET

        000001      RX$FIL= 0*2+RX$$GO              ;FILL BUFFER
        000003      RX$EMP= 1*2+RX$$GO              ;EMPTY BUFFER
        000005      RX$WRT= 2*2+RX$$GO              ;WRITE SECTOR
        000007      RX$RED= 3*2+RX$$GO              ;READ SECTOR
        000011      RX$STD= 4*2+RX$$GO              ;SET MEDIA DENSITY
        000013      RX$RST= 5*2+RX$$GO              ;READ STATUS
        000015      RX$WDD= 6*2+RX$$GO              ;WRITE SECTOR WITH DELETED DATA
        000017      RX$REC= 7*2+RX$$GO              ;READ ERROR CODE

                    ; RX ERROR CODES

        000400      RXE$UN= 000400                  ;UNIT SELECTED
        000200      RXE$DR= 000200                  ;DRIVE READY
        000100      RXE$DD= 000100                  ;DELETED DATA
        000040      RXE$DN= 000040                  ;DRIVE DENSITY
        000020      RXE$DE= 000020                  ;DENSITY ERROR
        000004      RXE$ID= 000004                  ;INITIALIZE DONE
        000001      RXE$CR= 000001                  ;CRC ERROR

                    ; MISCELLANEOUS DEFINITIONS

        000010      RETRY=  8.                      ;NUMBER OF RETRIES

                            .SBTTL BOOTS-TU58 DEFINITIONS AND PROTOCOL EQUATES
```

```
                        ; ABSOLUTE ADDRESS DEFINITIONS

        000002          FILNAM  = 000002                ;ADDRESS OF RAD50 FILENAME FOR
                                                        ; STAND-ALONE PROGRAM LOADING
        001000          DIRBUF  = 001000                ;START OF 512. WORD BUFFER USED
                                                        ; FOR RT-11 DIRECTORY OPERATIONS
                                                        ;  IN STAND-ALONE LOADING

                        ; TU58 ADDRESS DEFINITIONS

        176540          TISCSR  = RCSRS2                ;DL RECEIVER CONTROL AND STATUS
        176542          TISBFR  = RBUFS2                ;DL RECEIVER DATA BUFFER
        176544          TOSCSR  = XCSRS2                ;DL TRANSMITTER CONTROL AND STATUS
        176546          TOSBFR  = XBUFS2                ;DL TRANSMITTER DATA BUFFER


                        ; TU58 RADIAL SERIAL PROTOCOL CODES

                        ; FLAG BYTE DEFINITIONS:

        000001          R$$DAT  = ^B<00001>             ;DATA MESSAGE FLAG
        000002          R$$CTL  = ^B<00010>             ;CONTROL MESSAGE FLAG
        000004          R$$INT  = ^B<00100>             ;INITIALIZE FLAG
        000020          R$$CON  = ^B<10000>             ;CONTINUE FLAG
        000023          R$$XOF  = ^B<10011>             ;XOFF

                        ; CONTROL PACKET OPERATION CODES:

        000000          R$NOP   = 0.                    ;NO-OPERATION
        000001          R$INIT  = 1.                    ;INITIALIZE
        000002          R$READ  = 2.                    ;READ OPERATION
        000003          R$WRIT  = 3.                    ;WRITE OPERATION
        000004          R$COMP  = 4.                    ;COMPARE (NOP ON TU58)
        000005          R$POSI  = 5.                    ;POSITION OPERATION
        000006          R$ABRT  = 6.                    ;ABORT (NOP ON TU58)
        000007          R$DIAG  = 7.                    ;DIAGNOSE
        000010          R$GETS  = 8.                    ;GET STATUS
        000011          R$SETS  = 9.                    ;SET STATUS (NOP ON TU58)
        000012          R$GETC  = 10.                   ;GET CHARACTERISTICS
        000013          R$SETC  = 11.                   ;SET CHARACTERISTICS (NOP ON TU58)
        000100          R$END   = ^B<01000000>          ;*END MESSAGE

                        ; END PACKET SUCCESS CODES:

        000000          S$NORM  = 0.                    ;NORMAL SUCCESS
        000001          S$RETR  = 1.                    ;SUCCESS BUT WITH RETRIES
        177776          S$PART  = -2.                   ;PARTIAL OPERATION (END OF MEDIUM)
        177770          S$UNIT  = -8.                   ;INVALID UNIT NUMBER
        177767          S$CART  = -9.                   ;NO CARTRIDGE
        177765          S$WPRT  = -11.                  ;CARTRIDGE WRITE PROTECTED
        177757          S$DCHK  = -17.                  ;DATA CHECK ERROR
        177740          S$SEEK  = -32.                  ;SEEK ERROR (BLOCK NOT FOUND)
        177737          S$MOTR  = -33.                  ;MOTOR STOPPED
        177720          S$OPCD  = -48.                  ;INVALID OPERATION CODE
        177711          S$RECN  = -55.                  ;INVALID RECORD NUMBER

                                .SBTTL  BOOTS-RT11 DEFINITIONS AND EQUATES
```

```

                                ; RT-11 DIRECTORY STRUCTURE DEFINITIONS

        001000                  SEGALO  = DIRBUF                ;NUMBER OF SEGMENTS ALLOCATED
        001002                  NXTSEG  = DIRBUF+2              ;NUMBER OF NEXT LOGICAL SEGMENT
        001004                  HGHSEG  = DIRBUF+4              ;HIGHEST SEGMENT IN USE
        001006                  XTRBYT  = DIRBUF+6              ;NUMBER OF EXTRA BYTES PER ENTRY
        001010                  STRBLK  = DIRBUF+10             ;STARTING BLOCK# FOR FILES
                                                                ; IN THIS SEGMENT
        000016                  ENTSIZ  = 7*2                   ;SIZE OF A DIRECTORY ENTRY
        000010                  D.FLEN  = 10                    ;OFFSET TO FILE LENGTH IN ENTRY
        000400                  TENTAS  = 000400                ;FLAG FOR TENTATIVE FILE ENTRY
        001000                  EMPTYS  = 001000                ;FLAG FOR EMPTY AREA ENTRY
        002000                  PERMFS  = 002000                ;FLAG FOR PERMANENT FILE
        004000                  ENDSGS  = 004000                ;FLAG FOR END OF SEGMENT

                                ; RT-11 SYSTEM COMMUNICATIONS AREA DEFINITIONS

        000040                  RTSSTA  = 000040                ;START ADDRESS FOR PROGRAM
        000042                  RTSISP  = 000042                ;INITIAL STACK POINTER
        000044                  RTSJSW  = 000044                ;JOB STATUS WORD
        000046                  RTSUSR  = 000046                ;USR LOAD ADDRESS
        000050                  RTSHGH  = 000050                ;JOB HIGH MEMORY LIMIT
        000052                  RTSEMT  = 000052                ;(BYTE) EMT ERROR CODE
        000053                  RTSUER  = 000053                ;(BYTE) USER ERROR CODE
        000054                  RTSRMN  = 000054                ;BASE ADDRESS OF RESIDENT MONITOR
        000056                  RTSFCH  = 000056                ;(BYTE) CONSOLE FILL CHARACTER
        000057                  RTSFCT  = 000057                ;(BYTE) CONSOLE FILL COUNT
```

```
                                        .SBTTL BOOTS-RD/RX CONTROLLER DEFINITIONS

                                        ;REQUIRED DEFINITIONS FOR RD/RX50 BOOTSTRAP


                              ;UDA I/O PAGE ADDRESSES

        176150                UDAIP=176150                  ;ADDRESS OF INITIALIZE AND POLLING REG.
        176152                UDASA=UDAIP+2                 ;ADDRESS OF STATUS AND ADDRESS REG.

                              ;BIT DEFINITIONS FOR INITIALIZATION

        100000                ERR=100000                    ;ERROR
        040000                S4=040000                     ;STEP4
        020000                S3=020000                     ;STEP3
        010000                S2=010000                     ;STEP2
        004000                S1=004000                     ;STEP1
        000001                GO=000001                     ;GO
        100000                STEP=100000                   ;STEP INDICATOR

                              ;HOST COMMUNICATIONS AREA DEFINITIONS

        100000                OWN=100000                    ;OWNERSHIP OF PACKET ,0=HOST,1=UDA

                              ;COMMAND/RESPONSE PACKET DEFINITIONS

        000011                OP.ONL=11                     ;ON-LINE COMMAND OP-CODE
        000041                OP.RD=41                      ;READ COMMAND OP-CODE

        000004                P.UNIT=4.                     ;UNIT NUMBER OFFSET INTO PACKET
        000040                P.SHUN=32.                    ;SHADOW UNIT NUMBER
        000010                P.OPCD=8.                     ;OP-CODE OFFSET INTO PACKET
        000014                P.BCNT=12.                    ;BYTE COUNT OFFSET INTO PACKET
        000012                P.STS=10.                     ;END PACKET STATUS OFFSET
        177774                P.LEN=-4.                     ;LENGTH OF PACKET OFFSET

                                                            ;WRITABLE MEMORY USAGE
```

```
                                        .SBTTL BOOTS-RX01/RX02 BOOTSTRAP

                                ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
                                ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
                                ;;;;                                                        ;;;;
                                ;;;;                   FLOPPY BOOTSTRAP                     ;;;;
                                ;;;;                                                        ;;;;
                                ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
                                ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

                                ; THIS ROUTINE WILL BOOTSTRAP EITHER FLOPPY DRIVE, AT THE DENSITY OF THE
                                ; MEDIA MOUNTED IN THAT DRIVE.

172204                          RXBOOT:
172204  012746  177170                  MOV     #RXCS,-(SP)             ;NEED FLOPPY CSR FOR CHK240
172210  005737  177170                  TST     @#RXCS                  ;IF NOT THERE, TIME OUT VIA 4
172214  000240                          NOP                             ;TO ST173 AND RESET THE WORLD
172216  012701  001000                  MOV     #512.,R1                ;BYTE COUNT
172222  005000                          CLR     R0                      ;STARTING BLOCK NUMBER
172224  005004                          CLR     R4                      ;RAM BUFFER ADDRESS = 000000
172226  004767  000564                  CALL    DREAD                   ;LOAD IT ALL IN
```

```
                                          .SBTTL BOOTS-DISTINGUISHING TYPE OF BOOT BLOCK

                                   ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
                                   ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
                                   ;;;;                                                        ;;;;
                                   ;;;;      DISTINGUISH STANDARD FROM STAND-ALONE FROM        ;;;;
                                   ;;;;                NON-BOOTABLE VOLUMES.                   ;;;;
                                   ;;;;                                                        ;;;;
                                   ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
                                   ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

                                   ; THE CHK240 ROUTINE WILL REPEAT POWERUP SEQUENCE IF LOCATION 0 DOES NOT
                                   ; CONTAIN A VALID SECONDARY BOOTSTRAP (I.E., DOES NOT HAVE A 240 OR 260
                                   ; IN IT).  IT STARTS EXECUTION OF THE BOOTED PROGRAM IF THERE'S A 240,
                                   ; AND GOES TO THE STAND-ALONE PROGRAM LOADER IF THERE'S A 260.

172232                             CHK240:
172232  022737  000240  000000             CMP     #240,@#0                ;DID WE READ A VALID BOOTSTRAP?
172240  001410                             BEQ     1$
172242  022737  000260  000000             CMP     #260,@#0
172250  001447                             BEQ     STANDB                  ;STAND-ALONE VOLUMES START WITH 260
172252  004767  176162                     CALL    VECSET                  ;RESTORE WIPED-OUT VECTORS
172256                                     ABORT   <NO BOOT BLOCK ON VOLUME>

172262  012601                     1$:     MOV     (SP)+,R1                ;UNIT CSR ADDRESS
172264  012600                             MOV     (SP)+,R0                ;UNIT NUMBER
172266  005007                             CLR     PC                      ;STANDARD SECONDARY BOOTS
```

```
                                        .SBTTL BOOTS-TU58 BOOTSTRAP

                                ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
                                ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
                                ;;;;                                                        ;;;;
                                ;;;;            TU58 TAPE CASSETTE BOOTSTRAP                ;;;;
                                ;;;;                                                        ;;;;
                                ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
                                ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

                                        .ENABL  LSB
172270                          TUBOOT:
172270  012746  176540                  MOV     #TISCSR,-(SP)           ;CHK240 WANTS TU58 CSR
172274  012701  176544                  MOV     #TOSCSR,R1              ;R1 -> OUTPUT CSR FOR TU58 SERIAL LINE
172300  005003                          CLR     R3                      ;SET R3 = 0 (TWO NULLS)
172302  005211                          INC     @R1                     ;START TRANSMITTING BREAK TO TU58
172304  004767  001314                  CALL    CH8OUT                  ;SEND EIGHT NULLS
172310  105711                  1$:     TSTB    @R1                     ;IS TRANSMITTER READY AGAIN YET?
172312  100376                          BPL     1$                      ;IF PL NO - WAIT
172314  042711  000001                  BIC     #XC.BRK,@R1             ;ELSE STOP SENDING BREAK NOW
172320  012703                          MOV     (PC)+,R3                ;GET TWO INIT COMMANDS FOR TU58
172322     004     004                  .BYTE   RSSINT,RSSINT
172324  004715                          CALL    @R5                     ;AND TRANSMIT THEM
172326  005741                          TST     -(R1)                   ;DUMP ANY GARBAGE CHAR IN TISBUF
172330  105737  176540          2$:     TSTB    @#TISCSR                ;IS CHARACTER AVAILABLE FROM THE TU58?
172334  100375                          BPL     2$                      ;IF PL, NO - WAIT IN LOOP
172336  121127  000020                  CMPB    @R1,#RSSCON             ;IF SO, WAS IT A CONTINUE FLAG?
172342  001402                          BEQ     3$                      ;IF EQ, YES- GO AHEAD
172344                                  ABORT   <TU58 INITIALIZATION ERROR>

                                ; TU58 IS NOW INITIALIZED.  PREPARE TO READ BLOCK #0.

172350  005000                  3$:     CLR     R0                      ;BLOCK NUMBER = 0
172352  012701  001000                  MOV     #512.,R1                ;BYTE COUNT = ONE BLOCK
172356  004767  000212                  CALL    READZU                  ;ATTEMPT TO READ THE BLOCK
172362  100323                          BPL     CHK240                  ;IF PL, READ WAS SUCCESSFUL
172364                                  ABORT   <TU58 BLOCK 0 READ ERROR>
                                        .DSABL  LSB
```

```
                                        .SBTTL BOOTS-STAND-ALONE VOLUME BOOTSTRAP

                                ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
                                ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
                                ;;;;                                                       ;;;;
                                ;;;;            STAND-ALONE-VOLUME BOOTSTRAP               ;;;;
                                ;;;;                                                       ;;;;
                                ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
                                ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

                                ; THIS ROUTINE LOADS STAND-ALONE PROGRAMS (ASSUMED TO BE IN RT-11 .SAV
                                ; FILE FORMAT) FROM AN RT-11 FILE STRUCTURED TU58 CARTRIDGE.  IT IS
                                ; INVOKED IF THE FIRST WORD IN BLOCK 0 OF THE CARTRIDGE IS A 260.

172370                          STANDB:
172370  012700  000001                  MOV     #1,R0                   ;SET DIRECTORY SEGMENT #1
172374  006300                  GOR:    ASL     R0                      ;TWO BLOCKS PER SEGMENT
172376  022020                          CMP     (R0)+,(R0)+             ;ADD 4 TO R0, AS DIRECTORY STARTS
                                                                        ; IN BLOCK#6
172400  012701  002000                  MOV     #1024.,R1               ;PREPARE TO READ TWO BLOCKS
172404  012704  001000                  MOV     #DIRBUF,R4              ;INTO THE DIRECTORY BUFFER
172410  004767  000162                  CALL    READU                   ;READ THE SEGMENT
172414  100002                          BPL     2$                      ;IF PL, READ WAS SUCCESSFUL
172416                                  ABORT   <DIRECTORY READ ERROR>
172422  012704  001010          2$:     MOV     #STRBLK,R4              ;ELSE PREPARE TO PICK UP
                                                                        ; STARTING BLOCK
172426  012400                          MOV     (R4)+,R0                ;R0 = STARTING BLOCK FOR FILES
172430  010403                  CEN:    MOV     R4,R3                   ;SAVE POINTER TO CURRENT ENTRY
172432  032724  002000                  BIT     #PERMFS,(R4)+           ;IS THIS A PERMANENT FILE?
172436  001010                          BNE     4$                      ;IF BIT SET, YES - CHECK IF IT MATCHES
172440  022744  004000                  CMP     #ENDSGS,-(R4)           ;ELSE IS THIS THE END-OF-SEGMENT
                                                                        ; MARKER?
172444  001015                          BNE     EPB                     ;IF NE, NO - GO SKIP THIS ENTRY
172446  013700  001002                  MOV     @#NXTSEG,R0             ;ELSE GET NUMBER OF NEXT SEGMENT
172452  001350                          BNE     GOR                     ;IF NE, THERE IS ONE - GO READ IT
172454                                  ABORT   <FILE NOT FOUND>

172460  012705  000002          4$:     MOV     #FILNAM,R5              ;POINT TO RAD50 NAME OF DESIRED FILE
172464  022425                          CMP     (R4)+,(R5)+             ;CHECK FILE NAME, FIRST WORD
172466  001004                          BNE     EPB                     ;IF NE NOT DESIRED FILE

172470  022425                          CMP     (R4)+,(R5)+             ;...CHECK SECOND WORD OF FILENAME
172472  001002                          BNE     EPB                     ;IF NE NOT DESIRED ONE
172474  022425                          CMP     (R4)+,(R5)+             ;...FINALLY, CHECK EXTENSION
172476  001410                          BEQ     LOAD                    ;IF EQ, GOT IT - GO LOAD THIS
                                                                        ; ONE INTO MEMORY
172500  010304                  EPB:    MOV     R3,R4                   ;GET ENTRY POINTER BACK
172502  062704  000010                  ADD     #D.FLEN,R4              ;ADVANCE TO FILE SIZE OF ENTRY
172506  062400                          ADD     (R4)+,R0                ;UPDATE CURRENT FILE BASE
172510  022424                          CMP     (R4)+,(R4)+             ;AND SKIP TO NEXT FILE ENTRY
172512  063704  001006                  ADD     @#XTRBYT,R4             ;PLUS ANY EXTRA BYTES IN EACH ENTRY
172516  000744                          BR      CEN                     ;CONTINUE FILE SEARCH

172520  011401                  LOAD:   MOV     @R4,R1                  ;R1 = SIZE OF FILE IN BLOCKS
172522  000301                          SWAB    R1                      ; * 256. = WORD COUNT
172524  006301                          ASL     R1                      ; * 2 = BYTE COUNT
172526  004767  000042                  CALL    READZU                  ;READ THE PROGRAM FILE INTO MEMORY
```

```
-----> HALT AT PC=172456 INDICATES "FILE NOT FOUND"

172532  100002                  BPL     1$                      ;IF MI, ERROR IN READ-ABORT
172534                          ABORT   <STAND-ALONE FILE READ ERROR>
172540  013705  000040  1$:     MOV     @#RTSSTA,R5             ;GET PROGRAM START ADRS
172544  032705  000001          BIT     #1,R5                   ;IS ADRS EVEN?
172550  001402                  BEQ     STARTS                  ;IF EQ YES - OKAY
172552                          ABORT   <ILLEGAL TRANSFER ADDRESS>

172556                  STARTS:
172556  012601                  MOV     (SP)+,R1                ;PASS THE CSR ADDRESS
172560  112600                  MOVB    (SP)+,R0                ;GET UNIT NUMBER BOOTED
172562  013706  000042          MOV     @#RTSISP,SP             ;LOAD PROGRAM'S STACK POINTER
172566  005067  005204          CLR     TRAP4                   ;DISABLE TRAP TO 4 FEATURE
172572  000115                  JMP     @R5                     ;GO START PROGRAM EXECUTION

172574                  READZU:
172574  005004                  CLR     R4                      ;LOAD AT 0
172576  016602  000004  READU:  MOV     4(SP),R2                ;GET UNIT NUMBER
172602  105767  005162          TSTB    B.CNTL
172606  100503                  BMI     DREAD                   ;BIT 7 SET FOR RX01/RX02
172610  000167  000572          JMP     TREAD                   ;READ FROM TAPE


                                .SBTTL 173000G ENTRY POINT
```

```
             173000                              .=173000
     173000                             ST173::
     173000  106427  000340                      MTPS     #PRI7          ;CAN'T ASSUME ANYTHING HERE,
     173004  000005                              RESET                   ;BUT PWRSUP USUALLY DOES.
     173006  005000                              CLR      R0             ;DELAY FOR THE SAKE OF DLART,
     173010  077001                              SOB      R0,.           ;(MAINT. BIT CLEARED BY RESET
                                                                         ;JUST A LITTLE TOO LONG).
     173012  000167  175242                      JMP      PWRSUP
```

```
                .SBTTL BOOTS-CONTINUED
```

```
                                 .SBTTL BOOTS-RX01/RX02 READ ROUTINES

                         ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
                         ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
                         ;;;;                                                    ;;;;
                         ;;;;            FLOPPY DISK READ ROUTINES               ;;;;
                         ;;;;                                                    ;;;;
                         ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
                         ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

                         ; WITH REGISTERS SET UP AS BELOW, READ THE APPROPRIATE NUMBER OF
                         ; FULL SECTORS FROM THE FLOPPY, AT EITHER DENSITY, WITH EITHER
                         ; RXV21 DMA OR RXV11 PROGRAMMED I/O INTERFACE.
                         ;
                         ; R0: STARTING BLOCK NUMBER FOR TRANSFER.
                         ; R1: BYTE COUNT FOR TRANSFER
                         ; R2: UNIT NUMBER
                         ; R4: ADDRESS OF BUFFER TO RECEIVE DATA

                         .ENABL LSB
173016  010446           DREAD:  MOV     R4,-(SP)                ;SAVE BUFFER ADDRESS
173020  010046                   MOV     R0,-(SP)                ;SAVE STARTING LBN
173022  010146                   MOV     R1,-(SP)                ;SAVE BYTE COUNT

                         ; CHECK STATUS AND MEDIA DENSITY OF SELECTED DRIVE

173024  012701  177172           MOV     #RXDB,R1                ;SET UP R1 FOR BENEFIT OF RXGO
173030  005000                   CLR     R0                      ;INITIALIZE CURRENT UNIT/DENSITY WORD
173032  006002                   ROR     R2                      ;BIT 0 SET = UNIT 1
173034  103002                   BCC     1$
173036  052700  000020           BIS     #RX$SUN,R0              ;SET UNIT 1
173042  004567  000312   1$:     JSR     R5,RXGO                 ;START A READ STATUS OPERATION
173046  000013                   .WORD   RX$RST                  ; TO DETERMINE STATUS AND DENSITY
173050  111102                   MOVB    @R1,R2                  ;PICK UP LOW BYTE OF STATUS
173052  100402                   BMI     2$                      ;IF PL, DRIVE NOT READY
173054                           ABORT   <FLOPPY DRIVE NOT READY>
173060  032702  000040   2$:     BIT     #RXESDN,R2              ;CHECK MEDIA DENSITY
173064  001411                   BEQ     3$                      ;IF EQ, SINGLE DENSITY

                         ; DOUBLE DENSITY.
                         ; LOGICAL SECTOR NUMBER = LOGICAL BLOCK NUMBER * 2
                         ; SECTOR COUNT = BYTE COUNT/256.

173066  052700  000400           BIS     #RX$SDE,R0              ;SET DOUBLE DENSITY IN COMMAND
173072  012602                   MOV     (SP)+,R2                ;BYTE COUNT
173074  000302                   SWAB    R2                      ;DIVIDE BY 256
173076  012603                   MOV     (SP)+,R3                ;LBN
173100  006303                   ASL     R3                      ;MULTIPLY BY 2
173102  012704  000200           MOV     #128.,R4                ;WORDS PER SECTOR
173106  000410                   BR      4$

                         ; SINGLE DENSITY.
                         ; LOGICAL SECTOR NUMBER = LOGICAL BLOCK NUMBER * 4
                         ; SECTOR COUNT = BYTE COUNT/128.

173110  012602           3$:     MOV     (SP)+,R2                ;BYTE COUNT
173112  000302                   SWAB    R2                      ;DIVIDE BY 256
```

```
173114  006302                      ASL     R2                  ;AND MULTIPLY BY 2
173116  012603                      MOV     (SP)+,R3            ;LBN
173120  006303                      ASL     R3
173122  006303                      ASL     R3                  ;MULTIPLY BY 4
173124  012704  000100              MOV     #64.,R4             ;WORDS PER SECTOR

                            ; SET UP STACK AS FOLLOWS:
                            ;   0(SP) = LOGICAL SECTOR NUMBER
                            ;   2(SP) = SECTOR COUNT
                            ;   4(SP) = WORDS PER SECTOR
                            ;   6(SP) = BUFFER ADDRESS

173130  010446              4$:     MOV     R4,-(SP)            ;WORDS PER SECTOR
173132  010246                      MOV     R2,-(SP)            ;SECTOR COUNT
173134  010346                      MOV     R3,-(SP)            ;LOGICAL SECTOR NUMBER

                            ; START THE READ OPERATION.
                            ; THIS IS THE TOP OF THE LOOP.

173136  004567  000216      5$:     JSR     R5,RXGO             ;START A SECTOR READ
173142  000007                      .WORD   RX$RED

                            ; CONVERT LOGICAL SECTOR NUMBERS TO PHYSICAL TRACKS AND SECTORS.

173144  011603                      MOV     @SP,R3              ;GET LOGICAL SECTOR NUMBER
173146  012702  000010              MOV     #8.,R2              ;LOOP COUNT
173152  022703  006400      6$:     CMP     #26.*200,R3         ;DOES 26 GO INTO DIVIDEND?
173156  101002                      BHI     7$                  ;BRANCH IF NOT, C CLEAR (BHI => BCC)
173160  062703  171400              ADD     #-26.*200,R3        ;SUBTRACT 26 FROM DIVIDEND (C SET)
173164  006103              7$:     ROL     R3                  ;SHIFT DIVIDEND AND QUOTIENT
173166  005302                      DEC     R2                  ;DECREMENT LOOP COUNT
173170  003370                      BGT     6$                  ;BRANCH TILL DIVIDE DONE
173172  110302                      MOVB    R3,R2               ;COPY TRACK NUMBER
173174  105003                      CLRB    R3                  ;REMOVE TRACK NUMBER FROM REMAINDER
173176  000303                      SWAB    R3                  ;GET REMAINDER
173200  022703  000014              CMP     #12.,R3             ;C=1 IF 13<=R3<=25, ELSE C=0
173204  006103                      ROL     R3                  ;SECTOR*2 (2:1 INTERLEAVE)
                                                                ;(+1 (C) IF SECTOR 13-25)
173206  006302                      ASL     R2                  ;DOUBLE THE TRACK NUMBER
173210  060203                      ADD     R2,R3               ;SKEW THE SECTOR
173212  060203                      ADD     R2,R3               ; BY ADDING IN
173214  060203                      ADD     R2,R3               ;  6 * TRACK NUMBER
173216  006202                      ASR     R2                  ;UNDOUBLE THE TRACK NUMBER
173220  005202                      INC     R2                  ; AND MAKE IT 1-76 (SKIP TRACK 0
                                                                ;  FOR ANSI)
173222  162703  000032      8$:     SUB     #26.,R3             ;PUT SECTOR
173226  002375                      BGE     8$                  ; INTO RANGE
173230  062703  000033              ADD     #27.,R3             ;  1-26

                            ; READ THE SECTOR

173234  010311                      MOV     R3,@R1              ;SET SECTOR NUMBER
173236  004514                      JSR     R5,@R4              ;
173240  010211                      MOV     R2,@R1              ;SET TRACK NUMBER
173242  004514                      JSR     R5,@R4              ;PERFORM A SECTOR READ
173244  100002                      BPL     9$                  ;IF MI, ERROR
173246                              ABORT   <FLOPPY READ ERROR>
```

```
                                        ; EMPTY RXV11/RXV21 BUFFER INTO RAM

173252  004567  000102          9$:     JSR     R5,RXGO                 ;START EMPTY BUFFER FUNCTION
173256  000003                          .WORD   RXSEMP                  ; AND WAIT FOR TR
173260  032737  004000  177170          BIT     #RXSS02,@#RXCS          ;IS DMA AVAILABLE?
173266  001407                          BEQ     10$                     ;IF EQ NO - HANDLE AS RX01

                                        ; RX02 DMA OPERATION

173270  016611  000004                  MOV     4(SP),@R1               ;ELSE LOAD WORD COUNT
173274  004514                          JSR     R5,@R4                  ;WAIT FOR TR
173276  016611  000006                  MOV     6(SP),@R1               ;AND LOAD CURRENT BUS ADDRESS
173302  004514                          JSR     R5,@R4                  ;WAIT FOR DONE
173304  000410                          BR      12$

                                        ; RX01 PROGRAMMED I/O OPERATION

173306  016603  000004          10$:    MOV     4(SP),R3                ;GET WORD COUNT
173312  006303                          ASL     R3                      ;TURN WORD COUNT INTO BYTE COUNT
173314  016602  000006                  MOV     6(SP),R2                ;GET STARTING BUS ADDRESS
173320  111122                  11$:    MOVB    @R1,(R2)+               ;MOVE ONE BYTE FROM BUFFER TO MEMORY
173322  004514                          JSR     R5,@R4                  ;WAIT FOR TR OR DONE
173324  077303                          SOB     R3,11$                  ;LOOP FOR ALL BYTES IN FIRST SECTOR

                                        ; LOOP BACK IF NOT YET FINISHED

173326  016603  000004          12$:    MOV     4(SP),R3                ;GET WORD COUNT
173332  006303                          ASL     R3                      ;TURN INTO BYTE COUNT
173334  060366  000006                  ADD     R3,6(SP)                ;UPDATE BUS ADDRESS
173340  005216                          INC     @SP                     ;UPDATE LOGICAL SECTOR NUMBER
173342  005366  000002                  DEC     2(SP)                   ;DECREMENT SECTOR COUNT
173346  001273                          BNE     5$                      ;READ ANOTHER SECTOR
173350  062706  000010                  ADD     #8.,SP                  ;POP THE STACK
173354  000257                          CCC                             ;CLEAR CONDITION CODES
                                                                        ; TO SHOW SUCCESS.
173356  000207                          RETURN                          ;ALL DONE
                                        .DSABL LSB
```

```
                                        ; THE MAIN SUBROUTINE FOR SENDING DISK COMMANDS AND WAITING FOR
                                        ; THEIR COMPLETION.
                                        ;
                                        ; REGISTER USAGE:
                                        ;       R0 = DENSITY BIT ! UNIT SELECT BIT (PROTO FOR COMMANDS)
                                        ;       R1 = RXDB ADDRESS
                                        ;       R4 = RXGO TR/DONE TEST ROUTINE POINTER
                                        ;

173360  012504                          RXGO:   MOV     (R5)+,R4                ;COPY COMMAND WORD TO USE
173362  050004                                  BIS     R0,R4                   ;SET UNIT # AND DENSITY
173364  010437  177170                          MOV     R4,@#RXCS               ;START OPERATION
173370  010704                                  MOV     PC,R4                   ;COPY ADRS FOR LATER CALLS
173372  005741                                  TST     -(R1)                   ;R1 => RXCS
173374  032711  000240                  1$:     BIT     #RXSSTR!RXSSDN,@R1      ;WAIT FOR TR OR DONE
173400  001775                                  BEQ     1$                      ;IF EQ, NEITHER ARE TRUE YET
173402  005721                                  TST     (R1)+                   ;RESET R1 => RXDB AND CHECK FOR ERRORS
173404  000205                                  RTS     R5                      ;RETURN TO CALLER
```

```
                                        .SBTTL BOOTS-TU58 READ ROUTINES

                                ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
                                ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
                                ;;;;                                                        ;;;;
                                ;;;;            TU58 DECTAPE II READ ROUTINES               ;;;;
                                ;;;;                                                        ;;;;
                                ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
                                ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

                                ; STARTS A READ OPERATION ON THE TU58 BY TRANSMITTING A COMMAND PACKET
                                ;
                                ; INPUTS:
                                ;       R0 = STARTING BLOCK # FOR TRANSFER
                                ;       R1 = BYTE COUNT FOR TRANSFER
                                ;       R2 = UNIT NUMBER
                                ;       R4 = ADDRESS OF BUFFER TO RECEIVE DATA
                                ; OUTPUTS:
                                ;       R0, R1, R2 UNCHANGED
                                ; DESTROYS:
                                ;       R3, R4, R5

                                .ENABL  LSB
173406  010446                  TREAD:  MOV     R4,-(SP)                ;SAVE BUFFER ADDRESS
173410  005004                          CLR     R4                      ;INIT CHECKSUM
173412  012703  005002                  MOV     #10.+400+R$SCTL,R3      ;SET COMMAND FLAG AND LENGTH
173416  004767  000206                  CALL    CH2OUT                  ;OUTPUT TWO CHARS AND SET R5
173422  012703  000002                  MOV     #R$READ,R3              ;SEND READ COMMAND AND MODIFIER=0
173426  004715                          CALL    @R5
173430  010203                          MOV     R2,R3                   ;THEN UNIT NUMBER AND SWITCHES=0
173432  004715                          CALL    @R5
173434  005003                          CLR     R3                      ;PLUS A ZERO SEQUENCE NUMBER
173436  004715                          CALL    @R5
173440  010103                          MOV     R1,R3                   ;FOLLOWED BY THE BYTE COUNT
173442  004715                          CALL    @R5
173444  010003                          MOV     R0,R3                   ;AND THE BLOCK NUMBER
173446  004715                          CALL    @R5
173450  010403                          MOV     R4,R3                   ;FINALLY, TRANSMIT THE CHECKSUM
173452  004715                          CALL    @R5
```

```
                                ; NOW READY TO ACCEPT DATA MESSAGES FROM THE TU58

173454  012600                          MOV     (SP)+,R0                ;R0 -> DATA BUFFER
                                ;       CLC                             ;(CH2OUT LEAVES C CLEAR)
173456  006001                          ROR     R1                      ;R1 = WORD COUNT FOR TRANSFER
173460  004767  000116          1$:     CALL    7$                      ;GET FIRST WORD OF PACKET
173464  122703  000001                  CMPB    #RS$DAT,R3              ;IS THIS INDEED A DATA MESSAGE?
173470  001017                          BNE     3$                      ;IF NE NO - MAY BE END MESSAGE
173472  105003                          CLRB    R3                      ;ELSE CLEAR FLAGS
173474  000303                          SWAB    R3                      ;MOVE PACKET BYTE COUNT TO LOW BYTE
173476  106003                          RORB    R3                      ;AND CONVERT TO WORD COUNT
173500  160301                          SUB     R3,R1                   ;REMOVE FROM TRANSFER COUNT
173502  010305                          MOV     R3,R5                   ;AND COPY FOR LOOP COUNTER
173504  004767  000102          2$:     CALL    9$                      ;GET NEXT TWO WORDS
173510  010320                          MOV     R3,(R0)+                ;STORE IN BUFFER
173512  077504                          SOB     R5,2$                   ;LOOP FOR ENTIRE DATA MESSAGE
173514  004767  000044                  CALL    5$                      ;GET CHECKSUM AND COMPARE
173520  005701                          TST     R1                      ;HAVE ALL DATA RECORDS BEEN
                                                                        ; TRANSFERRED?
173522  001356                          BNE     1$                      ;IF NE NO
173524  004767  000052                  CALL    7$                      ;AND GET PROSPECTIVE
                                                                        ; END PACKET START
173530  004767  000056          3$:     CALL    9$                      ;GET OPCODE/SUCCESS BYTES
                                                                        ; OF END PACKET
173534  122703  000100                  CMPB    #RSEND,R3               ;IS THIS AN END PACKET?
173540  001402                          BEQ     4$                      ;IF NE NO - ABORT
173542                                  ABORT   <TU58 END PACKET MISSING>
173546  010300                  4$:     MOV     R3,R0                   ;SAVE SUCCESS CODE IN R0
173550  004767  000032                  CALL    8$                      ;READ REMAINDER OF END PACKET
173554  004767  000004                  CALL    5$                      ;AND CHECK ITS CHECKSUM
173560  000300                          SWAB    R0                      ;SET CC'S ON SUCCESS CODE OF TRANSFER
173562  000207                          RETURN                          ;RETURN TO CALLER

173564  004767  000064          5$:     CALL    CH2IN                   ;GET TWO CHECKSUM BYTES
173570  020403                          CMP     R4,R3                   ;DOES IT MATCH CALCULATED VALUE?
173572  001402                          BEQ     6$                      ;IF NE NO - ERROR
173574                                  ABORT   <TU58 CHECKSUM ERROR>

173600  000207                  6$:     RETURN                          ;ELSE RETURN WITH SUCCESS

173602  005004                  7$:     CLR     R4                      ;INIT CHECKSUM
173604  000402                          BR      9$                      ;AND GET THE FIRST WORD

173606  004717                  8$:     CALL    @PC                     ;READ 4 WORDS
173610  004717                          CALL    @PC
173612  004767  000036          9$:     CALL    CH2IN                   ;READ NEXT TWO BYTES
173616  060304                          ADD     R3,R4                   ;ADD INTO CHECKSUM
173620  005504                          ADC     R4                      ; WITH END-AROUND CARRY
173622  000207                          RETURN                          ;AND BACK TO CALLER
                                .DSABL  LSB
```

```
                                        ; CH2OUT -- WRITE TWO BYTES TO THE TU58
                                        ;
                                        ; WRITES TWO BYTES TO INTERFACE AND UPDATES CHECKSUM.
                                        ;
                                        ; INPUTS:
                                        ;       R3 = TWO BYTES TO BE OUTPUT; LOW BYTE FIRST
                                        ;       R4 = CURRENT CHECKSUM WORD
                                        ; OUTPUTS:
                                        ;       R3 UNCHANGED
                                        ;       R4 UPDATED TO NEW CHECKSUM
                                        ;       R5 POINTING TO CH2OUT ROUTINE FOR EASIER FUTURE CALLS

 173624                                 CHROUT:
 173624  004717                                 CALL    @PC                     ;ENTRY POINT TO OUTPUT 8 CHARACTERS
 173626  004717                                 CALL    @PC
 173630                                 CH2OUT:
 173630  010705                                 MOV     PC,R5                   ;SET R5 TO FOLLOWING ROUTINE ADRS
 173632  060304                                 ADD     R3,R4                   ;UPDATE CHECKSUM WORD
 173634  005504                                 ADC     R4                      ; WITH END-AROUND CARRY
 173636  004717                                 CALL    @PC                     ;REPEAT FOR BOTH CHARACTERS
 173640  105737  176544                 1$:     TSTB    @#TOSCSR                ;IS INTERFACE READY FOR OUTPUT?
 173644  100375                                 BPL     1$                      ;IF PL NO - WAIT
 173646  110337  176546                         MOVB    R3,@#TOSBFR             ;ELSE TRANSMIT CHARACTER TO TU58
 173652  000407                                 BR      CHRET                   ;MERGE WITH OTHER ROUTINE TO RETURN

                                        ; CH2IN -- READ TWO BYTES FROM THE TU58
                                        ; CHIN  -- READ A SINGLE BYTE FROM THE TU58
                                        ;
                                        ; INPUTS:
                                        ;       NONE.
                                        ; OUTPUTS:
                                        ;       R3 = CHARACTER(S) READ

 173654  004717                         CH2IN:  CALL    @PC                     ;READ TWO, NOT ONE
 173656  105003                         CHIN:   CLRB    R3                      ;AND ZERO OUT SPACE FOR NEW ONE
 173660  105737  176540                 1$:     TSTB    @#TISCSR                ;IS A CHARACTER AVAILABLE?
 173664  100375                                 BPL     1$                      ;IF PL NO
 173666  153703  176542                         BISB    @#TISBFR,R3             ;ELSE SET INTO REGISTER
 173672  000303                         CHRET:  SWAB    R3                      ;MOVE CURRENT CHARACTER OVER
 173674  000207                                 RETURN                          ;AND RETURN TO CALLER
```

```
        164000                              .=164000


                                                    .SBTTL BOOTS-PROGRAM ENTRY POINT

                                        ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
                                        ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
                                        ;;;;                                                     ;;;;
                                        ;;;;  BOOTSTRAP INITIALIZATION AND COMMAND INTERPRETER  ;;;;
                                        ;;;;                                                     ;;;;
                                        ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
                                        ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

                                        ; A 'D' WAS ENTERED IN RESPONSE TO THE ODT PROMPT, SO WE GET
                                        ; HERE AND EXPECT 'D','X','U','L' OR 'Y' NEXT, FOLLOWED BY A CR OR A
                                        ; UNIT NUMBER.  WE SET BITS UP IN B.CNTL AS FOLLOWS:
                                        ;
                                        ;    BIT  7: 0 = TU58
                                        ;            1 = RX01/02
                                        ;            USED BY STAND-ALONE VOLUME LOADER TO SELECT PROPER
                                        ;            READ ROUTINE.
                                        ;    BIT  0: DEVICE NUMBER
                                        ;
                                        ; NOTE: IF NO MEMORY WAS FOUND AT 000000, BIT 15 OF B.CNTL,
                                        ;   CALLED "NO.LOW" WILL BE SET AND THE BOOTSTRAPS WILL BE
                                        ;   DISABLED.

164000                                  BOOTS::
164000  012737  000072  176544                  MOV     #TUBAUD,@#TOSCSR        ;SET TU58 BAUD RATE

                                        ; JUMP HERE WITH ODT IF BOOTING TU58'S AT OTHER THAN DEFAULT BAUD RATE

164006                                  STTUBD::
164006  010667  013752                          MOV     SP,IN.USR               ;PERMIT HALTS AND BREAKS
                                                                                ; BY MAKING IN.USR NON-ZERO


164012  005004                                  CLR     R4                      ;ASSEMBLE NEW B.CNTL HERE
164014  004767  005510                          CALL    GETCHR                  ;KEYBOARD CHARACTER IN R2
164020  120227  000104                          CMPB    R2,#'D                  ;DD = TU58 CASSETTE
164024  001463                                  BEQ     UNINUM                  ;R4 IS CLEAR FOR DD
164026  012704  000200                          MOV     #DEVBIT,R4              ;R4 BIT 7 IS SET FOR DX, DY
164032  020227  000130                          CMP     R2,#'X                  ;DX = RX01 OR RX02
164036  001456                                  BEQ     UNINUM                  ;DY = RX01 OR RX02, THE CODE'S
164040  020227  000131                          CMP     R2,#'Y                  ; THE SAME- IT KNOWS BOTH DEN-
164044  001453                                  BEQ     UNINUM                  ;  SITIES, DMA, NON-DMA
164046  012704  000340                          MOV     #340,R4                 ;MASK FOR RD/RX
164052  020227  000125                          CMP     R2,#'U                  ;LOOK FOR "DU"
164056  001001                                  BNE     RDLIN                   ;IF NOT MUST BE DL
164060  000407                                  BR      RDUNIT                  ;IF YES GET RD/RX UNIT NO.
164062  012704  000300                  RDLIN:  MOV     #300,R4                 ;MASK FOR RL
164066  020227  000114                          CMP     R2,#'L                  ;DID OPERATOR TYPE DL ?
164072  001421                                  BEQ     RLUNIT                  ;IF YES FIND UNIT NO.
164074                                          ABORT   <ILLEGAL DEVICE NAME>   ;NON-EXISTANT UNIT
```

```

164100  004767  005424          RDUNIT: CALL    GETCHR                  ;FIND OUT UNIT NO.
164104  022702  000015                  CMP     #15,R2                  ;IS IT A <CR> ?
164110  001003                          BNE     CMSK                    ;NO MUST BE A NUMBER
164112  005002                          CLR     R2                      ;YES MUST BE UNIT 0
164114  000167  000124                  JMP     VECT                    ;GO TO RD/RX BOOT WITH VALID UNIT

164120  162702  000060          CMSK:   SUB     #60,R2                  ;MASK OUT ASCII CHARACTER
164124  022702  000003                  CMP     #3,R2                   ;NO MORE THAN 4 UNITS
164130  002433                          BLT     NOUNIT                  ;IF > 3 THEN ILLEGAL UNIT
164132  000167  000106                  JMP     VECT                    ;GO TO RD/RX WITH UNIT NO.

164136  004767  005366          RLUNIT: CALL    GETCHR                  ;READ UNIT NUMBER FOR RL
164142  022702  000015                  CMP     #15,R2                  ;IS IT A <CR>
164146  001003                          BNE     CKMSK                   ;IF NOT FIND UNIT NO.
164150  005002                          CLR     R2                      ;R0 HAS UNIT 0
164152  000167  000066                  JMP     VECT                    ;GO TO RL MASK

164156  162702  000060          CKMSK:  SUB     #60,R2                  ;MASK OUT ASCII CHARACTER
164162  022702  000003                  CMP     #3,R2                   ;IS UNIT NO. > 3 ?
164166  002414                          BLT     NOUNIT                  ;UNIT NO. TO HIGH
164170  000167  000050                  JMP     VECT                    ;GO TO BOOT MASK

164174  004767  005330          UNINUM: CALL    GETCHR                  ;GET DEVICE NUMBER OR CR
164200  022702  000015                  CMP     #15,R2                  ;IS IT CR?
164204  001410                          BEQ     UNIFOR+2                ;CR MEANS DRIVE 0
164206  162702  000060                  SUB     #'0,R2                  ;DRIVE 0?
164212  001405                          BEQ     UNIFOR+2                ;YUP.
164214  005302                          DEC     R2                      ;DRIVE 1?
164216  001402                          BEQ     UNIFOR                  ;YES, SKIP THE ABORT
164220                          NOUNIT:
164220                                  ABORT   <ILLEGAL UNIT NUMBER>

164224  005204                  UNIFOR: INC     R4                      ;FOR UNIT 1.
164226  110467  013536                  MOVB    R4,B.CNTL               ;SET DEVICE, UNIT INFORMATION.
164232  005767  013532                  TST     B.CNTL                  ;TEST NO.LOW
164236  100002                          BPL     4$                      ;WE HAVE LOW MEMORY
                                                                        ;WE DON'T, SO GO TO ODT
164240                                  ABORT   <NO LOW MEMORY, CAN'T BOOT>

                                ; BEFORE PROCEEDING, WE SET UP THE BUS TIMEOUT TRAP VECTOR, ENABLE
                                ; TRAP TO 4 EMULATION AND RESET THE BUS.  WE DO A DELAY (SEE
                                ; EXPLANATION BELOW) AND SET UP THE STACK SO THE STAND-ALONE BOOTER AND
                                ; DEVICE PRIMARY BOOTSTRAPS CAN GET THE INFORMATION THEY NEED PASSED
                                ; TO THEM IN R0 AND R1 (SEE CHK240, BELOW).

164244                          4$:
164244  012737  164362  000004  VECT:   MOV     #BADBOT,@#4             ;IF WE TIME OUT, WE WANT TO RE-
164252  012737  000300  000006          MOV     #PRI6,@#6               ;INITIALIZE EVERYTHING.
164260  000005                          RESET                           ;FOR NOW, INIT. THE BUS.

                                ;       NOTE: THE PREVIOUS INSTRUCTION ALSO AFFECTS SOME DEVICES
                                ;       WHICH PERFORM A LONG INITIALIZATION SEQUENCE, SUCH AS RX02'S,
                                ;       WHICH DO AN AUTOMATIC BOOT FROM DRIVE 0.  THE LONG DELAY BELOW
                                ;       IS NECESSARY IN ORDER TO ASSURE DRIVE 1 IS READY IF A BOOT
                                ;       IS DESIRED FROM IT.

```

```
-----> HALT AT PC=164242 INDICATES "NO LOW MEMORY, CAN'T BOOT"

164262                                 DELAY   R0,R1,9.                ;DELAY 2 SECONDS
164274  012706  177644                 MOV     #$STACK,SP              ;INITIALIZE THE STACK.
164300  010667  013472                 MOV     SP,TRAP4                ;SET UP TRAP-TO-4 EMULATION
                                                                       ;BY MAKING TRAP4 NON-ZERO
164304  012716  037776                 MOV     #37776,(SP)             ;SOME BOOTS NEED A MEMORY-TOP
                                                                       ; ADDRESS HERE, SO 8K WILL DO
164310  010200                         MOV     R2,R0                   ;PUT UNIT NUMBER IN R0
164312  010402                         MOV     R4,R2                   ;BOOT CONTROL WORD HERE
164314  042702  177776                 BIC     #^C<DEVNUM>,R2          ;WANT ONLY UNIT NO. IN R2
164320  010246                         MOV     R2,-(SP)                ;AND WE'LL SAVE IT TOO.

164322  022704  000340                 CMP     #340,R4                 ;MASK FOR RD/RX BOOT
164326  001411                         BEQ     DXDYBT                  ;GO DIRECTLY TO BOOT
164330  022704  000300                 CMP     #300,R4                 ;MASK FOR RL BOOT
164334  001410                         BEQ     RL02BT                  ;GO DIRECTLY TO BOOT

164336  105704                         TSTB    R4                      ;BIT 7 SET FOR RX01/02
164340  100002                         BPL     TU58BT                  ;ACTUALLY BMI RXBOOT
164342  000167  005636                 JMP     RXBOOT                  ;GO TO FLOPPY BOOT
164346  000167  005716         TU58BT: JMP     TUBOOT                  ;GO TO TU58 BOOT

164352  000167  000014         DXDYBT: JMP     RDBOOT                  ;GO TO RD/RX BOOT CODE

164356  000167  000216         RL02BT: JMP     RLBOOT                  ;GO TO RL BOOT CODE

164362                         BADBOT:
164362  012706  177644                 MOV     #$STACK,SP              ;RESTORE THE STACK
164366                                 ABORT   <UNEXPECTED TIMEOUT DURING BOOT>
```


KXT11-A5 1K FIRMWARE   MACRO M1200 04-NOV-83 13:50 PAGE 64

```
                                          .SBTTL BOOTS-RD/RX READ ROUTINE
                                   ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
                                   ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
                                   ;;;;                                                            ;;;;
                                   ;;;;                RD/RX READ ROUTINE                          ;;;;
                                   ;;;;                                                            ;;;;
                                   ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
                                   ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;


164372                             RDBOOT:

164372  012701  176150             UDAM:   MOV     #UDAIP,R1                    ;LOAD CSR INTO R1

                                   ;INIT THE UDA

164376                             BEGIN:
164376  010021                             MOV     R0,(R1)+                     ;INITIALIZE THE DATA
                                                                                ;CHANGE POINTER TO UDASA REG.
164400  012705  004000                     MOV     #S1,R5                       ;BEGIN WITH STEP ONE
164404  012703  164412                     MOV     #TABLE,R3                    ;SET UP TABLE POINTER
164410  000406                             BR      ILOOP                        ;GET POINTER TO DATA TABLE

                                   ;TABLE OF DATA POINTED TO BY R3

                                   ;FIRST FOUR WORDS ARE USED FOR INITIALIZATION

164412  100000                     TABLE:  .WORD   STEP                         ;BPL .+2  ;STEP 1-LENGTH AND VECTOR
164414  002154                             .WORD   RING                         ;BMI ILOOP  ;STEP 2-ADDRESS OF HOST
                                                                                ;COMMUNICATION AREA

164416  000000                             .WORD   0                            ;STEP 3- HIGH ADDRESS BITS
164420  000001                             .WORD   GO                           ;STEP 4- GO BIT TO ENABLE UDA

                                   ;THE NEXT BYTES ARE USED TO FILL COMMAND PACKET

164422     011                             .BYTE   OP.ONL                       ;ON-LINE COMMAND OP-CODE
164423     000                             .BYTE   0                            ;(NO BYTE COUNT)
164424     041                             .BYTE   OP.RD                        ;READ COMMAND OP-CODE
164425     002                             .BYTE   512./400                     ;BYTE COUNT (HIGH BYTE)


164426  005711                     ILOOP:  TST     (R1)                         ;CHECK FOR ERROR
164430  100760                             BMI     UDAM                         ;RESTART IF SET
164432  031105                             BIT     (R1),R5                      ;TEST FOR STEP BIT FROM UDA
164434  001774                             BEQ     ILOOP                        ;LOOP UNTIL EITHER BIT SETS
164436  012311                             MOV     (R3)+,(R1)                   ;WRITE WORD TO UDA FROM TABLE
164440  006305                             ASL     R5                           ;SHIFT TO NEXT STEP
164442  100371                             BPL     ILOOP                        ;LOOP IF NOT AT LAST STEP

                                                   ;PUT DISK ON-LINE AND READ FIRST BLOCK

164444  012704  000044                     MOV     #36.,R4                      ;RCMD2 WORD LENGTH SET TO 36
164450  005002                     SLOOP:  CLR     R2                           ;CLEAR COMMAND PACKET BUFFER (AND MORE)
164452  005022                     CLOOP:  CLR     (R2)+                        ;CLEAR EACH WORD
```

```
164454  020227  002154                    CMP     R2,#RING                   ;UNTIL R2 POINTS TO RING
164460  001374                            BNE     CLOOP

164462  010437  002064                    MOV     R4,@#RCMD2+P.LEN           ;WRITE LENGTH WITH SOME NUMBER 36.

164466  112337  002100                    MOVB    (R3)+,@#RCMD2+P.OPCD       ;LOAD OP-CODE FROM TABLE
164472  111337  002105                    MOVB    (R3),@#RCMD2+P.BCNT+1      ;LOAD BYTE COUNT
164476  010037  002074                    MOV     R0,@#RCMD2+P.UNIT          ;LOAD UNIT NUMBER
164502  005037  002102                    CLR     @#RCMD2+P.STS              ;NO MODIFIERS ALLOWED
164506  012722  002004                    MOV     #RMSG,(R2)+                ;LOAD RING WITH POINTER TO MESSAGE PACKET
164512  010522                            MOV     R5,(R2)+                   ;MARK PACKET OWNED BY UDA
164514  012722  002070                    MOV     #RCMD2,(R2)+               ;POINTER TO COMMAND PACKET
164520  010522                            MOV     R5,(R2)+                   ;MARK PACKET OWNED BY UDA
164522  016102  177776                    MOV     -2(R1),R2                  ;TELL UDA TO LOOK AT COMM AREA

164526  005737  002156          XDONE:    TST     @#RING+2                   ;LOOK IF MESSAGE PACKET RECEIVED
164532  100775                            BMI     XDONE                      ;WAIT FOR IT
164534  032737  000037  002016            BIT     #37,@#RMSG+P.STS           ;CHECK STATUS
164542  001402                            BEQ     10$                        ;CHECK FOR CONTROLLER ERROR
164544  000000                            HALT                               ;ERROR:CHECK RMSG+P.STS
164546  000711                            BR      RDBOOT                     ;REDO BOOT
164550  162704  000004          10$:      SUB     #4.,R4                     ;MAKE WORD LENGTH 32 FROM 36
164554  105723                            TSTB    (R3)+                      ;CHECK IF READ COMMAND SENT
164556  001734                            BEQ     SLOOP                      ;GO BACK IF BYTE COUNT ZERO

164560  022737  000240  000000            CMP     #240,@#0                   ;VALID BOOT BLOCK ?
164566  001402                            BEQ     SECBT                      ;YES, GO TO IT
164570                                    ABORT   <INVALID BOOT BLOCK>

164574  000137  000000          SECBT:    JMP     @#0                        ;START AT ADDRESS 0
```

```






                                         ;*MISCELLANEOUS DEFINITIONS
         000012                          LF=     12                ;;CODE FOR LINE FEED
         000015                          CR=     15                ;;CODE FOR CARRIAGE RETURN

                                         ;*GENERAL PURPOSE REGISTER DEFINITIONS
         000000                          R0=     %0
         000001                          R1=     %1
         000002                          R2=     %2
         000003                          R3=     %3
         000004                          R4=     %4
         000005                          R5=     %5
         000006                          R6=     %6
         000007                          R7=     %7
         000006                          SP=     %6
         000007                          PC=     %7


                                         ;*DATA BIT DEFINITIONS (BIT00 TO BIT15)
         100000                          BIT15=  100000
         040000                          BIT14=  40000
         020000                          BIT13=  20000
         010000                          BIT12=  10000
         004000                          BIT11=  4000
         002000                          BIT10=  2000
         001000                          BIT9=   1000
         000400                          BIT8=   400
         000200                          BIT7=   200
         000100                          BIT6=   100
         000040                          BIT5=   40
         000020                          BIT4=   20
         000010                          BIT3=   10
         000004                          BIT2=   4
         000002                          BIT1=   2
         000001                          BIT0=   1
```

```
                                .SBTTL  LOCAL DATA


        177524                  SWITCH  =       177524                  ; TERMINATOR SWITCH ADDRESS
                                                                        ; THIS IS WHERE TO READ THE
                                                                        ; SWITCH SETTINGS FOR THE
                                                                        ; TERMINATOR BOARD, BITS 0-11
                                                                        ; THIS IS A READ ONLY REGISTER
```

```
        .SBTTL  RLV-11 BOOTSTRAP

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;;                                                    ;;;;
;;;;                RLV-11 READ ROUTINE                 ;;;;
;;;;                                                    ;;;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

;*
;* THE RLV-11 BOOTSTRAP WILL USE R0 AS THE PRIMARY UNIT TO LOAD THE
;* SECONDARY BOOTSTRAP.
;*
;* THE RLV BOOTSTRAP WILL PERFORM THE FOLLOWING FUNCTIONS:
;*
;* SET THE DEFAULT DRIVE VALUE,ZERO, IN R0.
;* LOAD THE COMMAND AND STATUS REGISTER ADDRESS INTO R1 (174400).
;*
;* RESET THE DRIVE. RESET THE CONTROLLER.
;* READ THE HEADER FROM THE PRESENT TRACK.
;* WAIT TWO MINUTES FOR THE DISK TO COME READY.
;* CALCULATE THE DIFFERENCE TO TRACK ZERO AND DO SEEK.
;*
;* READ DATA FROM TRACK 0, SECTOR 0, HEAD 0, INTO MEMORY.
;* LOAD MEMORY WITH DATA STARTING AT ADDRESS 000000.
;* TEST LOCATION ZERO FOR A "NOP" INSTRUCTION AND IF CORRECT START.
;*
;* START THE SECONDARY BOOTSTRAP AT LOCATION ZERO.
;*
;* R0= UNIT NUMBER
;* R1= ADDRESS OF CSR.
;* R2= ERROR RETRY COUNTER.
;* R3= CONTENTS OF CSR.
;* R4= COMMAND WORD.
;* R5= ERROR STATUS REGISTER.
;*
```

```
;* CSR REGISTER VALUES FOR RLV.
;*
;* BIT 15        100 000 READ ONLY       ERROR
;* BIT 14        040 000 READ ONLY       DRIVE ERROR.
;* BIT 13        020 000 READ ONLY       NON-EXISTENT MEMORY.
;* BIT 12        010 000 READ ONLY       DATA LATE OR HEADER NOT FOUND.
;* BIT 11        004 000 READ ONLY       CRC ERROR ON HEADER OR DATA.
;* BIT 10        002 000 READ ONLY       OPERATION INCOMPLETE.
;* BIT 9 - 8     001 400 READ/WRITE      DRIVE SELECT BITS.
;*
;* BIT 7         000 200 READ/WRITE      CONTROLLER READY.
;* BIT 6         000 100 READ/WRITE      INTERRUPT ENABLE.
;* BIT 5         000 040 READ/WRITE      BUS ADDRESS EXTENSION
;* BIT 4         000 020 READ/WRITE      BUS ADDRESS EXTENSION.
;* BIT 3         000 010 READ/WRITE      FUNCTION BITS.
;* BIT 2         000 004 READ/WRITE      FUNCTION BITS.
;* BIT 1         000 002 READ/WRITE      FUNCTION BITS.
;* BIT 0         000 001 READ ONLY       DRIVE READY.
;*
;* FUNCTION BITS 1, 2, 3.
;* FUNCTION      BIT SETTING             MODE
;* 0             000 000                 MAINTENANCE MODE
;* 1             000 002                 WRITE CHECK
;* 2             000 004                 GET STATUS.
;* 3             000 006                 SEEK.
;* 4             000 010                 READ HEADER.
;* 5             000 012                 WRITE DATA
;* 6             000 014                 READ DATA.
;* 7             000 016                 READ DATA WITHOUT HEADER CHECK.
;*
;* DRIVE ERROR STATUS REGISTER.
;* BIT SETTING           ERROR
;* 15                    WRITE DATA ERROR.
;* 14                    HEADER CURRENT ERROR.
;* 13                    WRITE LOCK.
;* 12                    SEEK TIME OUT.
;* 11                    SPIN ERROR.
;* 10                    WRITE GATE ERROR.
;* 9                     VOLUME CHECK.
;* 8                     DRIVE SELECT ERROR.
;* 7                     RESERVED (SET TO A  0  ).
;* 6                     HEAD SELECT ERROR.
;* 5                     COVER OPEN.
;* 4                     HEADS OUT.
;* 3                     BRUSH HOME.
;*
;* 0-2           ONE OF THE FOLLOWING ERRORS.
;* 0                     LOAD STATE.
;* 1                     SPIN -UP
;* 2                     BRUSH CYCLE.
;* 3                     LOAD HEADS.
;* 4                     SEEK TRACK COUNTING.
;* 5                     SEEK LINEAR MOTOR.
;* 6                     UNLOAD HEADS.
;* 7                     SPIN DOWN.
```

```
164600  042700  000004          RLBOOT: BIC     #4,R0           ;UNIT NUMBER LIMIT 0-3
164604  005037  000000                  CLR     @#0             ;MAKE LOCATION 0 =ZERO,FOR CORRECT READING OF
                                                                ;NOP INSTRUCTION OF SECONDARY BOOT.
164610  012701  174400                  MOV     #174400,R1      ;R1=CSR ADDRESS
164614  012706  001100          RLREDO: MOV     #1100,SP        ;SET STACK POINTER



                                                ;INITIALIZE RLV BOOTSTRAP
164620                          RLINIT:
164620  112702  000010                  MOVB    #10,R2          ;R2=ERROR RETRY COUNTER
164624  010004                          MOV     R0,R4           ;STORE UNIT NUMBER FOR CSR DRIVE SETUP
164626  000304                          SWAB    R4              ;PLACE DRIVE NUMBER IN HIGH BYTE
                                                                ;SET ERROR RETRY COUNTER = 10
                                                                ;SET CORRECT DRIVE SELECT BITS IN COMMAND WORD R4.

                                                ;RESET CONTROLLER AND DRIVE

164630  004767  000152          10$:    JSR     PC,RESDRV       ;RESET DRIVE.
164634  103003                          BCC     20$             ;BRANCH IF CARRY = 0, NO ERROR.
164636  077204                          SOB     R2,10$          ;CARRY = 1 ERROR, SUBTRACT 1 FROM ERROR RETRY
                                                                ;COUNTER AND RETRY UNTIL COUNTER IS ZERO.
                                                ;*************************************************************
                                                ;CANNOT CLEAR ERROR FLAGS AFTER CONTROLLER AND DRIVE RESET.
                                                ;RLERR1 ROUTINE WILL "HALT" OR JUMP TO EPROM ON ERROR.
164640  004767  000340                  JSR     PC,RLERR1       ;R0=DRIVE SELECTED.
                                                                ;R1=CSR ADDRESS.
                                                                ;R2=000000 ERROR RETRY COUNTER
                                                                ;R2=RLV BOOTSTRAP PAGE(IF EPROM SELECTED).
                                                                ;R3=CONTENTS OF CSR REGISTER
                                                                ;R4=COMMAND WORD
                                                ;*************************************************************

                                                ;READ HEADER

164644  012702  002200          20$:    MOV     #1152.,R2       ;ERROR RETRY COUNTER WAIT FOR > TWO
                                                                ;MINUTES OR DRIVE READY.
164650  004767  000156                  JSR     PC,RDHEAD       ;READ HEADER ON PRESENT CYLINDER
                                                                ;RETURN WITH HEADER INFORMATION IN R5.

                                                ;CALCULATE DIFFERENCE TO CYLINDER 0 AND SEEK


                                                ;SEEK ROUTINE
                                                                ;R5 MUST CONTAIN PRESENT HEADER INFO
                                                                ;TO PERFORM A SEEK DIFFERENCE MUST BE
                                                                ;LOADED INTO DISK ADDRESS (RLDA) WITH
                                                                ;BIT 0 SET AND BIT 1 CLEARED.
                                                                ;CYLINDER DIFFERENCE IN BITS 7-14 RL01
                                                                ;CYLINDER DIFFERENCE IN BITS 7-15 RL02

164654  012702  000010          SEEK:   MOV     #10,R2          ;RESET ERROR RETRY COUNTER

164660  042704  000016          10$:    BIC     #16,R4          ;CLEAR FUNCTION BITS
```

```
164664  052704  000006          BIS     #6,R4           ;MAKE COMMAND WORD FUNCTION BITS= 3

164670  042705  000177          BIC     #177,R5         ;CLEAR THE FOLLOWING BITS IN THE HEADER WORD.

                                                        ;THIS WILL LEAVE ONLY THE CYLINDER ADDRESS.

164674  005205                  INC     R5              ;BIT0 MUST BE SET IN (DA) TO DO A SEEK.
164676  010561  000004          MOV     R5,4(R1)        ;SET CYLINDER DIFFERENCE IN DISK ADDRESS

164702  004767  000230          JSR     PC,RLGO         ;DO FUNCTION, RETURN WITH FOLLOWING
164706  103005                  BCC     READ            ;CARRY = 0 NO ERRORS DO BRANCH
                                                        ;CARRY = 1 RETRY 10 TIMES
164710  004767  000116          JSR     PC,RDHEAD       ;FIRST READ HEADER
164714  077217                  SOB     R2,10$          ;SUBTRACT 1 FROM ERROR RETRY COUNTER
                                                        ;AND IF NOT ZERO RETRY SEEK.

                                                ;*****************************************************************
                                                ;SEEK ERROR

164716  004767  000262          JSR     PC,RLERR1       ;A LOAD DEVICE ERROR HAS OCCURRED,JUMP TO
                                                        ;ERROR ROUTINE. TEST FOR FURTHER PROGRAM
                                                        ;DIALOGUE OR EXECUTE A "HALT".
                                                        ;IF "HALT" IS EXECUTED AND THE USER "PROCEEDS"
                                                        ;THE PROGRAM WILL RERUN THE BOOTSTRAP.
                                                        ;IF THE "EPROM" IS SELECTED REFER TO
                                                        ;BOOTSTRAP ERROR ROUTINE FOR MORE INFORMATION
                                                        ;ON "EPROM" EXECUTION.
                                                        ;ERROR REGISTER CONTAIN
                                                        ;R0 = DRIVE UNIT
                                                        ;R1 = CSR ADDRESS
                                                        ;R2 = ERROR RETRY COUNTER
                                                        ;R3 = CONTENTS OF CSR
                                                        ;R4 = COMMAND WORD
                                                        ;R5 = DRIVE STATUS IF DRIVE ERROR.
                                                        ;R5 = CONTENTS OF CSR WITH CONTROLLER ERROR.
                                                ;*****************************************************************

                                                ;CLEAR FUNCTION BITS IN COMMAND WORD.

                                                ;READ DATA FROM CYL 0 HEAD 0 SECTOR 0

                                                ;READ DATA INTO MEMORY STARTING AT 000000.


                                                ;READ CYL 0 SECTOR 0 HEAD 0


164722  012702  000010  READ:   MOV     #10,R2          ;RESET ERROR RETRY COUNTER
                                                        ;TO DO A READ THE FOLLOWING MUST BE SET-UP

164726  042704  000016  10$:    BIC     #16,R4          ;CLEAR FUNCTION BITS

164732  052704  000014          BIS     #14,R4          ;MAKE COMMAND WORD FUNCTION BITS=6.

164736  005061  000002          CLR     2(R1)           ;BASE ADDRESS = 000 000

164742  005061  000004          CLR     4(R1)           ;DISK ADDRESS = SECTOR 0 - HEAD 0 - CYL 0 -
```

```
164746  012761  177400  000006          MOV     #177400,6(R1)   ;(MP)WORD COUNT = 400 ( 177400 TWO'S COMPLEMENT VALUE NEEDED
164754  004767  000156                  JSR     PC,RLGO         ;DO READ, GO WILL RETURN WITH THE FOLLOWING,
164760  103003                          BCC     TST24           ;CARRY = 0 NO ERROR
164762  077217                          SOB     R2,10$          ;CARRY = 1 ERROR RETRY 10 TIMES
                                                        ;*******************************************************************
                                                                ;ERROR COULD NOT READ CYL 0 SECTOR 0 HEAD 0
164764  004767  000214                  JSR     PC,RLERR1       ;A LOAD DEVICE ERROR HAS OCCURRED,JUMP TO
                                                                ;ERROR ROUTINE. TEST FOR FURTHER PROGRAM
                                                                ;DIALOGUE OR EXECUTE A "HALT".
                                                                ;IF "HALT" IS EXECUTED AND THE USER "PROCEEDS"
                                                                ;THE PROGRAM WILL RERUN THE BOOTSTRAP.
                                                                ;IF THE "EPROM" IS SELECTED REFER TO
                                                                ;BOOTSTRAP ERROR ROUTINE FOR MORE INFORMATION
                                                                ;ON "EPROM" EXECUTION.
                                                                ;ERROR REGISTER CONTAIN;
                                                                ;R0 = DRIVE UNIT
                                                                ;R1 = CSR ADDRESS
                                                                ;R2 = ERROR RETRY COUNTER
                                                                ;R3 = CONTENTS OF CSR
                                                                ;R4 = COMMAND WORD
                                                                ;R5 = DRIVE STATUS IF DRIVE ERROR.
                                                                ;R5 = CONTENTS OF CSR IF CONTROLLER ERROR.
                                                        ;*******************************************************************

                                                        ;ROUTINE TO CLEAR THE FUNCTION BITS IN THE
                                                        ;TEST FOR PROPER BOOTSTRAP INDICATOR
                                                        ;"NOP" (000240) IN LOCATION IS STANDARD
                                                        ;INDICATOR FOR A SECONDARY BOOTSTRAP.
                                                        ;IF NOP IS IN LOCATION ZERO THEN START IT.
                                                        ;TEST FOR SECONDARY BOOTSTRAP.

164770  022737  000240  000000  TST24:  CMP     #240,@#0        ;TEST LOCATION ZERO FOR A "NOP" (240)
164776  001402                          BEQ     SECBOT          ;BRANCH IF NOP THERE

                                                        ;*******************************************************************
                                                        ;SECONDARY BOOTSTRAP REQUIRES A NOP IN LOCATION 0.  NONE THERE.

165000  004767  000200                  JSR     PC,RLERR1       ;A LOAD DEVICE ERROR HAS OCCURRED,JUMP TO
                                                                ;ERROR ROUTINE. TEST FOR FURTHER PROGRAM
                                                                ;DIALOGUE OR EXECUTE A "HALT".
                                                                ;IF "HALT" IS EXECUTED AND THE USER "PROCEEDS"
                                                                ;THE PROGRAM WILL RERUN THE BOOTSTRAP.
                                                                ;IF THE "EPROM" IS SELECTED REFER TO
                                                                ;BOOTSTRAP ERROR ROUTINE FOR MORE INFORMATION
                                                                ;ON "EPROM" EXECUTION.
                                                                ;ERROR REGISTER CONTAIN;
                                                                ;R0 = UNIT NUMBER
                                                                ;R1 = CSR ADDRESS
                                                                ;R2 = ERROR RETRY COUNTER
                                                                ;R3 = CONTENTS OF CSR
                                                                ;R4 = COMMAND WORD
                                                                ;R5 = DRIVE STATUS
                                                        ;*******************************************************************
```

```
165004  005007                    SECBOT: CLR     PC              ;START AT LOCATION 0



                                                          ;RE-SET DRIVE AND THEN READ STATUS
165006                            RESDRV:
165006  042704  000016                    BIC     #16,R4          ;CLEAR FUNCTION BITS

165012  052704  000004                    BIS     #4,R4           ;RE-SET DRIVE WILL CLEAR DRIVE ERRORS AND READ
                                                                  ;DRIVE STATUS,SET FUNCTION BITS IN COMMAND WORD.
165016  012761  000013  000004            MOV     #13,4(R1)       ;A DRIVE RESET IN A GET STATUS WITH (DA) SET.
                                                                  ;SET DISK ADDRESS (DA) TO 13.
165024  004767  000106                    JSR     PC,RLGO         ;"GO" WILL DO FUNCTION IN R4
                                                                  ;IF NO RETURN IS MADE, THEN A ERROR IN THE
                                                                  ;RLGO ROUTINE.
                                                                  ;REFER TO THE RLGO ROUTINE FOR CAUSE OF ERROR.
                                                                  ;REGISTERS VALUES WILL BE
                                                                  ;CARRY BIT = 1 ERROR
                                                                  ;CARRY BIT = 0 NO ERROR
                                                                  ;R0 = UNIT NUMBER
                                                                  ;R1 = CSR ADDRESS
                                                                  ;R2 = ADDRESS OF ERROR CALL.
                                                                  ;R3 = CSR CONTENTS
                                                                  ;R4 = CSR ADDRESS
165030  000207                            RTS     PC              ;RETURN TO CALLER WITH INDICATORS SET BY RLGO.
```

```
                                                        ;READ HEADER ROUTINE
165032                              RDHEAD:
165032  042704  000016                      BIC     #16,R4          ;CLEAR FUNCTION BITS

165036  052704  000010                      BIS     #10,R4          ;SET COMMAND FUNCTION TO READ A HEADER.


165042  004767  000070                      JSR     PC,RLGO         ;DO COMMAND IN R4 RETURN WITH THE FOLLOWING
165046  103014                              BCC     40$             ;CARRY = 0 NO ERROR, DO BRANCH.
                                                                    ;CARRY = 1 ERROR.
                                                                    ;TEST THAT ERROR IS LEGAL ONE.
165050  032703  040000                      BIT     #40000,R3       ;TEST IF DRIVE ERROR.
165054  001002                              BNE     20$             ;DRIVE ERROR IS BIT 14,BRANCH IF DRIVE ERROR.

                                                        ;CONTROLLER ERROR
165056  010305                              MOV     R3,R5           ;R5 = CONTENTS OF CSR.
165060  000402                              BR      35$             ;GO TO ERROR RETRY  ROUTINE

                                                        ;DRIVE ERROR

165062  004767  000020              20$:    JSR     PC,RDSTAT       ;GET DRIVE STATUS
                                                                    ;THIS ROUTINE WILL LOAD R5 WITH DRIVE STATUS

165066  004767  177714              35$:    JSR     PC,RESDRV       ;THERE IS A ERROR, RESET DRIVE

165072  077221                              SOB     R2,RDHEAD       ;CARRY = 1 ERROR
                                                                    ;SUBTRACT 1 FROM RETRY COUNTER AND IF
                                                                    ;NOT ZERO REDO READ HEADER COMMAND.


                                                        ;*************************************************************
                                                        ;READ HEADER ERROR
                                                        ;COULD NOT READ A HEADER.ERROR RETRY COUNTER IS ZERO.
165074  004767  000104                      JSR     PC,RLERR1       ;A LOAD DEVICE ERROR HAS OCCURRED,JUMP TO
                                                                    ;ERROR ROUTINE. TEST FOR FURTHER PROGRAM
                                                                    ;DIALOGUE OR EXECUTE A "HALT".
                                                                    ;IF "HALT" IS EXECUTED AND THE USER "PROCEEDS"
                                                                    ;THE PROGRAM WILL RERUN THE BOOTSTRAP.
                                                                    ;IF THE "EPROM" IS SELECTED REFER TO
                                                                    ;BOOTSTRAP ERROR ROUTINE FOR MORE INFORMATION
                                                                    ;ON "EPROM" EXECUTION.
                                                                    ;ERROR REGISTER CONTAIN
                                                                    ;R0 = UNIT NUMBER
                                                                    ;R1 = CSR ADDRESS
                                                                    ;R2 = ERROR RETRY COUNTER
                                                                    ;R3 = CONTENTS OF CSR
                                                                    ;R4 = COMMAND WORD
                                                                    ;R5 = DRIVE STATUS
                                                                    ;R5 = CONTENTS OF CSR IF CONTROLLER ERROR.
                                                        ;*************************************************************


                                                                    ;MOVE HEADER DATA TO R5.
165100  016105  000006              40$:    MOV     6(R1),R5        ;R5 CONTAINS HEADER WORD ON PRESENT CYLINDER.
165104  000207                              RTS     PC              ;RETURN TO CALLER WITH INDICATORS SET BY RLGO.
```

```
                                                      ;READ ERROR STATUS
 165106                          RDSTAT:
 165106  010405                          MOV     R4,R5            ;DO NOT DESTROY ORIGINAL COMMAND WORD

 165110  042705  000016                  BIC     #16,R5           ;USE R5 TO TEST FOR ERROR IN DRIVE.
                                                                  ;CLEAR FUNCTION BITS

 165114  052705  000004                  BIS     #4,R5            ;SET FUNCTION RITS TO DO GET STATUS


 165120  012761  000003  000004          MOV     #3,4(R1)         ;SET DISK ADDRESS REGISTER(DA) TO 3 READ STATUS.

 165126  010511                          MOV     R5,(R1)          ;EXECUTE COMMAND IN R5


 165130  016105  000006                  MOV     6(R1),R5         ;MULTI-PURPOSE REGISTER CONTAINS
 165134  000207                          RTS     PC               ;EXIT READ STATUS
```

```

                                                 ;GO ROUTINE
                                                              ;THIS ROUTINE WILL EXECUTE THE COMMAND WORD.
                                                              ;CHECK FOR CONTROLLER READY
                                                              ;CHECK FOR ERROR BIT SET AND SET PROGRAM ERROR FLAG
                                                              ;LOAD CSR CONTENTS INTO R3.
165136                          RLGO:
165136  010411                          MOV     R4,(R1)       ;EXECUTE COMMAND IN R4


165140  005003                  RLGO1:  CLR     R3            ;CALL TO HERE WHEN ANOTHER ROUTINE LOADED CSR
165142  012746  000200                  MOV     #128.,-(SP)   ;CLEAR WATCHDOG TIMER AND SET EXTENDED TIMER .
165146  105711                  10$:    TSTB    (R1)          ;TEST FOR CONTROLLER DONE FLAG.
165150  100407                          BMI     20$           ;BRANCH WHEN SET
165152  005203                          INC     R3            ;FLAG NOT SET BUMP TIMER AND TEST FLAG AGAIN.
165154  001374                          BNE     10$           ;BRANCH AS LONG AS TIME STILL REMAINS
165156  005316                          DEC     (SP)          ;SUBTRACT ONE FROM EXTENDED TIMER AND BRANCH
165160  001372                          BNE     10$           ;BACK AS LONG AS EXTENDED COUNTER DOES NOT GET TO ZERO
                                                  ;************************************************************
                                                              ;TIME OUT ERROR
165162  011103                          MOV     (R1),R3       ;NO CONTROLLER DONE FLAG SET IN TIME ALLOWED.
165164  004767  000014                  JSR     PC,RLERR1     ;TIME OUT ERROR.
                                                              ;NO CONTROLLER READY IN TIME ALLOWED.
                                                              ;ERROR REGISTERS ARE
                                                              ;R0= DRIVE SELECTED
                                                              ;R1= ADDRESS OF CSR
                                                              ;R2= ADDRESS OF ERROR CALL.
                                                              ;R3= CONTENTS OF CSR
                                                              ;R4= COMMAND WORD
                                                  ;************************************************************

165170                          20$:
165170  005726                          TST     (SP)+         ;CORRECT STACK FOR PROPER RETURN.
165172  005711                          TST     (R1)          ;CONTROLLER ERROR FLAG SET
165174  100001                          BPL     NEXT          ;TEST FOR ERROR BIT15 , BRANCH IF NOT SET.
165176  000261                          SEC                   ;ERROR FLAG SET. SET CARRY = 1.
165200  011103                  NEXT:   MOV     (R1),R3       ;LOAD R3 WITH CONTENTS OF CSR
165202  000207                          RTS     PC            ;EXIT ROUTINE
                                                              ;CARRY = 0 NO ERROR
                                                              ;CARRY = 1 ERROR
                                                              ;R0= DRIVE SELECTED
                                                              ;R1= ADDRESS OF CSR
                                                              ;R2= ERROR RETRY COUNTER
                                                              ;R3= CONTENTS OF CSR
                                                              ;R4= COMMAND WORD

165204                          RLERR1:                       ;RLV11 LOAD DEVICE ERROR ROUTINE.

165204  004767  177676          10$:    JSR     PC,RDSTAT     ;PUT DRIVE STATUS REGISTER IN R5.
165210  011602                          MOV     (SP),R2       ;MAKE R2 ADDRESS OF ERROR CALL


165212  000000                          HALT                  ;RLV  DEVICE ERROR.

```

```
                                                                 ;THE ERROR WAS CAUSED BY A RLV LOAD DEVICE.
                                                                 ;STACK POINTS TO ROUTINE THAT CALLED ERROR "HALT".
     165214  000167  177374              JMP      RLREDO         ;PROCEEDING FROM THIS "HALT" WILL RERUN THE
                                                                 ;RLV BOOTSTRAP. REFER TO SPECIFIC ERROR
                                                                 ;FOR CAUSE OF FAILURE.

                                                                 ; R0 = UNIT NUMBER
                                                                 ; R1 = CSR ADDRESS
                                                                 ; R2 = ADDRESS OF ERROR CALL.
                                                                 ; R3 = CONTENTS OF CSR
                                                                 ; R4 = COMMAND WORD.
                                                                 ; R5 = ERROR STATUS REGISTER OF DRIVE



                                                                 .SBTTL END STATEMENT

             000001                 .END
```

```
AROUN2   172010       ERRBIT   171762       PP.BIS=  000001       READZU   172574       RX$S02=  004000
AUTOBA   170512 G     E.EXT =  000100       PP.BI0=  000000       REGOUT   171260       RSABRT=  000006
BADBOT   164362       E.INT =  000010       PP.BI1=  000002       RESDRV   165006       RSCOMP=  000004
BAUDRS=  000032       E.PAR =  000001       PP.BI2=  000004       RESTAR   172004 G     RSDIAG=  000007
BD.003=  000000       FAKOUT   170424       PP.BI3=  000006       RETRY =  000010       RSEND =  000100
BD.006=  000010       FILNAM=  000002       PP.BI4=  000010       RFLAG =  000200       RSGETC=  000012
BD.012=  000020       GETCHR   171530       PP.BI5=  000012       RING     002154       RSGETS=  000010
BD.024=  000030       GETNUM   171632       PP.BI6=  000014       RLBOOT   164600       RSINIT=  000001
BD.048=  000040       GO    =  000001       PP.BI7=  000016       RLERR1   165204       RSNOP =  000000
BD.096=  000050       GOR      172374       PP.C  =  176204       RLGO     165136       RSPOSI=  000005
BD.192=  000060       HGHSEG=  001004       PP.CHI=  000010       RLGO1    165140       RSREAD=  000002
BD.384=  000070       HKBDQ    171160       PP.CLO=  000001       RLINIT   164620       RSSETC=  000013
BEGIN    164376       HKBDS    171162       PP.CWR=  176206       RLO2BT   164356       RSSETS=  000011
BIT0  =  000001       HVBAUD   170576       PP.DRA=  000020       RLREDO   164614       RSWRIT=  000003
BIT1  =  000002       ILOOP    164426       PP.DRB=  000002       RLUNIT   164136       RS$CON=  000020
BIT10 =  002000       INBYTE   170570       PP.MDA=  000040       RMSG     002004       RS$CTL=  000002
BIT11 =  004000       INBYTS   170576       PP.MDB=  000004       RPOINT=  177756 G     RS$DAT=  000001
BIT12 =  010000       INITS    171770       PP.MD2=  000100       RTSEMT=  000052       RS$INT=  000004
BIT13 =  020000       IN.USR=  177764 G     PP.MOD=  000200       RTSFCH=  000056       RS$XOF=  000023
BIT14 =  040000       KBDQ     170714       PRINT    170726       RTSFCT=  000057       R.HALT=  100000
BIT15 =  100000       KBDS     170722       PRI6  =  000300       RTSHGH=  000050       R.NXM =  000200
BIT2  =  000004       LCSET    171266       PRI7  =  000340       RTSISP=  000042       R.PC  =  177766 G
BIT3  =  000010       LEDOFF=  000017       PUTCHR   171542       RTSJSW=  000044       R.STAK=  000001
BIT4  =  000020       LF    =  000012       PUTCLF   171574       RTSRMN=  000054       R.TYPE=  177762 G
BIT5  =  000040       LOAD     172520       PUTLF    171604       RTSSTA=  000040       R.STRT   170036 G
BIT6  =  000100       LOCDSP   171272       PUTSTR   171562       RTSUER=  000053       R6      =%000006
BIT7  =  000200       MODE  =  000221       PWRSUP   170260 G     RTSUSR=  000046       R7      =%000007
BIT8  =  000400       MSGQ     171750       P.BCNT=  000014       RXBOOT   172204       SAVPC =  177752 G
BIT9  =  001000       MSGS     171751       P.LEN =  177774       RXCS  =  177170       SAVPS =  177754 G
BOOTS    164000 G     NEXNUM   171620       P.OPCD=  000010       RXDB  =  177172       SECBOT   165004
BRKNO0   170010 G     NEXT     165200       P.SHUN=  000040       RXESCR=  000001       SECBT    164574
BUFFER   001000       NOCT     171662       P.STS =  000012       RXESDD=  000100       SEEK     164654
B.CNTL=  177770 G     NOUNIT   164220       P.UNIT=  000004       RXESDE=  000020       SEGALO=  001000
CEN      172430       NO.LOW=  100000       QODT     170706       RXESDN=  000040       SLOOP    164450
CHIN     173656       NXTSEG=  001002       RAMBOT=  177600       RXESDR=  000200       SPACE =  000040
CHK240   172232       OCTSTR   171676       RAMTOP=  177776       RXESID=  000004       SRET     171656
CHRET    173672       OCTST0   171672       RBUFS1=  177562       RXESUN=  000400       STANDB   172370
CH2IN    173654       ODT      170622 G     RBUFS2=  176542       RXGO     173360       START    172000 G
CH2OUT   173630       ODTFLG=  177750 G     RB.BRK=  004000       RXSEMP=  000003       STARTS   172556
CH8OUT   173624       ODTLOC=  177746 G     RB.ERR=  100000       RX$FIL=  000001       STEP  =  100000
CKMSK    164156       ODTSTK=  177744 G     RB.FRM=  020000       RX$REC=  000017       STRBLK=  001010
CLOOP    164452       ODTWHY=  177774 G     RB.OVR=  040000       RX$RED=  000007       STTUBD   164006 G
CMSK     164120       ONENUM   171616       RCMD     171164       RX$RST=  000013       ST173    173000 G
CR    =  000015       OP.ONL=  000011       RCMD1    171252       RX$STD=  000011       SWCMD    171232
DEVBIT=  000200       OP.RD =  000041       RCMD2    002070       RX$WDD=  000015       SWITCH=  177524
DEVNUM=  000001       OUT      170510       RCSRS1=  177560       RX$WRT=  000005       S$CART=  177767
DIAGNO   172010       OWN   =  100000       RCSRS2=  176540       RX$SDE=  000400       S$DCHK=  177757
DIRBUF=  001000       O.CNTL=  177772 G     RC.ACT=  004000       RX$SDN=  000040       S$MOTR=  177737
DONE     171614       PATERN   171770       RC.DUN=  000200       RX$SER=  100000       S$NORM=  000000
DREAD    173016       PBR0  =  000010       RC.IEN=  000100       RX$SFN=  000016       S$OPCD=  177720
DXDYBT   164352       PBR1  =  000020       RDBOOT   164372       RX$SGO=  000001       S$PART=  177776
D.FLEN=  000010       PBR2  =  000040       RDHEAD   165032       RX$SIE=  000100       S$RECN=  177711
EMPTYS=  001000       PCMD     171064       RDLIN    164062       RX$SIN=  040000       S$RETR=  000001
ENDSGS=  004000       PERMFS=  002000       RDSTAT   165106       RX$STR=  000200       S$SEEK=  177740
ENTSIZ=  000016       PP.A  =  176200       RDUNIT   164100       RX$SUN=  000020       S$UNIT=  177770
EPB      172500       PP.B  =  176202       READ     164722       RX$SXA=  030000       S$WPRT=  177765
ERR   =  100000       PP.BIC=  000000       READU    172576       RX$SXX=  003000       S1    =  004000
```

```
KXT11-A5 1K FIRMWARE    MACRO M1200  04-NOV-83 13:50  PAGE 72-3
SYMBOL TABLE

$2     = 010000          TOSCSR= 176544          UDAIP = 176150          V102    170464          XC.PBE= 000002
$3     = 020000          TRAP4 = 177776 G        UDAM    164372          XBUFS1= 177566          XC.RDY= 000200
$4     = 040000          TREAD   173406          UDASA = 176152          XBUFS2= 176546          XDONE   164526
TABLE    164412          TST24   164770          UNIFOR  164224          XCSRS1= 177564          XTRBYT= 001006
TENTAS= 000400          TUBAUD= 000072          UNINUM  164174          XCSRS2= 176544          $STACK= 177644 G
TISBFR= 176542          TUBOOT  172270          USERSP= 177760 G        XC.BRK= 000001          $$$BRK  170000 G
TISCSR= 176540          TUSBBT  164346          VECSET  170440 G        XC.IEN= 000100          $$$LTC  170006 G
TOSBFR= 176546          T.BIT = 000020          VECT    164244          XC.MNT= 000004

. ABS.  174000     000
        000000     001
ERRORS DETECTED:  0

VIRTUAL MEMORY USED:  8952 WORDS  ( 35 PAGES)
DYNAMIC MEMORY:  19748 WORDS  ( 75 PAGES)
ELAPSED TIME:  00:01:14
FALCON,FALCON/CRF/-SP=FALCON.MAC;45
```

```
FALCON      CREATED BY  MACRO  ON 4-NOV-83 AT 13:50       PAGE 1
SYMBOL CROSS REFERENCE                                    CREF   V01

SYMBOL   VALUE           REFERENCES

AROUN2     172010          45-1565    #47-1576
AUTOBA     170512  G       28-881     #29-955
BADBOT     164362          63-2457    #63-2492
BAUDRS   = 000032         #9-216       29-956
BD.003   = 000000         #7-128
BD.006   = 000010         #7-129
BD.012   = 000020         #7-130
BD.024   = 000030         #7-131       9-216
BD.048   = 000040         #7-132
BD.096   = 000050         #7-133
BD.192   = 000060         #7-134
BD.384   = 000070         #7-135       9-220
BEGIN      164376         #64-2514
BIT0     = 000001         #5-44        7-150      8-182      8-200      10-242     10-250     36-1250    #65-2627
BIT1     = 000002         #5-45        7-143      8-180      8-191      8-193      8-195      8-197      #65-2626
BIT10    = 002000         #5-54       #65-2617
BIT11    = 004000         #5-55        6-83       6-104     #65-2616
BIT12    = 010000         #5-56       #65-2615
BIT13    = 020000         #5-57        6-102     #65-2614
BIT14    = 040000         #5-58        6-99      #65-2613
BIT15    = 100000         #5-59        6-97       10-240     10-246    #65-2612
BIT2     = 000004         #5-46        7-137      8-178      8-191      8-192      8-195      8-196      #65-2625
BIT3     = 000010         #5-47        7-122      8-176      8-191      8-192      8-193      8-194      #65-2624
BIT4     = 000020         #5-48        7-123      8-174      9-236     #65-2623
BIT5     = 000040         #5-49        7-124      8-171      39-1363   #65-2622
BIT6     = 000100         #5-50        6-90       7-115      8-170      39-1361   #65-2621
BIT7     = 000200         #5-51        6-86       7-110      8-168      9-234      10-241     10-248     #65-2620
BIT8     = 000400         #5-52       #65-2619
BIT9     = 001000         #5-53       #65-2618
BOOTS      164000  G       34-1149    #63-2383
BRKNO0     170010  G       15-379     #15-387     22-674
BUFFER     001000         #14-346
B.CNTL   = 177770  G      #13-323     *27-866     54-2028   *63-2444    63-2445
CEN        172430         #54-1981     54-2005
CHIN       173656         #62-2346
CHK240     172232         #52-1903     53-1950
CHRET      173672          62-2335    #62-2350
CH2IN      173654          61-2294     61-2306   #62-2345
CH2OUT     173630          60-2247    #62-2327
CH8OUT     173624          53-1931    #62-2324
CKMSK      164156          63-2424    #63-2428
CLOOP      164452         #64-2552     64-2554
CMSK       164120          63-2413    #63-2417
CR       = 000015         #5-64        38-1324    39-1373    41-1430    42-1458    44-1523   #65-2596
DEVBIT   = 000200         #10-248      63-2397
DEVNUM   = 000001         #10-250      63-2475
DIAGNO     172010          34-1153    #47-1578
DIRBUF   = 001000         #49-1743     49-1799    49-1800    49-1801    49-1802    49-1803    54-1974
DONE       171614          41-1423    #41-1437
DREAD      173016          51-1885     54-2029   #57-2068
DXDYBT     164352          63-2479    #63-2488
D.FLEN   = 000010         #49-1806     54-2001
```

```
FALCON      CREATED BY  MACRO  ON 4-NOV-83 AT 13:50       PAGE 2

SYMBOL CROSS REFERENCE                                     CREF   V01

SYMBOL  VALUE            REFERENCES

EMPTYS  = 001000         #49-1808
ENDSGS  = 004000         #49-1810    54-1984
ENTSIZ  = 000016         #49-1805
EPB       172500          54-1986    54-1993     54-1996    #54-2000
ERR     = 100000         #50-1838
ERRBIT    171762         #45-1545    47-1595
E.EXT   = 000100         #10-254     45-1543
E.INT   = 000010         #10-255     45-1544
E.PAR   = 000001         #10-256     47-1590
FAKOUT    170424          28-880    #28-883
FILNAM  = 000002         #49-1741    54-1991
GETCHR    171530          34-1146    37-1288     38-1323    #40-1396     40-1398     42-1457     63-2394     63-2411     63-2422
                          63-2433
GETNUM    171632          34-1164    38-1331    #42-1460
GO      = 000001         #50-1843    64-2530
GOR       172374         #54-1970    54-1988
HGHSEG  = 001004         #49-1801
HKBDQ     171160         #36-1256    38-1319     38-1329     38-1335     38-1337     39-1370     39-1372
HKBDS     171162         #36-1257    38-1325     39-1367
HVBAUD    170576          30-970    #30-991
ILOOP     164426          64-2519   #64-2540     64-2543     64-2546
INBYTE    170570          30-968    #30-980      30-992
INBYTS    170576          30-971    #30-987
INITS     171770         #45-1554    47-1598
IN.USR  = 177764  G      #13-325     15-378     *15-392     *20-612      21-645     *22-668     *23-720     *28-884     *36-1252
                         *63-2389
KBDQ      170714         #33-1118    34-1160     34-1162     34-1165     34-1173     36-1237     36-1256     37-1274     37-1278
                          37-1280    37-1290
KBDS      170722          33-1114   #33-1121     36-1257     47-1629
LCSET     171266          34-1171   #38-1315
LEDOFF  = 000017         #9-212      27-844      47-1580
LF      = 000012         #5-63       38-1334     39-1365     39-1369     41-1435     44-1523    #65-2595
LOAD      172520          54-1998   #54-2007
LOCDSP    171272          37-1297   #38-1316     39-1380
MODE    = 000221         #9-207      25-770      47-1579
MSGQ      171750          33-1119   #44-1522
MSGS      171751          33-1122   #44-1523
NEXNUM    171620         #42-1456    42-1468
NEXT      165200          72-3046   #72-3048
NOCT      171662          42-1463   #42-1473
NOUNIT    164220          63-2419    63-2430    #63-2440
NO.LOW  = 100000         #10-246     27-866
NXTSEG  = 001005         #49-1800    54-1987
OCTSTR    171676          38-1320    39-1377    #43-1491
OCTST0    171672          33-1110   #43-1490     47-1628
ODT       170622  G       15-393     23-726      28-886     #33-1081
ODTFLG  = 177750  G      #13-335    *33-1127    *37-1272     38-1336     39-1371
ODTLOC  = 177746  G      #13-336    *37-1296    *38-1315     38-1343     39-1364    *39-1375     39-1376
ODTSTK  = 177744  G      #13-338     13-339      33-1091
ODTWHY  = 177774  G      #13-316    *33-1086    *36-1235
ONENUM    171616          37-1273   #42-1454
OP.ONL  = 000011         #50-1852    64-2534
```

```
SYMBOL CROSS REFERENCE                                         CREF   V01

SYMBOL   VALUE            REFERENCES

OP.RD    = 000041         #50-1853    64-2536
OUT        170510          28-908    #28-911
OWN      = 100000         #50-1848
O.CNTL   = 177772   G     #13-319     38-1346     39-1357
PATERN     171770         #45-1560    47-1603
PBR0     = 000010         #7-122      7-129       7-131       7-133       7-135
PBR1     = 000020         #7-123      7-130       7-131       7-134       7-135
PBR2     = 000040         #7-124      7-132       7-133       7-134       7-135
PCMD       171064          34-1156   #36-1222
PERMFS   = 002000         #49-1809    54-1982
PP.A     = 176200         #8-159      47-1588
PP.B     = 176202         #8-160     *47-1587
PP.BIC   = 000000         #8-201
PP.BIS   = 000001         #8-200      9-212
PP.BI0   = 000000         #8-198
PP.BI1   = 000002         #8-197
PP.BI2   = 000004         #8-196
PP.BI3   = 000006         #8-195
PP.BI4   = 000010         #8-194
PP.BI5   = 000012         #8-193
PP.BI6   = 000014         #8-192
PP.BI7   = 000016         #8-191      9-212
PP.C     = 176204         #8-161
PP.CHI   = 000010         #8-176
PP.CLO   = 000001         #8-182      9-207
PP.CWR   = 176206         #8-158     *25-770     *27-844     *47-1579    *47-1580
PP.DRA   = 000020         #8-174      9-207
PP.DRB   = 000002         #8-180
PP.MDA   = 000040         #8-171
PP.MDB   = 000004         #8-178
PP.MD2   = 000100         #8-170
PP.MOD   = 000200         #8-168      9-207
PRINT      170726          33-1120   #33-1124
PRI6     = 000300         #9-229      28-879      33-1125     63-2458
PRI7     = 000340         #9-230      28-906      28-910      36-1248     55-2037
PUTCHR     171542          38-1322    39-1374     39-1379    #40-1400     40-1402     41-1424     41-1431     41-1436     43-1498
PUTCLF     171574         #41-1429    43-1490
PUTLF      171604         #41-1435
PUTSTR     171562          33-1126   #41-1421     41-1425
PWRSUP     170260   G     #25-762     46-1571     55-2042
P.BCNT   = 000014         #50-1858   *64-2559
P.LEN    = 177774         #50-1860   *64-2556
P.OPCD   = 000010         #50-1857   *64-2558
P.SHUN   = 000040         #50-1856
P.STS    = 000012         #50-1859   *64-2561     64-2570
P.UNIT   = 000004         #50-1855   *64-2560
QODT       170706          33-1107   #33-1111
RAMBOT   = 177600         #9-224      26-810
RAMTOP   = 177776         #9-225      26-815
RBUFS1   = 177562         #6-73       25-785      30-963      30-967      33-1082     40-1399
RBUFS2   = 176542         #6-77       49-1750
RB.BRK   = 004000         #6-104
```

```
FALCON      CREATED BY  MACRO  ON 4-NOV-83 AT 13:50        PAGE 4
SYMBOL CROSS REFERENCE                                      CREF   V01

SYMBOL  VALUE            REFERENCES

RB.ERR  = 100000         #6-97
RB.FRM  = 020000         #6-102
RB.OVR  = 040000         #6-99
RCMD      171164          34-1158   #37-1271
RCMD1     171252          37-1281   #37-1294
RCMD2     002070         #14-354    *64-2556   *64-2558   *64-2559   *64-2560   *64-2561    64-2564
RCSRS1  = 177560         #6-72       25-786     30-965     40-1397
RCSRS2  = 176540         #6-76       47-1596    49-1749
RC.ACT  = 004000         #6-83
RC.DUN  = 000200         #6-86       25-786
RC.IEN  = 000100         #6-90       25-786
RDBOOT    164372          63-2488   #64-2508    64-2573
RDHEAD    165032          69-2777    69-2807   #70-2938    70-2962
RDLIN     164062          63-2404   #63-2406
RDSTAT    165106          70-2957   #71-2994    72-3061
RDUNIT    164100          63-2405   #63-2411
READ      164722          69-2805   #69-2842
READU     172576          54-1975   #54-2027
READZU    172574          53-1949    54-2010   #54-2025
REGOUT    171260          37-1283    37-1292   #37-1296
RESDRV    165006          69-2758   #69-2915    70-2960
RESTAR    172004  G       22-684    #46-1572
RETRY   = 000010         #49-1735
RFLAG   = 000200         #9-234      37-1272
RING      002154         #14-358     64-2526    64-2553    64-2568
RLBOOT    164600          63-2490   #69-2740
RLERR1    165204          69-2765    69-2814    69-2861    69-2893    70-2969    72-3033   #72-3059
RLGO      165136          69-2804    69-2855    69-2922    70-2944   #72-3018
RLGO1     165140         #72-3022
RLINIT    164620         #69-2749
RLO2BT    164356          63-2481   #63-2490
RLREDO    164614         #69-2744    72-3070
RLUNIT    164136          63-2408   #63-2422
RMSG      002004         #14-350     64-2562    64-2570
RPOINT  = 177756  G      #13-331    *33-1102    37-1295
RTSEMT  = 000052         #49-1819
RTSFCH  = 000056         #49-1822
RTSFCT  = 000057         #49-1823
RTSHGH  = 000050         #49-1818
RTSISP  = 000042         #49-1815    54-2021
RTSJSW  = 000044         #49-1816
RTSRMN  = 000054         #49-1821
RTSSTA  = 000040         #49-1814    54-2013
RTSUER  = 000053         #49-1820
RTSUSR  = 000046         #49-1817
RXBOOT    172204         #51-1878    63-2485
RXCS    = 177170         #49-1694    49-1695    51-1879    51-1880    58-2167   *59-2212
RXDB    = 177172         #49-1695    57-2074
RXESCR  = 000001         #49-1731
RXESDD  = 000100         #49-1727
RXESDE  = 000020         #49-1729
RXESDN  = 000040         #49-1728    57-2084
```

```
SYMBOL CROSS REFERENCE                                       CREF  V01

SYMBOL   VALUE            REFERENCES

RXESDR  = 000200          #49-1726
RXESID  = 000004          #49-1730
RXESUN  = 000400          #49-1725
RXGO      173360           57-2079   57-2124   58-2165   #59-2210
RXSEMP  = 000003          #49-1715   58-2166
RXSFIL  = 000001          #49-1714
RXSREC  = 000017          #49-1721
RXSRED  = 000007          #49-1717   57-2125
RXSRST  = 000013          #49-1719   57-2080
RXSSTD  = 000011          #49-1718
RXSWDD  = 000015          #49-1720
RXSWRT  = 000005          #49-1716
RXSSDE  = 000400          #49-1704   57-2091
RXSSDN  = 000040          #49-1707   59-2215
RXSSER  = 100000          #49-1699
RXSSFN  = 000016          #49-1709
RXSSGO  = 000001          #49-1710   49-1714   49-1715   49-1716   49-1717   49-1718   49-1719   49-1720   49-1721
RXSSIE  = 000100          #49-1706
RXSSIN  = 040000          #49-1700
RXSSTR  = 000200          #49-1705   59-2215
RXSSUN  = 000020          #49-1708   57-2078
RXSSXA  = 030000          #49-1701
RXSSXX  = 003000          #49-1703
RXSS02  = 004000          #49-1702   58-2167
RSABRT  = 000006          #49-1773
RSCOMP  = 000004          #49-1771
RSDIAG  = 000007          #49-1774
RSEND   = 000100          #49-1779   61-2285
RSGETC  = 000012          #49-1777
RSGETS  = 000010          #49-1775
RSINIT  = 000001          #49-1768
RSNOP   = 000000          #49-1767
RSPOSI  = 000005          #49-1772
RSREAD  = 000002          #49-1769   60-2248
RSSETC  = 000013          #49-1778
RSSETS  = 000011          #49-1776
RSWRIT  = 000003          #49-1770
RSSCON  = 000020          #49-1762   53-1941
RSSCTL  = 000002          #49-1760   60-2246
RSSDAT  = 000001          #49-1759   61-2267
RSSINT  = 000004          #49-1761   53-1936   53-1936
RSSXOF  = 000023          #49-1763
R.HALT  = 100000          #10-240    15-390    22-703
R.NXM   = 000200          #10-241    20-610    23-725    36-1235
R.PC    = 177766   G      #13-324   *22-692   *22-696    22-698
R.STAK  = 000001          #10-242    21-625    36-1235
R.TYPE  = 177762   G      #13-327   *15-390    20-608   *20-610   *21-625   *21-631   *22-703   *23-725    33-1086
                           33-1106   33-1112  *33-1124
R.STRT    170036   G      #20-587    46-1573
R6      =%000006          #65-2605
R7      =%000007          #65-2606
SAVPC   = 177752   G      #13-334   *15-388   *22-694   *28-880    33-1109  *36-1211   36-1254   37-1282
```

```
SYMBOL CROSS REFERENCE                                        CREF   V01

SYMBOL  VALUE             REFERENCES

SAVPS   = 177754  G       #13-333    *15-389    *22-693    *28-879    *36-1216    36-1253    37-1291    38-1343
SECBOT    165004            69-2888  #69-2910
SECBT     164574            64-2579  #64-2582
SEEK      164654          #69-2791
SEGALO  = 001000          #49-1799
SLOOP     164450          #64-2551    64-2576
SPACE   = 000040          #5-65       38-1321
SRET      171656            42-1459  #42-1470
STANDB    172370            52-1907  #54-1968
START     172000  G       #46-1570
STARTS    172556            54-2015  #54-2018
STEP    = 100000          #50-1844    64-2525
STRBLK  = 001010          #49-1803    54-1978
STTUBD    164006  G       #63-2388
ST173     173000  G       #55-2036
SWCMD     171232            37-1276  #37-1287
SWITCH  = 177524          #66-2632
SSCART  = 177767          #49-1787
SSDCHK  = 177757          #49-1789
SSMOTR  = 177737          #49-1791
SSNORM  = 000000          #49-1783
SSOPCD  = 177720          #49-1792
SSPART  = 177776          #49-1785
SSRECN  = 177711          #49-1793
SSRETR  = 000001          #49-1784
SSSEEK  = 177740          #49-1790
SSUNIT  = 177770          #49-1786
SSWPRT  = 177765          #49-1788
S1      = 004000          #50-1842    64-2517
S2      = 010000          #50-1841
S3      = 020000          #50-1840
S4      = 040000          #50-1839
TABLE     164412            64-2518  #64-2525
TENTAS  = 000400          #49-1807
TISBFR  = 176542          #49-1750    62-2349
TISCSR  = 176540          #49-1749    53-1927    53-1939    62-2347
TOSBFR  = 176546          #49-1752   *62-2334
TOSCSR  = 176544          #49-1751    53-1928    62-2332   *63-2384
TRAP4   = 177776  G       #13-314     23-718    *54-2022   *63-2469
TREAD     173406            54-2030  #60-2244
TST24     164770            69-2856  #69-2887
TUBAUD  = 000072          #9-220      63-2384
TUBOOT    172270          #53-1926    63-2486
TUSBBT    164346            63-2484  #63-2486
T.BIT   = 000020          #9-236      38-1348
UDAIP   = 176150          #50-1833    50-1834    64-2510
UDAM      164372          #64-2510    64-2541
UDASA   = 176152          #50-1834
UNIFOR    164224            63-2435    63-2437    63-2439   #63-2443
UNINUM    164174            63-2396    63-2399    63-2401   #63-2433
USERSP  = 177760  G       #13-329    *33-1090    33-1095
VECSET    170440  G         27-863   #28-902     52-1908
```

```
FALCON      CREATED BY  MACRO  ON 4-NOV-83 AT 13:50      PAGE 7
SYMBOL CROSS REFERENCE                                   CREF   V01

SYMBOL   VALUE          REFERENCES

VECT       164244        63-2415    63-2420    63-2426    63-2431    #63-2457
V102       170464        28-904    #28-907
XBUFS1   = 177566       #6-75      *40-1403
XBUFS2   = 176546       #6-79       49-1752
XCSRS1   = 177564       #6-74      *25-782     25-789    *29-956    *30-997     40-1401
XCSWS2   = 176544       #6-78       49-1751
XC.BRK   = 000001       #7-150      53-1934
XC.IEN   = 000100       #7-115
XC.MNT   = 000004       #7-137      45-1552
XC.PBE   = 000002       #7-143      9-216      9-220      45-1551     45-1552
XC.RDY   = 000200       #7-110      25-789
XDONE      164526       #64-2568    64-2569
XTRBYT   = 001006       #49-1802    54-2004
$STACK   = 177644  G    #13-339     25-763     28-885     63-2468     63-2493
$$SBRK     170000  G    #15-377     28-905
$$SLTC     170006  G    #15-381     28-909
```

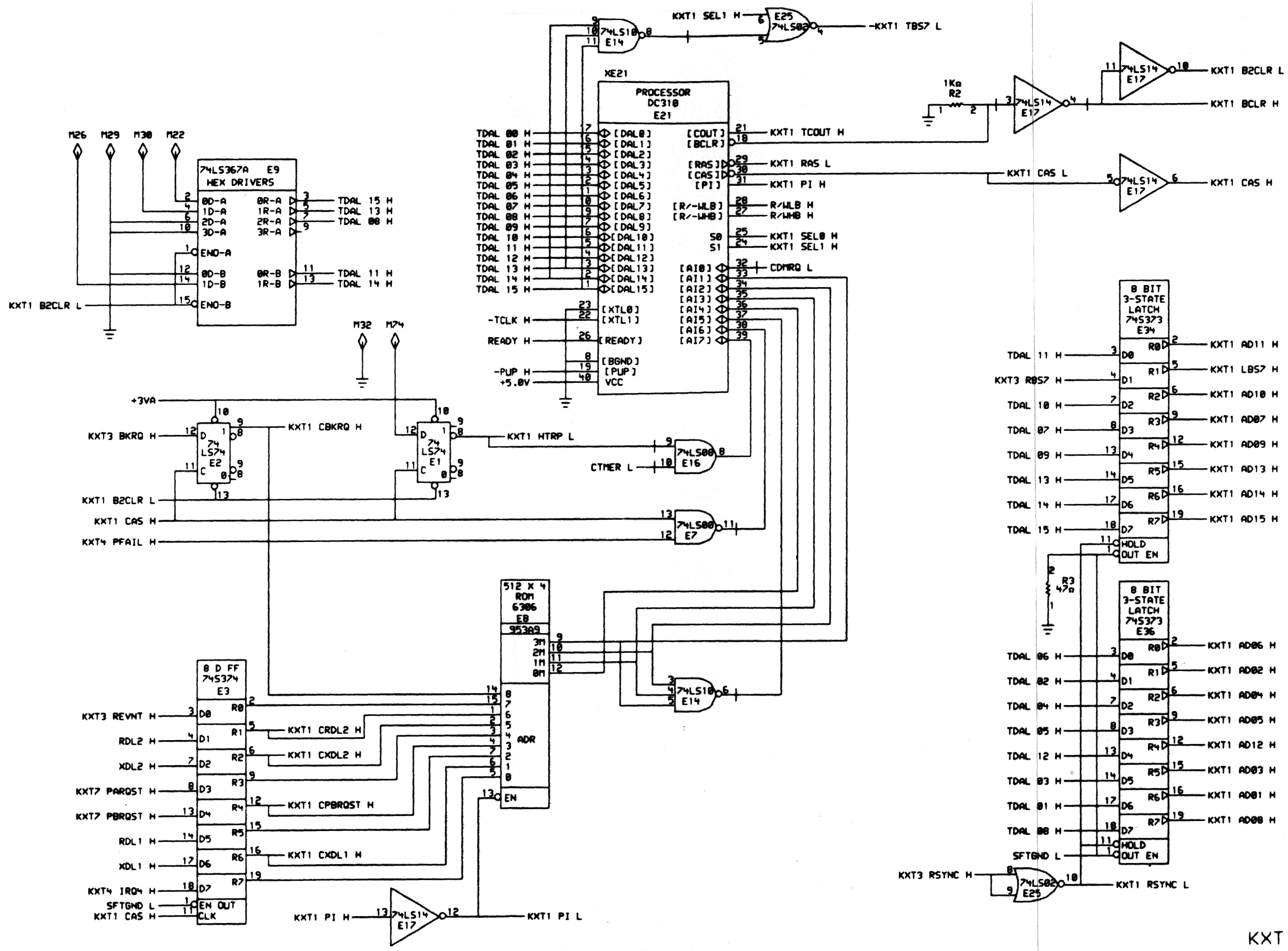
```
MACRO CROSS REFERENCE                                           CREF   V01

MACRO NAME        REFERENCES

ABORT             #11-272    52-1909    53-1943    53-1951    54-1977    54-1989    54-2012    54-2016    57-2083    57-2161
                   61-2287   61-2297    63-2409    63-2441    63-2448    63-2494    64-2580
DELAY             #12-302    63-2467
```

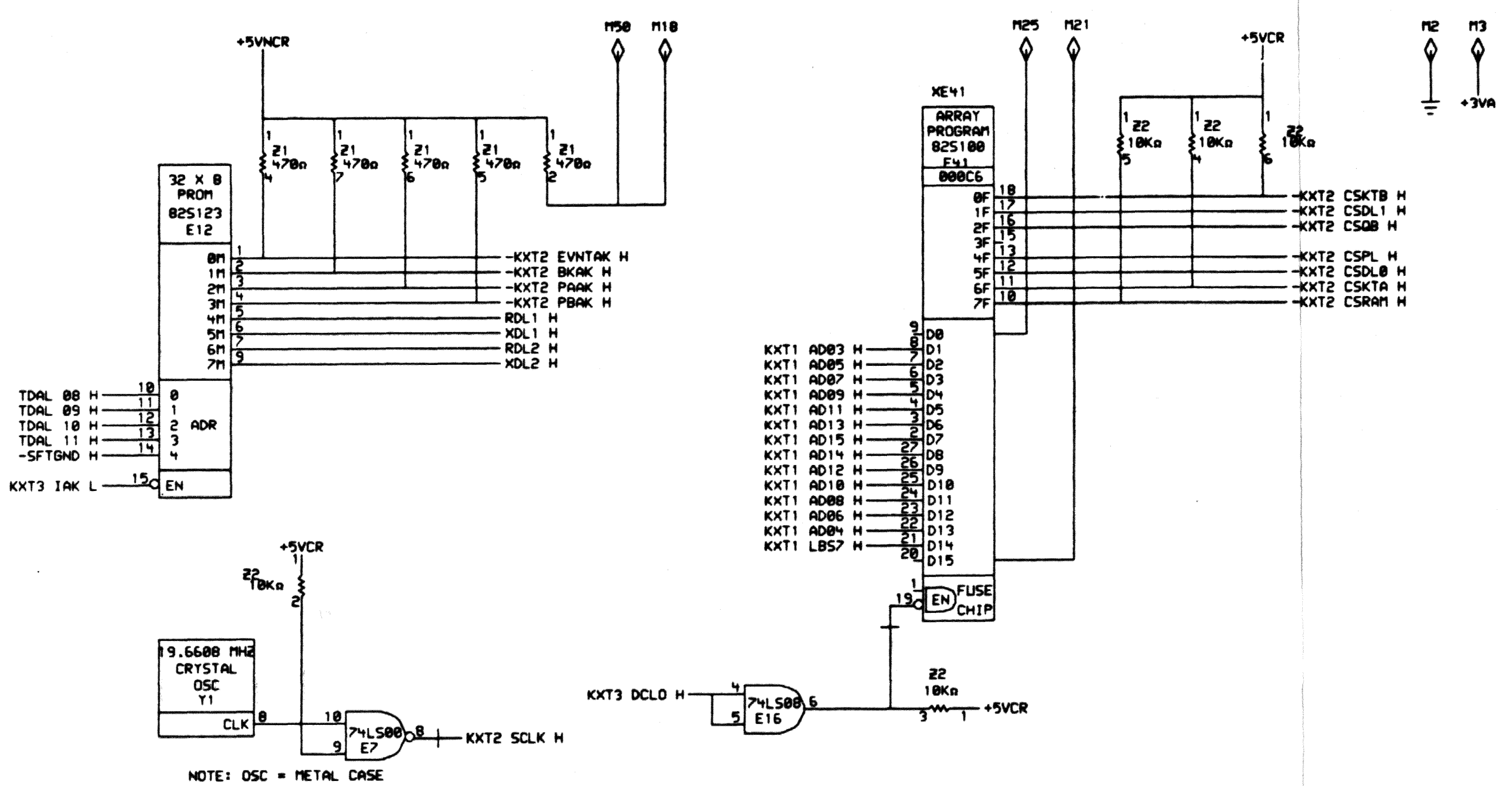
APPENDIX F
SBC-11/21 PLUS SCHEMATIC

Appendix F provides the user with the electrical schematics for the SBC-11/21 PLUS module.



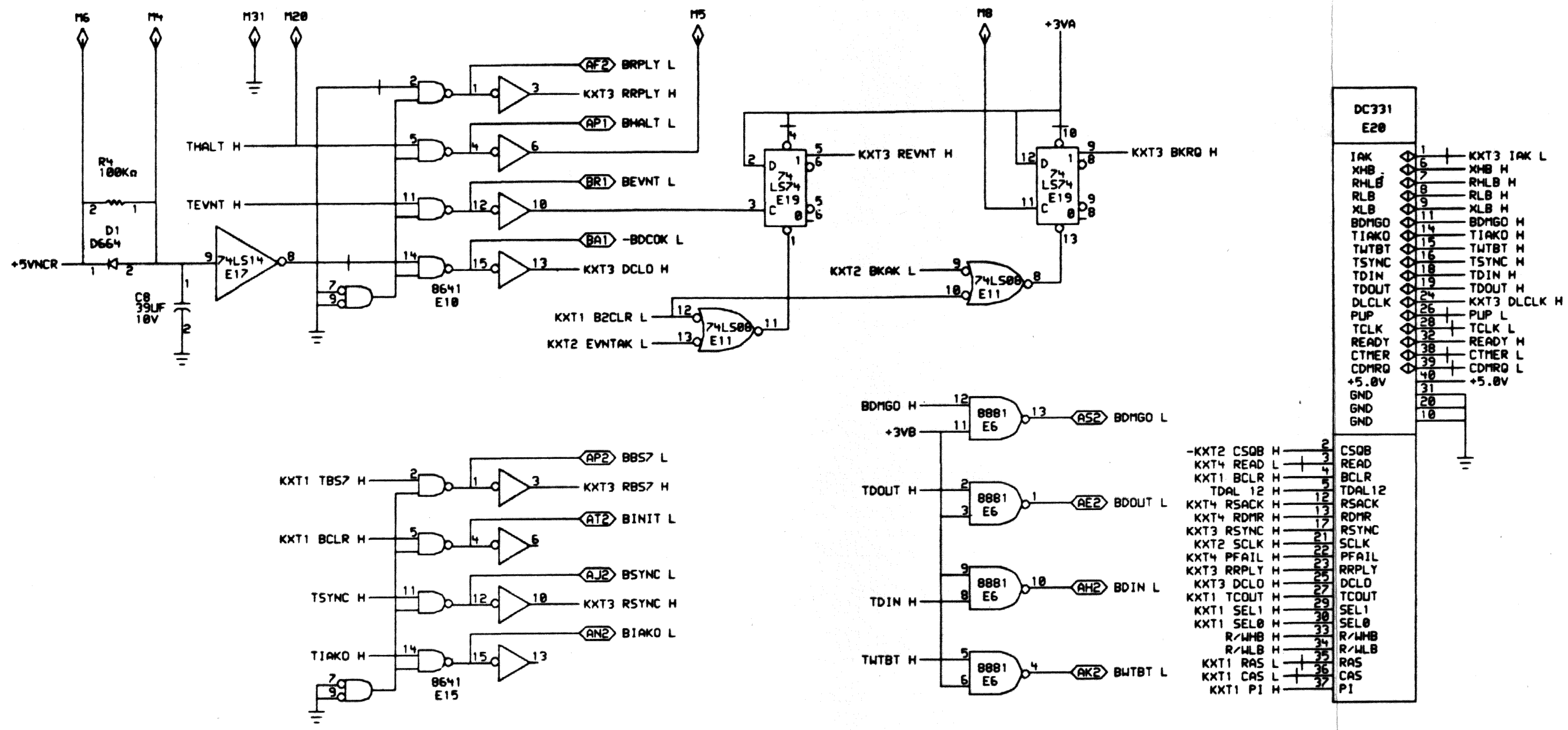
KXT1 CPU

SHEET 1 OF 9

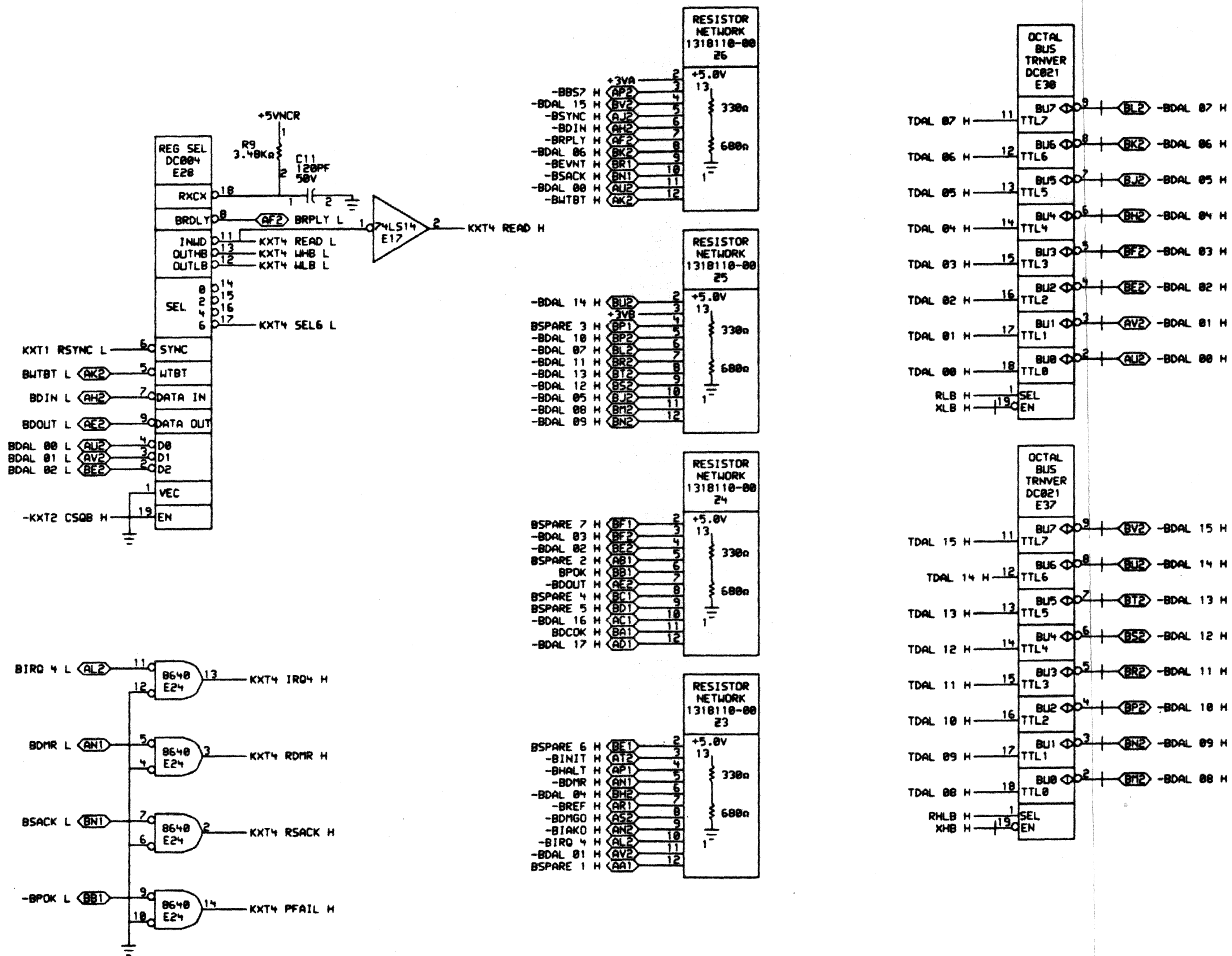


KXT2 DECODE

SHEET 2 OF 9

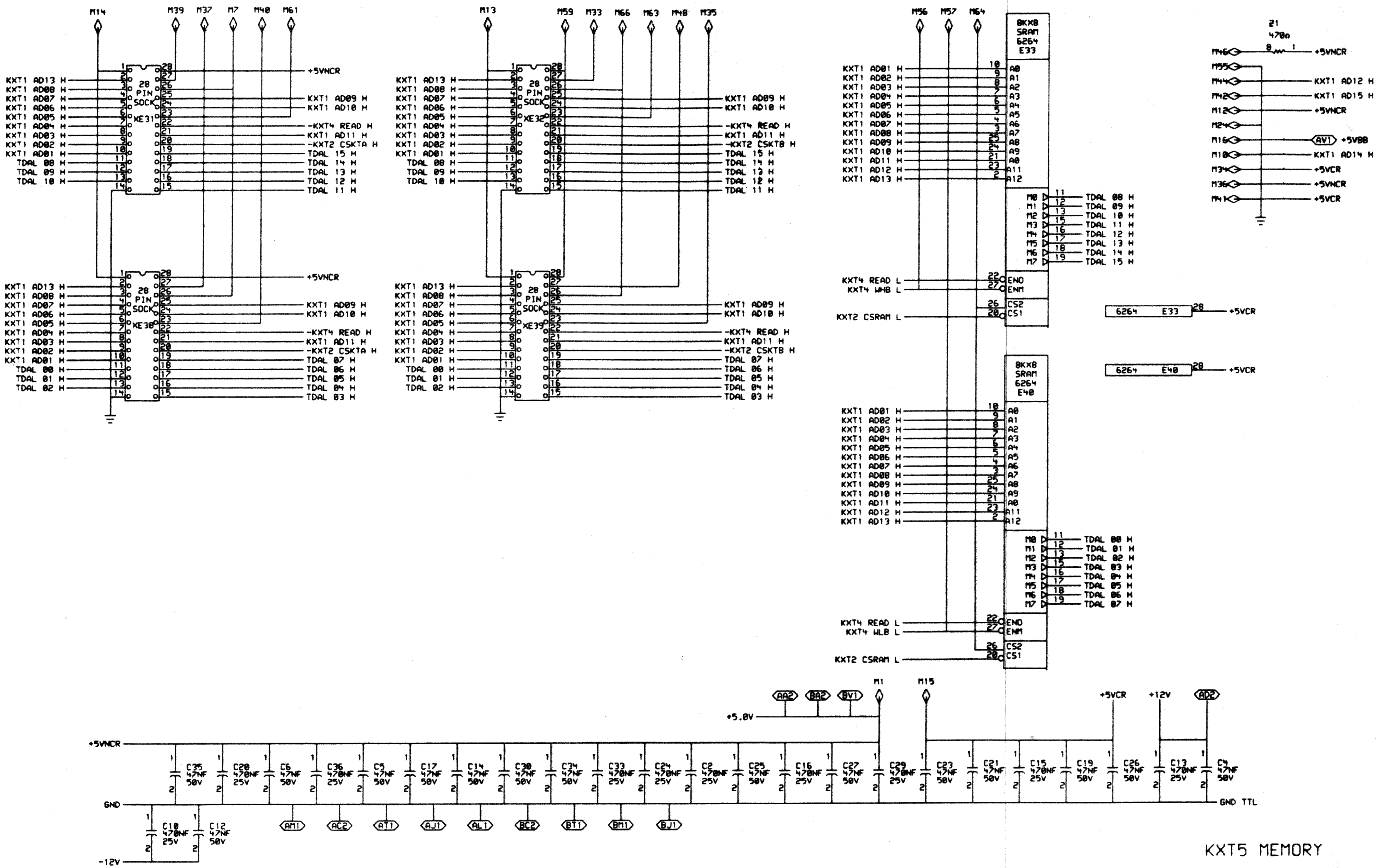


KXT3 BUS INTERFACE



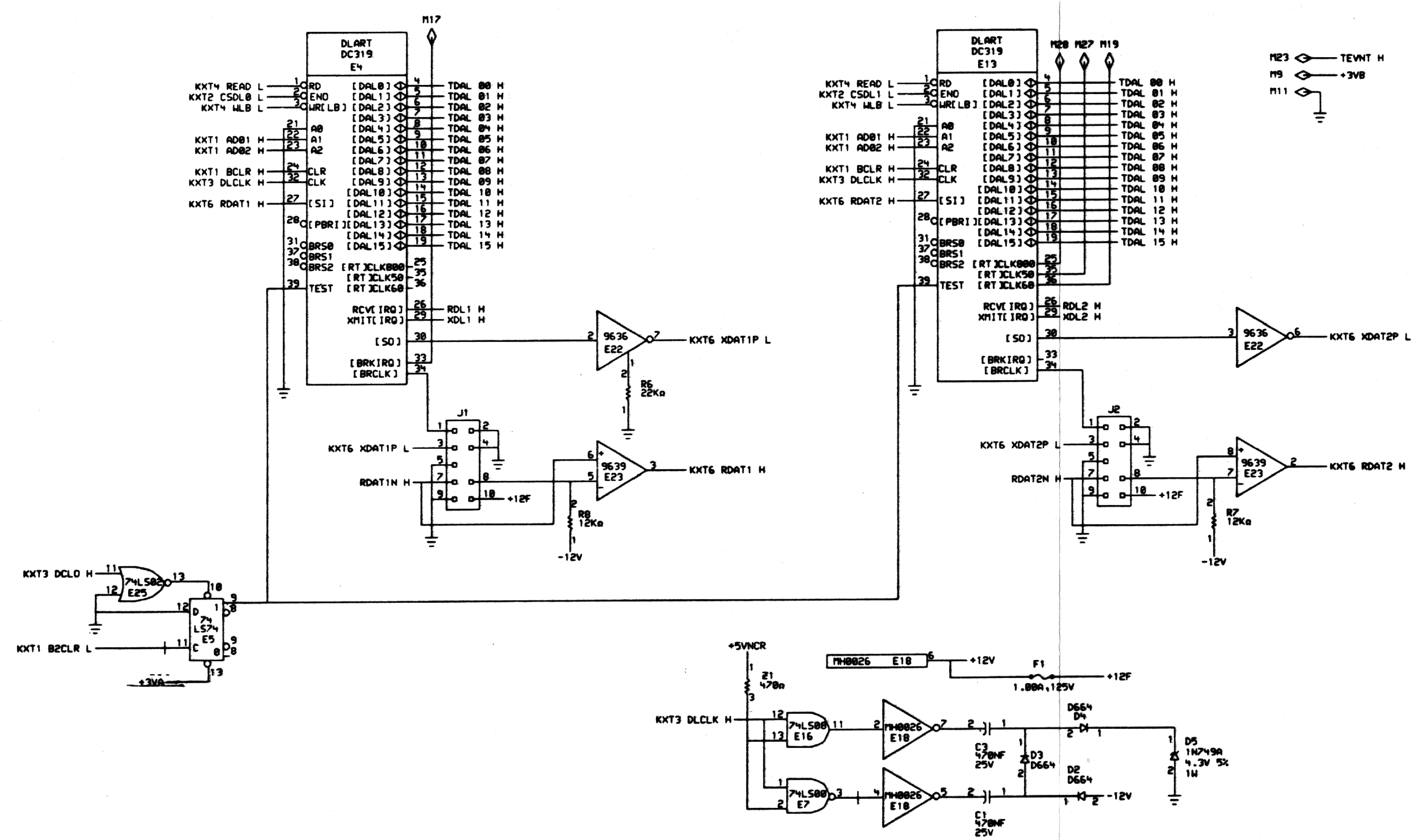
KXT4 BUS I/O

SHEET 4 OF 9



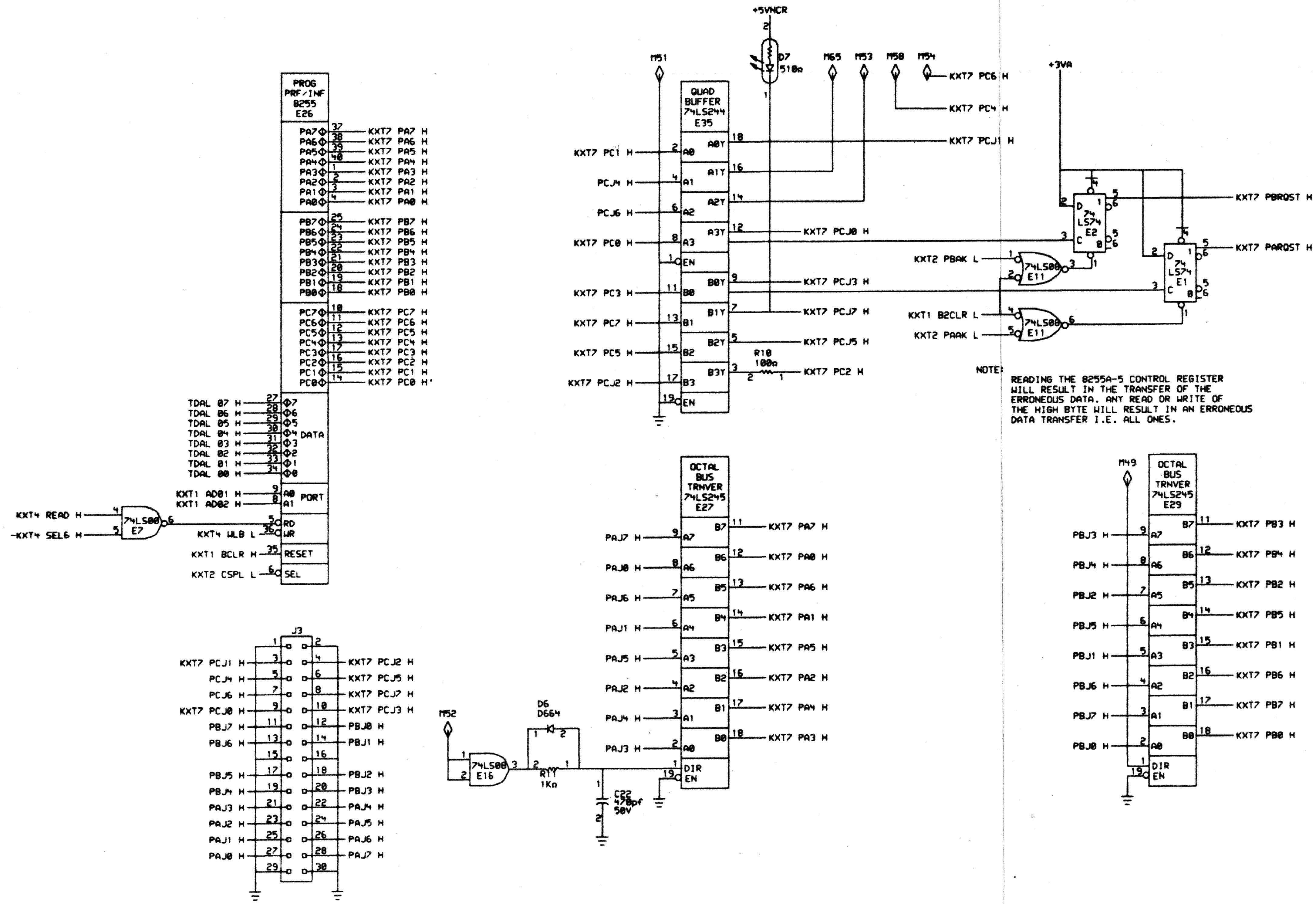
KXT5 MEMORY

SHEET 5 OF 9



KXT6 SERIAL I/O

SHEET 6 OF 9



NOTE: CONNECT TO PC6 IN MODE 2.

KXT7 PARALLEL I/O

SHEET 7 OF 9

Vertical location (A-D)   Direction of line (Left, Right, Up, Down)
or electrical (Input, Output, Both)
KEY: SS-VH,D   or backplane pin (Pin)
Schematic Sheet   Horizontal location (1-8)

Signal	References
+12F	6-B3,L 6-B3,L 6-B6,L
+12V	5-A2,D 6-B3,L
+3VA	1-C7,R 2-D3,D 3-D4,D 4-D5,R 6-B7,R 7-D3,D
+3VB	3-C4,R 4-C5,R 6-D2,L
+5.0V	1-C5,R 3-C2,L 5-A4,R
+5VBB	5-D1,L <AV1>
+5VCR	2-B4,L 2-C6,D 2-D3,D 5-A2,D 5-C2,L 5-C2,L 5-D1,L 5-D1,L
+5VNCR	2-D7,D 3-C8,R 4-D6,D 5-A8,R 5-C7,L 5-D1,L 5-D1,L 5-D1,L 5-D7,L 6-B5,D 7-D4,D
-12V	5-A8,R 6-A3,L 6-B2,D 6-B5,D
BBS7 L	3-C6,L <AP2> 4-D5,R <AP2>
BDAL 00 L	4-C3,L <AU2> 4-C7,R <AU2> 4-D5,R <AU2>
BDAL 01 L	4-A5,R <AV2> 4-C3,L <AV2> 4-C7,R <AV2>
BDAL 02 L	4-B5,R <BE2> 4-C3,L <BE2> 4-C7,R <BE2>
BDAL 03 L	4-B5,R <BF2> 4-C3,L <BF2>
BDAL 04 L	4-B5,R <BH2> 4-D3,L <BH2>
BDAL 05 L	4-C5,R <BJ2> 4-D3,L <BJ2>
BDAL 06 L	4-D3,L <BK2> 4-D5,R <BK2>
BDAL 07 L	4-C5,R <BL2> 4-D3,L <BL2>
BDAL 08 L	4-B3,L <BM2> 4-C5,R <BM2>
BDAL 09 L	4-B3,L <BN2> 4-C5,R <BN2>
BDAL 10 L	4-B3,L <BP2> 4-C5,R <BP2>
BDAL 11 L	4-B3,L <BR2> 4-C5,R <BR2>
BDAL 12 L	4-B3,L <BS2> 4-C5,R <BS2>
BDAL 13 L	4-B3,L <BT2> 4-C5,R <BT2>
BDAL 14 L	4-B3,L <BU2> 4-C5,R <BU2>
BDAL 15 L	4-B3,L <BV2> 4-D5,R <BV2>
BDAL 16 L	4-B5,R <AC1>
BDAL 17 L	4-B5,R <AD1>
BDCOK H	3-C6,L <BA1> 4-B5,R <BA1>
BDIN L	3-B4,L <AH2> 4-C7,R <AH2> 4-D5,R <AH2>
BDMGO H	3-C2,L 3-C4,R
BDMGO L	3-C4,L <AS2> 4-A5,R <AS2>
BDMR L	4-B5,R <AN1> 4-B7,R <AN1>
BDOUT L	3-B4,L <AE2> 4-B5,R <AE2> 4-C7,R <AE2>
BEVNT L	3-C6,L <BR1> 4-D5,R <BR1>
BHALT L	3-D6,L <AP1> 4-B5,R <AP1>
BIAKO L	3-B6,L <AN2> 4-A5,R <AN2>
BINIT L	3-B6,L <AT2> 4-B5,R <AT2>
BIRQ 4 L	4-A5,R <AL2> 4-B7,R <AL2>
BPOK H	4-A7,R <BB1> 4-B5,R <BB1>
BREF L	4-B5,R <AR1>
BRPLY L	3-D6,L <AF2> 4-D5,R <AF2> 4-D6,L <AF2>
BSACK L	4-A7,R <BN1> 4-D5,R <BN1>
BSPARE 1 H	4-A5,R <AA1>
BSPARE 2 H	4-B5,R <AB1>
BSPARE 3 H	4-C5,R <BP1>
BSPARE 4 H	4-B5,R <BC1>
BSPARE 5 H	4-B5,R <BD1>
BSPARE 6 H	4-B5,R <BE1>
BSPARE 7 H	4-B5,R <BF1>
BSYNC L	3-B6,L <AJ2> 4-D5,R <AJ2>
BWTBT L	3-B4,L <AK2> 4-C7,R <AK2> 4-D5,R <AK2>
CDMRQ L	1-C4,L 3-C2,L
CTMER L	1-C5,R 3-C2,L
GND	5-A8,R
GND TTL	5-A2,L
KXT1 AD01 H	1-A2,L 5-C3,R 5-C6,R 5-C8,R 5-D3,R 5-D6,R 5-D8,R 6-D4,R 6-D7,R 7-B7,R
KXT1 AD02 H	1-B2,L 5-C3,R 5-C6,R 5-C8,R 5-D3,R 5-D6,R 5-D8,R 6-D4,R 6-D7,R 7-B7,R
KXT1 AD03 H	1-A2,L 2-C5,R 5-C3,R 5-C6,R 5-C8,R 5-D3,R 5-D6,R 5-D8,R
KXT1 AD04 H	1-B2,L 2-C5,R 5-C3,R 5-C6,R 5-C8,R 5-D3,R 5-D6,R 5-D8,R
KXT1 AD05 H	1-B2,L 2-C5,R 5-B3,R 5-C6,R 5-C8,R 5-D3,R 5-D6,R 5-D8,R
KXT1 AD06 H	1-B2,L 2-C5,R 5-B3,R 5-C6,R 5-C8,R 5-D3,R 5-D6,R 5-D8,R
KXT1 AD07 H	1-C2,L 2-C5,R 5-B3,R 5-C6,R 5-C8,R 5-D3,R 5-D6,R 5-D8,R
KXT1 AD08 H	1-A2,L 2-C5,R 5-B3,R 5-C6,R 5-C8,R 5-D3,R 5-D6,R 5-D8,R
KXT1 AD09 H	1-C2,L 2-C5,R 5-B3,R 5-C4,L 5-C7,L 5-D3,R 5-D4,L 5-D7,L
KXT1 AD10 H	1-C2,L 2-C5,R 5-B3,R 5-C4,L 5-C7,L 5-D3,R 5-D4,L 5-D7,L
KXT1 AD11 H	1-C2,L 2-C5,R 5-B3,R 5-C4,L 5-C7,L 5-D3,R 5-D4,L 5-D7,L
KXT1 AD12 H	1-A2,L 2-C5,R 5-B3,R 5-D1,L 5-D3,R
KXT1 AD13 H	1-C2,L 2-C5,R 5-B3,R 5-C6,R 5-C8,R 5-D3,R 5-D6,R 5-D8,R
KXT1 AD14 H	1-B2,L 2-C5,R 5-D1,L
KXT1 AD15 H	1-B2,L 2-C5,R 5-D1,L
KXT1 B2CLR L	1-B7,R 1-C7,R 1-D2,L 3-C5,R 6-B8,R 7-C3,R
KXT1 BCLR H	1-D2,L 3-B6,R 3-C3,R 6-D4,R 6-D7,R 7-B7,R
KXT1 CAS H	1-A7,R 1-B7,R 1-D2,L
KXT1 CAS L	1-D3,L 3-B3,R
KXT1 CBKRQ H	1-C6,L
KXT1 CPBRQST H	1-A6,L
KXT1 CRDL2 H	1-B6,L
KXT1 CXDL1 H	1-A6,L
KXT1 CXDL2 H	1-A6,L
KXT1 HTRP L	1-C5,L
KXT1 LBS7 H	1-C2,L 2-C5,R
KXT1 PI H	1-A6,R 1-D4,L 3-B3,R
KXT1 PI L	1-A5,L
KXT1 RAS L	1-D4,L 3-B3,R
KXT1 RSYNC L	1-A2,L 4-C7,R
KXT1 SEL0 H	1-C4,L 3-B3,R
KXT1 SEL1 H	1-C4,L 1-D4,R 3-B3,R
KXT1 TBS7 H	1-D4,L 3-C6,R
KXT1 TCOUT H	1-D4,L 3-B3,R
KXT2 BKAK L	2-C6,L 3-C4,R
KXT2 CSDL0 L	2-C3,L 6-D7,R
KXT2 CSDL1 L	2-C3,L 6-D4,R
KXT2 CSKTA L	2-C3,L 5-C7,L 5-D7,L
KXT2 CSKTB L	2-C3,L 5-C4,L 5-D4,L
KXT2 CSPL L	2-C3,L 7-B7,R
KXT2 CSQB L	2-C3,L 3-C3,R 4-C7,R
KXT2 CSRAM L	2-C3,L 5-B3,R 5-C3,R
KXT2 EVNTAK L	2-C6,L 3-C5,R
KXT2 PAAK L	2-C6,L 7-C3,R
KXT2 PBAK L	2-C6,L 7-C3,R
KXT2 SCLK H	2-B6,L 3-B3,R
KXT3 BKRQ H	1-C7,R 3-D4,L
KXT3 DCLO H	2-B5,R 3-B3,R 3-C6,L 6-B8,R
KXT3 DLCLK H	3-C2,L 6-B5,R 6-D4,R 6-D7,R
KXT3 IAK L	2-C7,R 3-D2,L
KXT3 RBS7 H	1-C3,R 3-C6,L
KXT3 REVNT H	1-B7,R 3-D5,L
KXT3 RRPLY H	3-B3,R 3-D6,L
KXT3 RSYNC H	1-A3,R 3-B3,R 3-B6,L
KXT4 IRQ4 H	1-A7,R 4-B7,L
KXT4 PFAIL H	1-B7,R 3-B3,R 4-A7,L
KXT4 RDMR H	3-B3,R 4-B7,L
KXT4 READ H	4-C6,L 7-B8,R
KXT4 READ L	3-C3,R 4-C6,L 5-B3,R 5-C3,R 5-C4,L 5-C7,L 5-D4,L 5-D7,L 6-D4,R 6-D7,R
KXT4 RSACK H	3-B3,R 4-A7,L
KXT4 SEL6 L	4-C6,L 7-B8,R
KXT4 WHB L	4-C6,L 5-C3,R
KXT4 WLB L	4-C6,L 5-B3,R 6-D4,R 6-D7,R 7-B7,R
KXT6 RDAT1 H	6-C5,L 6-D7,R
KXT6 RDAT2 H	6-C2,L 6-D4,R
KXT6 XDAT1P L	6-C5,L 6-C6,R
KXT6 XDAT2P L	6-C2,L 6-C3,R
KXT7 PA0 H	7-B4,L 7-D6,L
KXT7 PA1 H	7-B4,L 7-D6,L
KXT7 PA2 H	7-B4,L 7-D6,L
KXT7 PA3 H	7-A4,L 7-D6,L
KXT7 PA4 H	7-B4,L 7-D6,L
KXT7 PA5 H	7-B4,L 7-D6,L
KXT7 PA6 H	7-B4,L 7-D6,L
KXT7 PA7 H	7-B4,L 7-D6,L
KXT7 PARQST H	1-A7,R 7-C2,L
KXT7 PB0 H	7-A2,L 7-C6,L
KXT7 PB1 H	7-B2,L 7-C6,L
KXT7 PB2 H	7-B2,L 7-C6,L
KXT7 PB3 H	7-B2,L 7-C6,L
KXT7 PB4 H	7-B2,L 7-C6,L
KXT7 PB5 H	7-B2,L 7-C6,L
KXT7 PB6 H	7-B2,L 7-D6,L
KXT7 PB7 H	7-B2,L 7-D6,L
KXT7 PBRQST H	1-A7,R 7-D2,L
KXT7 PC0 H	7-C5,R 7-C6,L
KXT7 PC1 H	7-C6,L 7-D5,R
KXT7 PC2 H	7-C4,L 7-C6,L
KXT7 PC3 H	7-C5,R 7-C6,L
KXT7 PC4 H	7-C6,L 7-D3,L
KXT7 PC5 H	7-C5,R 7-C6,L
KXT7 PC6 H	7-C6,L 7-D3,L
KXT7 PC7 H	7-C5,R 7-C6,L
KXT7 PCJ0 H	7-B7,R 7-C4,L
KXT7 PCJ1 H	7-B7,R 7-D3,L
KXT7 PCJ2 H	7-B6,L 7-C5,R
KXT7 PCJ3 H	7-B6,L 7-C4,L
KXT7 PCJ5 H	7-B6,L 7-C4,L
KXT7 PCJ7 H	7-B6,L 7-C4,L
PAJ0 H	7-A7,R 7-B5,R
PAJ1 H	7-A7,R 7-B5,R
PAJ2 H	7-A7,R 7-B5,R
PAJ3 H	7-A5,R 7-A7,R
PAJ4 H	7-A6,L 7-B5,R
PAJ5 H	7-A6,L 7-B5,R
PAJ6 H	7-A6,L 7-B5,R

SHEET 8 OF 9

PAJ7 H	7-A6,L	7-B5,R				
PBJ0 H	7-A2,R	7-B6,L				
PBJ1 H	7-A6,L	7-B2,R				
PBJ2 H	7-A6,L	7-B2,R				
PBJ3 H	7-A6,L	7-B2,R				
PBJ4 H	7-A7,R	7-B2,R				
PBJ5 H	7-A7,R	7-B2,R				
PBJ6 H	7-A7,R	7-B2,R				
PBJ7 H	7-B2,R	7-B7,R				
PCJ4 H	7-B7,R	7-D5,R				
PCJ6 H	7-B7,R	7-D5,R				
PUP L	1-C5,R	3-C2,L				
R/WHB H	1-D4,L	3-B3,R				
R/WLB H	1-D4,L	3-B3,R				
RDAT1N H	6-B6,R					
RDAT2N H	6-B3,R					
RDL1 H	1-A7,R	2-C6,L	6-C6,L			
RDL2 H	1-A7,R	2-C6,L	6-C3,L			
READY H	1-C5,R	3-C2,L				
RHLB H	3-C2,L	4-A4,R				
RLB H	3-C2,L	4-C4,R				
SFTGND L	1-A3,R	1-A7,R	2-C7,R			
TCLK L	1-C5,R	3-C2,L				
TDAL 00 H	1-D5,R	4-C4,R	5-B2,L	5-C6,R	5-C8,R	6-D3,L
	6-D6,L	7-C7,R				
TDAL 01 H	1-A3,R	1-D5,R	4-C4,R	5-B2,L	5-C6,R	5-C8,R
	6-D3,L	6-D6,L	7-C7,R			
TDAL 02 H	1-B3,R	1-D5,R	4-C4,R	5-B2,L	5-C6,R	5-C8,R
	6-D3,L	6-D6,L	7-C7,R			
TDAL 03 H	1-A3,R	1-D5,R	4-C4,R	5-B2,L	5-C4,L	5-C7,L
	6-D3,L	6-D6,L	7-C7,R			
TDAL 04 H	1-B3,R	1-D5,R	4-C4,R	5-B2,L	5-C4,L	5-C7,L
	6-D3,L	6-D6,L	7-C7,R			
TDAL 05 H	1-B3,R	1-D5,R	4-D4,R	5-B2,L	5-C4,L	5-C7,L
	6-D3,L	6-D6,L	7-C7,R			
TDAL 06 H	1-B3,R	1-D5,R	4-D4,R	5-B2,L	5-C4,L	5-C7,L
	6-D3,L	6-D6,L	7-C7,R			
TDAL 07 H	1-C3,R	1-D5,R	4-D4,R	5-B2,L	5-C4,L	5-C7,L
	6-D3,L	6-D6,L	7-C7,R			
TDAL 08 H	1-A3,R	1-D5,R	1-D6,L	2-C7,R	4-B4,R	5-D2,L
	5-D6,R	5-D8,R	6-D3,L	6-D6,L		
TDAL 09 H	1-C3,R	1-D5,R	2-C7,R	4-B4,R	5-D2,L	5-D6,R
	5-D8,R	6-D3,L	6-D6,L			
TDAL 10 H	1-C3,R	1-C5,R	2-C7,R	4-B4,R	5-C2,L	5-D6,R
	5-D8,R	6-D3,L	6-D6,L			
TDAL 11 H	1-C3,R	1-C5,R	1-C6,L	2-C7,R	4-B4,R	5-C2,L
	5-D4,L	5-D7,L	6-D3,L	6-D6,L		
TDAL 12 H	1-A3,R	1-C5,R	3-C3,R	4-B4,R	5-C2,L	5-D4,L
	5-D7,L	6-D3,L	6-D6,L			
TDAL 13 H	1-C3,R	1-C5,R	1-D6,L	4-B4,R	5-C2,L	5-D4,L
	5-D7,L	6-C3,L	6-C6,L			
TDAL 14 H	1-B3,R	1-C5,R	1-C6,L	4-B3,R	5-C2,L	5-D4,L
	5-D7,L	6-C3,L	6-C6,L			
TDAL 15 H	1-B3,R	1-C5,R	1-D6,L	4-B4,R	5-C2,L	5-D4,L
	5-D7,L	6-C3,L	6-C6,L			
TDIN H	3-B4,R	3-C2,L				
TDOUT H	3-C2,L	3-C4,R				
TEVNT H	3-C7,R	6-D2,L				
THALT H	3-D7,R					
TIAKO H	3-B6,R	3-C2,L				
TSYNC H	3-B6,R	3-C2,L				
TWTBT H	3-B4,R	3-C2,L				
XDL1 H	1-A7,R	2-C6,L	6-C6,L			
XDL2 H	1-A7,R	2-C6,L	6-C3,L			
XHB H	3-D2,L	4-A4,R				
XLB H	3-C2,L	4-C4,R				

APPENDIX G
GLOSSARY

AD01–AD15 – A 15-bit on-board address bus used to address memory and peripheral devices. Generated by two 8-bit latches that are loaded from the TDAL bus. See also BBS7.

AI0–AI7 – Input lines used by the microprocessor for interrupts and DMA requests.

ASPI – Microprocessor transaction that allows the microprocessor to recognize and accept pending interrupts or DMA requests.

Autobaud – Self-adjusting baud rates for SLU1 only. Implemented by firmware in the optional Macro-ODT ROM.

BBS7 – LSI-11 bus signal indicating that the device addressed is in the I/O page.

BDAL 0–15 – Multiplexed data and address lines of the LSI-11 bus connected through the backplane.

BDCOK – LSI-11 bus signal that goes high 3 ms after dc power is applied and goes low 4 ms after ac power is removed.

BDIN – LSI-11 bus data input strobe.

BDMGI – LSI-11 bus signal from the BDMGO bus pin. It enters each module on the BDMGI pin and exits on the BDMGO pin. It represents the bus grant for a DMA transaction.

BDMGO – See BDMGI.

BDMR – DMA request signal from the LSI-11 bus.

BDOUT – LSI-11 bus data output strobe.

BEVNT – LSI-11 bus signal used to generate REVNT. Can be used to initiate an interrupt.

BHALT – LSI-11 bus halt signal used for a priority 7 interrupt that vectors through location 140.

BINIT – LSI-11 bus signal used to initialize all the devices on the bus.

BIRQ4 – LSI-11 bus, level 4 priority interrupt request that is used to initiate the internal IRQ4 signal.

BKRQ – Internal control signal initiated by BHALT or BREAK detect from terminal.

BPOK – LSI-11 bus signal that goes high 70 ms after BDCOK and goes low when ac power is lost.

BREAK – Initiated by pressing the BREAK key. Causes the transmission line to the SLU to be forced to the space state (logical zero). This condition is sensed by SLU1 and causes the SBC-11/21 PLUS to generate BKRQ that can be used for interrupts.

BRPLY – Slave's acknowledge of an LSI-11 bus cycle.

BSACK – Acknowledges receipt of a DMA grant signal.

BSYNC – LSI-11 bus cycle control signal.

BWTBT – LSI-11 bus write byte control signal.

CAS – An output from the microprocessor that acts as data strobe. Used for the read/write, DMA, and ASPI transactions.

Condition codes – The least significant four bits of the processor status word that monitor the results of the last instruction executed.

Configuration – Allows the user to select optional features of the module by inserting jumper wires.

Control and status register (CSR) – Internal register in an I/O interface that allows the program to control and monitor the operation of that interface.

Control word – The data contained in the control register of the parallel I/O chip that determines the configuration of the parallel I/O interface.

COUT – An output from the microprocessor clock that is asserted once during each microcycle.

CSKTA – The RAM/ROM socket set A chip select strobe.

CSKTB – The RAM/ROM socket set B chip select strobe.

CTMER – Time-out interrupt that has the same effect as HALT.

Cycle slip – This condition exists when the READY input is pulsed while RAS is asserted. It causes the microprocessor to be idle, and no transactions occur.

DATI – LSI-11 bus transaction that transfers sixteen bits of data from the slave to the master.

DATO – LSI-11 bus transaction that transfers sixteen bits of data from the master to the slave.

DATO(B) – LSI-11 bus transaction that transfers eight bits of data from the master to the slave.

DMA – Direct memory access for transferring blocks of data without program intervention.

DMA transaction – A microprocessor transaction during which the microprocessor gives up bus mastership to another device for direct transfer of memory data.

EIA RS-232C – Electronics Industries Association serial line interface standard.

EIA RS-423 – Electronics Industries Association serial line interface standard.

Fetch/read – Microprocessor transaction that transfers data from memory or I/O into the microprocessor. The data may be an instruction (fetch) or an operand (read).

Firmware – The programs that reside in the PROM or ROM hardware.

FPLA – Field programmable logic array. Used to decode memory addresses.

HALT – The highest priority interrupt. Causes the microprocessor to go to the restart address and loads the PSW with 340.

Handshaking protocol – The series of events used to establish data transfers.

IAK – Microprocessor transaction to acknowledge an interrupt and secure a vector from an on-board location or from the LSI-11 bus.

Interrupts – Interruption of the normal program execution to service an external request.

Interrupt protocol – Signal sequence required to initiate and service interrupts.

Interrupt vector – The location in which the address of the interrupt service routine is stored.

IRQ4 – See BIRQ4

KXT11-A2 – See Macro-ODT.

KXT11-A5 – See Macro-ODT.

LSI-11 bus – An asynchronous bus that provides interconnections for LSI-11 type modules.

Macro-ODT – The KXT11-A5 and KXT11-A2 optional firmware for the SBC-11/21 PLUS.

Maskable – A priority level that can be inhibited by loading the PSW with a higher priority code.

Memory mapping – Creating regions of memory via jumper configurations to determine the on-board portions and the LSI-11 bus portions of memory.

Microcycle – The time necessary to execute one microinstruction. A transaction may use three or four microcycles.

Mode register – An internal microprocessor register used to define the start and restart addresses.

Nibble – The upper or lower half of a byte that consists of four bits.

Nonmaskable – A priority level that is higher than the level selectable by the PSW.

NOP – A transaction that produces no useful output. It is used to introduce a delay or wait period.

Parallel I/O – Parallel data interface.

Parallel I/O handshaking – Control signals used to establish parallel data transfers.

PARQST – Parallel I/O port A interrupt request.

PBRQST – Parallel I/O port B interrupt request.

PI (priority in) – A microprocessor output signal used to strobe interrupt and DMA requests into the microprocessor.

Power fail (PFAIL) – A nonmaskable interrupt caused by a power failure that causes the microprocessor to vector through location 24 to the power fail routine.

Priority – Bits 5–7 of the PSW. Used to define the priority level of the microprocessor.

PSW register – A microprocessor register that contains the processor status word (PSW).

PUP – An input to the microprocessor that controls the power-up sequence. When it is switched from high to low, the microprocessor power-up sequence is initiated.

RAM – Random access memory defined as read/write memory.

RAS – Microprocessor output used as an address strobe in read/write, IAK, and DMA transactions.

RCSR – Serial line receiver control status register.

RDBR – Serial line receiver data buffer register.

RDL1 – Serial line receiver number 1 interrupt signal.

RDL2 – Serial line receiver number 2 interrupt signal.

READY – Input to the microprocessor that causes cycle slips when pulsed.

Restart address – Jumper-selectable address that the microprocessor jumps to when executing a HALT interrupt.

REVNT – See BEVNT.

ROM – Read only memory that cannot be written into.

R/–WHB – A microprocessor output that is low for high byte write transactions and high for read transactions.

R/–WLB – A microprocessor output that is low for low byte write transactions and high for read transactions.

RTI – Return from interrupt instruction.

SEL0/SEL1 – Microprocessor outputs used to define the transaction being performed.

Serial I/O – Asynchronous serial line units for the transfer of serial data. SLU1 and SLU2 are two such units used in the SBC-11/21 PLUS.

Slew rate resistor – A resistor installed on the module that is compatible with the baud rate selected.

Split speed – A process that sends data at one baud rate and receives data at a different baud rate. SBC-11/21 PLUS does not support split speed operation.

Spurious halts – Halt conditions that are not programmed or introduced from an error condition.

Stack pointer – The register that contains the address of the last word stored on the stack.

Start address – A jumper-selectable address that the microprocessor goes to during power-up.

TCSR – Serial line transmitter control status register.

TDAL 0–15 – Internal on-board bus used for multiplexed data and address lines. See BDAL 0–15.

TDBR – Serial line transmitter data buffer register.

Trace bit – Bit 4 of the PSW that causes a trap to location 14.

Transaction – A sequence of microcycles used to complete a designated microprocessor function such as read, write, ASPI, or IAK.

Tri-state – A high impedance condition of the bus lines.

Vector address – Memory location the microprocessor accesses for the address of the interrupt service routine during an interrupt.

Wait state – A condition during which the microprocessor performs no useful transactions while waiting for a response or data.

Wake-up circuit – Holds BDCOK negated for 70 ms after dc power has been applied.

APPENDIX H
SBC-11/21 PLUS AND SBC-11/21 DIFFERENCES

H.1 INTRODUCTION
This appendix is primarily for a current user of the SBC-11/21 who is either considering or in the process of using the SBC-11/21 PLUS, instead of the SBC-11/21.

H.2 OVERVIEW
SBC-11/21 PLUS is an enhanced SBC-11/21. The differences between SBC-11/21 and SBC-11/21 PLUS are as follows:

1. On-board RAM. SBC-11/21 PLUS is shipped with 16Kb of SRAM, increased from the 4Kb shipped with the SBC-11/21.

2. In order to make use of the additional RAM on the SBC-11/21 PLUS, new memory maps are used selected by the FPLA located in socket XE41. (See Figure 1-1.)

3. Two of the new maps make provision for the Macro-ODT ROMs sold as option KXT11-A5. These now reside in memory from octal 164000 to 174000.

4. If the KXT11-A5 option is installed in socket set A and the FPLA shipped with the SBC-11/21 PLUS (installed in socket XE41) is used, then the SBC is configured for the SBC-11/21 PLUS mode. If the KXT11-A2 option is installed in socket set A and the optional FPLA (option 23-77C6-00) is installed in socket XE41 (see Figure 1-1), then the SBC is configured for the SBC-11/21 mode. This enables the user to operate the SBC-11/21 PLUS with the same memory maps as the SBC-11/21, but restricts its operation to the same amount of on-board memory and the same storage devices as the SBC-11/21.

5. Additional mass storage devices are supported on the SBC-11/21 PLUS with the KXT11-A5 option. RL01/RL02 and RD51/RX50 in addition to TU58 and RX01/RX02 mass storage devices.

6. The non-JEDEC pinout memories are no longer supported on SBC-11/21 PLUS. Not used on the SBC-11/21 PLUS are 1K × 8 memory chips.

7. Up to 16Kb of additional SRAM may now be plugged into socket set B.

8. A number of changes have been made to the wirewrap pins. They have been simplified where possible, with provision made for some larger memory devices.

H.3 OPTIONAL FPLA
The SBC-11/21 PLUS is shipped containing an FPLA in socket XE41. This creates the four memory maps shown in Figure 2-7. It maps in the memory devices (16Kb) designated E33 and E40. The coding for this FPLA is found in Table H-1, and may be contained in a device such as the Signetics 82S100. If the

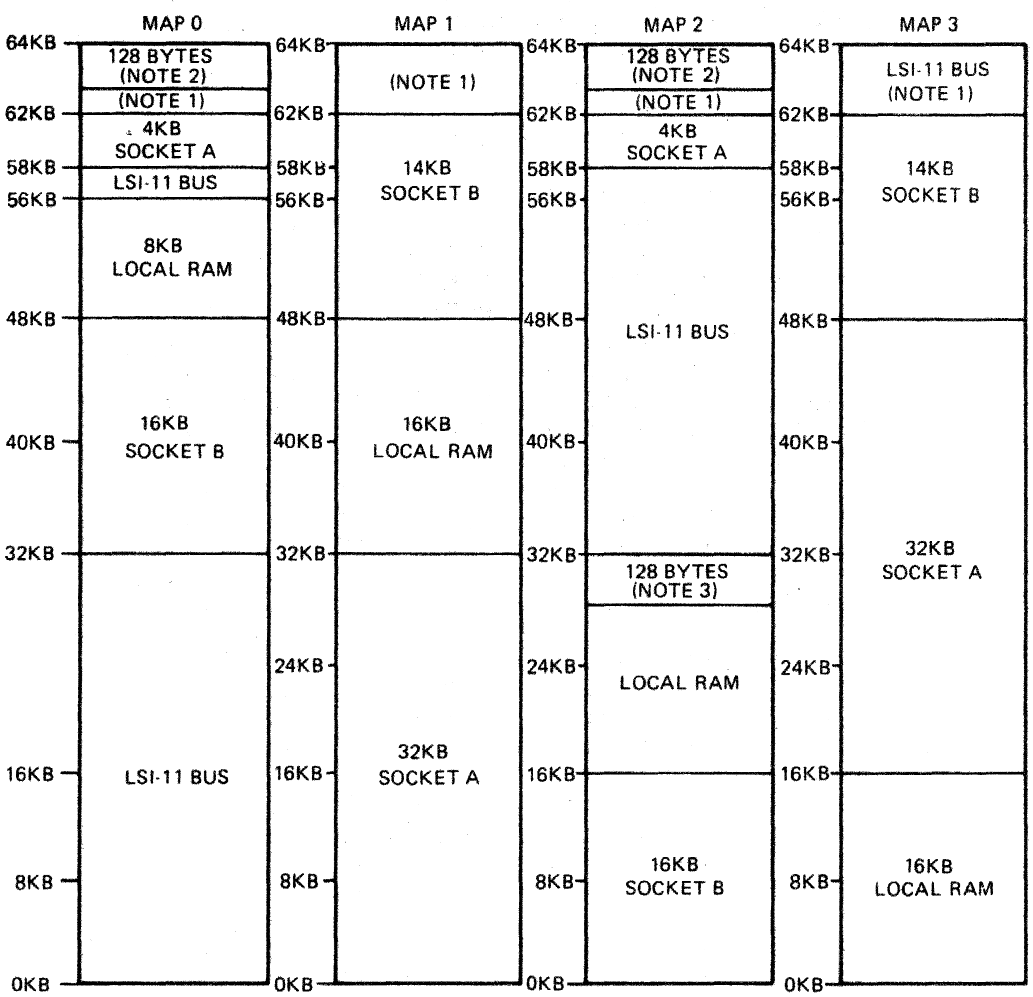
Table H-1 FPLA Code for the SBC-11/21 PLUS

```
               111111              *A LLLLHLLL
               5432109876543210

*P 00     *I -H--HHH---HHH--L     *F A.......
*P 01     *I -H--HHH---HHH--H     *F .....A..
*P 02     *I -HHHHHH---HHLHH-     *F .....A..
*P 03     *I -HHHHHH---HHLHL-     *F ..A.....
*P 04     *I -HLHHHH---HHLH--     *F .....A..
*P 05     *I -H-HHHH---HHLL--     *F .....A..
*P 06     *I -H-LHHH---HHL---     *F .....A..
*P 07     *I -H--LHH---HH----     *F .....A..
*P 08     *I -H--HHH---HLH---     *F .....A..
*P 09     *I -HHHHHH---HLLH--     *F .....A..
*P 10     *I -HLHHHH---HLLHH-     *F .....A..
*P 11     *I -HLHHHH---HLLHL-     *F ......A.
*P 12     *I -H-HHHH---HLLL--     *F .....A..
*P 13     *I -H-LHHH---HLL---     *F .....A..
*P 14     *I -H-HLHH---HLH---     *F .....A..
*P 15     *I -H-LLHH---HLHH--     *F .....A..
*P 16     *I -HHLLHH---HLHL--     *F .....A..
*P 17     *I -HLLLHH---HLHLH-     *F .....A..
*P 18     *I -HLLLHH---HLHLL-     *F ...A....
*P 19     *I -H--LHH---HLL---     *F .....A..
*P 20     *I -H---LH---H-----     *F .....A..
*P 21     *I -L-----HHH------     *F .....A..
*P 22     *I -H----H---L----L     *F .A......
*P 23     *I -H----H---L----H     *F .......A
*P 24     *I -H----L---H----L     *F .A......
*P 25     *I -H----L--------H     *F .......A
*P 26     *I -H----L---L----L     *F .....A..
*P 27     *I LL-----HHL-----L     *F A.......
*P 28     *I -L-----HHL-----H     *F .......A
*P 29     *I HL-----HHL-----L     *F .....A..
*P 30     *I LL-----LH------L     *F .......A
*P 31     *I LL-----LH------H     *F A.......
*P 32     *I HL-----LH------L     *F .....A..
*P 33     *I HL-----LH------H     *F .A......
*P 34     *I LL------L------L     *F .....A..
*P 35     *I LL------L------H     *F .A......
*P 36     *I HL-----LL------L     *F .......A
*P 37     *I HL-----LL------H     *F A.......
*P 38     *I HL-----HL------H     *F .A......
*P 39     *I HL-----HLL-----L     *F A.......
*P 40     *I HL----LHLH-----L     *F A.......
*P 41     *I HL----HHLHL----L     *F A.......
*P 42     *I HL---LHHLHH----L     *F A.......
*P 43     *I HL---HHHLHHL---L     *F A.......
*P 44     *I HL--LHHHLHHH---L     *F A.......
*P 45     *I HL--HHHHLHHHL--L     *F A.......
*P 46     *I HL--HHHHLHHHH--L     *F .....A..
*P 47     *I ----------------     *F ........
```

user wishes to support the memory maps which are provided on the SBC-11/21 (Figure H-1), then either option 23-77C6-00 may be purchased from Digital or the FPLA code in Table H-2 may be coded into an 82S100. If this mapping scheme is used then it should be noted that only 4Kb of the on-board memory will be available to the user.

Additional mapping schemes are possible but not supported by Digital.



NOTES:

1. THIS SECTION CONTAINS THE LOCAL I/O ADDRESSES FOR THE SLUS AND PPI. ALL UNASSIGNED ADDRESSES ARE ASSUMED TO RESIDE ON THE LSI-11 BUS.
2. ADDRESSES 177777 - 177600 IN MAPS 0 AND 2 ARE RAM SCRATCHPAD LOCATIONS USED BY MACRO-ODT.
3. ADDRESSES 77777 - 77600 IN MAP 2 ARE ALLOCATED TO THE LSI-11 BUS.

MR-12140

Figure H-1 SBC-11/21 PLUS Memory Maps

Table H-2 Option 23-77C6-00 FPLA Code

```
                111111                  LLLLLLLL
                5432109876543210

*P 00    *I  -H--HHH---HHH---  *F   .....A..
*P 01    *I  -HHHHHH---HHLHH-  *F   .....A..
*P 02    *I  -HHHHHH---HHLHL-  *F   ..A.A...
*P 03    *I  -HLHHHH---HHLH--  *F   .....A..
*P 04    *I  -H-HHHH---HHLL--  *F   .....A..
*P 05    *I  -H-LHHH---HHL---  *F   .....A..
*P 06    *I  -H--LHH---HH----  *F   .....A..
*P 07    *I  -H--HHH---HLH---  *F   .....A..
*P 08    *I  -HHHHHH---HLLH--  *F   .....A..
*P 09    *I  -HLHHHH---HLLHH-  *F   .....A..
*P 10    *I  -HLHHHH---HLLHL-  *F   ....A.A.
*P 11    *I  -H-HHHH---HLLL--  *F   .....A..
*P 12    *I  -H-LHHH---HLL---  *F   .....A..
*P 13    *I  -H-HLHH---HLH---  *F   .....A..
*P 14    *I  -H-LLHH---HLHH--  *F   .....A..
*P 15    *I  -HHLLHH---HLHL--  *F   .....A..
*P 16    *I  -HLLLHH---HLHLH-  *F   .....A..
*P 17    *I  -HLLLHH---HLHLL-  *F   ...AA...
*P 18    *I  -H--LHH---HLL---  *F   .....A..
*P 19    *I  -H---LH---H-----  *F   .....A..
*P 20    *I  LH----H---L----L  *F   .A..A..A
*P 21    *I  HH----H---L-----  *F   .....A..
*P 22    *I  LH----H---L----H  *F   .....A..
*P 23    *I  -H----L---H-----  *F   A...A...
*P 24    *I  -H---HL---L-----  *F   A...A...
*P 25    *I  -H---LL---LH----  *F   A...A...
*P 26    *I  -H--HLL---LL----  *F   A...A...
*P 27    *I  -H--LLL---LLH---  *F   A...A...
*P 28    *I  -H-HLLL---LLL---  *F   A...A...
*P 29    *I  -H-LLLL---LLLH--  *F   A...A...
*P 30    *I  -HHLLLL---LLLL--  *F   A...A...
*P 31    *I  -HLLLLL---LLLLH-  *F   A...A...
*P 32    *I  -HLLLLL---LLLLL-  *F   .....A..
*P 33    *I  -L-----HHL------  *F   .....A..
*P 34    *I  -L-----LHH------  *F   .....A..
*P 35    *I  -L-----LHL------  *F   .....A..
*P 36    *I  LL-----HL-------  *F   .....A..
*P 37    *I  HL-----HL------L  *F   .....A..
*P 38    *I  HL-----HL------H  *F   .A..A...
*P 39    *I  LL-----LL------L  *F   .....A..
*P 40    *I  HL-----LL------H  *F   ....A..A
*P 41    *I  HL-----LLH-----L  *F   .A..A...
*P 42    *I  LL-----LLH-----H  *F   .....A..
*P 43    *I  HL-----LLL-----L  *F   ....A..A
*P 44    *I  LL----HLLL-----H  *F   .A..A...
*P 45    *I  LL----LLLL-----H  *F   ....A..A
*P 46    *I  -LLLLLLHHHLLLLL-  *F   .....A..
*P 47    *I  -L----LHHH------  *F   A...A...
```

H.4 MEMORY MAPS

The memory maps used in the SBC-11/21 PLUS are shown in Figure 2-7. These are replaced by those found in Figure H-1 if the optional FPLA (option 23-77C6-00) is used.

H.5 KXT11-A2 AND KXT11-A5 MACRO-ODT ROMS

If the user wishes to use the SBC-11/21 PLUS memory maps and desires to have the capability of using either RL01/RL02 or RD51/RX50 storage devices in the system, then the KXT11-A5 optional ROMs should be used. The ROMs are mapped in addresses 164000 to 173776 (octal) when using either map 0 or map 2. The code resides in two 2K × 8 memory chips. It is important to note that the code for addresses 164000 to 167776 is contained in the UPPER address space of the chip, and the code for addresses 170000 to 173776 is contained in the LOWER address space of the chip.

For those users who wish to maintain the SBC-11/21 memory maps and who do not wish to use either the RL01/RL02 or the RD51/RX50 storage devices, the KXT11-A2 ROMs should be used. The ROMs are mapped in addresses 170000 to 173776 (octal) when using map 0.

For detailed configuration help, consult either the *KXT11-AB SBC-11/21 PLUS Configuration Guide* or Chapter 2 of this manual.

H.6 MEMORY DEVICES SUPPORTED

Devices now supported by the SBC-11/21 PLUS are listed in Tables 2-10 and 2-11. The non-JEDEC standard pinout memories as well as all 1K × 8 chips are no longer usable with the SBC-11/21, when configured in any mode.

Devices no longer supported on Tables 2-10 and 2-11 are:

Socket set A:

EPROMs

INTEL	2758	1K × 8
TI	TMS2508	1K × 8
TI	TMS2564	8K × 8

PROMs

INTEL	3628	1K × 8
SIGNETICS	82LS181	1K × 8

Socket set B:

EPROMs

INTEL	2758	1K × 8
TI	TMS2508	1K × 8
TI	TMS2564	8K × 8

PROMs

INTEL	3628	$1K \times 8$
SIGNETICS	82LS181	$1K \times 8$

This applies when the SBC-11/21 PLUS is used in either SBC-11/21 PLUS or SBC-11/21. Examples of memories supported by SBC-11/21 PLUS are found on Tables 2-10 and 2-11.

H.7 WIREWRAP CONFIGURATION COMPARISONS

In total, there are four fewer wirewrap pins on SBC-11/21 PLUS than on SBC-11/21. While eighteen of the functionality pins have been removed, seven have been added to accommodate additional memory chips and seven ground or power pins have been added. Some of the functionality pins were incorporated within the Gate Array (E20) and thus are unreachable.

Table H-3 lists the functional equivalent pins for the SBC-11/21 PLUS as compared to the SBC-11/21.

Those posts on SBC-11/21 that are no longer available are:

M2	Clock system input
M9	–IAK output
M10	–CTMER interrupt request input
M11	Clock osc. output
M14	TMER (timeout error) output
M31	READ H strobe
M43	Socket set A, high and low byte, pin 2
M60	Socket set A, high and low byte, pin 22
M38	Socket set A, high and low byte, pin 20
M47	Sockets A and B, high and low byte, pin 21
M41	Socket set A, chip select (–CSKTA)
M45	Address line 11
M34	Socket set B, chip select (–CSKTB)
M42	Socket set B, high and low byte, pin 2
M62	Socket set B, high and low byte, pin 22
M32	Socket set B, high and low byte, pin 20
M36	Address line 13
M64	Read strobe (–READ)

H.8 RT-11 ON SBC-11/21 PLUS

The SBC-11/21 PLUS supports RT-11 V5.1, and supports either the RT-11(SJ) or the RT-11(FB) monitor. Additionally, SBC-11/21 PLUS when operated in SBC-11/21 mode can also run RT-11(FB).

H.8.1 SBC-11/21 PLUS and RT-11(SJ) (SBC-11/21 PLUS mode)

The SBC-11/21 PLUS should be configured with socket set B containing $8K \times 8$ SRAMs. Memory map 2 is normally used so that on-board memory is mapped in between 0 and 100000 (octal). If more than 16Kb of memory is desired when running RT-11(SJ) then map 0 may be used as for the FB monitor below. The KXT11-A5 option should be plugged into socket set A to support RT-11. In this arrangement SBC-11/21 PLUS can boot from RX01/RX02, TU58, RL01/RL02 or RD51/RX50.

H.8.2 SBC-11/21 PLUS with RT-11(FB) (SBC-11/21 PLUS mode)

The SBC-11/21 PLUS should be configured with socket set B containing 8K × 8 SRAMs for a full 56Kb RT-11(FB) system. Memory map 0 is required. The KXT11-A5 option should be plugged into socket set A. In this arrangement SBC-11/21 PLUS can boot from RX01/RX02, TU58, RL01/RL02 or RD51/RX50. A Q-Bus memory board must be resident on the Q-Bus mapped into the memory space 0 to 100000 (octal).

H.8.3 SBC-11/21 PLUS with RT-11(FB) (SBC-11/21 mode)

The optional FPLA (21-77C6-00) should be inserted in the socket XE41. This, along with the KXT11-A2 option in socket set A will put the SBC-11/21 PLUS into the SBC-11/21 mode. Memory map 0 should be selected and socket set B is unused. There must be a minimum of 56Kb of memory on a memory module plugged into the Q-Bus and mapped into the space 0 to 160000. In this set-up the SBC-11/21 PLUS will emulate an SBC-11/21 in that the SBC-11/21 memory maps will be available. The amount of memory on-board the SBC-11/21 maps in will also be the same; i.e. only 4Kb of the on-board 16Kb of RAM available is utilized.

H.9 SBC-11/21 PLUS AND MICROPOWER/PASCAL

SBC-11/21 PLUS supports Micropower/Pascal V1.5. A current application built under V1.4 or earlier for running on SBC-11/21 will NOT run as is. In order to run an existing or new application on SBC-11/21 PLUS, it must be built under Version 1.5. When building, the only difference between SBC-11/21 PLUS and SBC-11/21 is that it is necessary to specify "PROCESSOR type=T-11+" in the configuration file. Note that the KXT11 macro options are unchanged. If T-11 is selected as the processor type, the application will get built for running on either an SBC-11/21 or an SBC-11/21 PLUS operating in SBC-11/21 mode. (See the definition of SBC-11/21 and SBC-11/21 PLUS modes in Section H.3.)

Table H-3 Functional Equivalent Pins (Cont)

SBC-11/21 PLUS	SBC-11/21	Appendix F Schematic Sheet Number	Description
M1	M1	5	System +5 V power (+5 VNCR)
M2	M3	2	System GND
M3	M12	2	High logic level (+3 Vdc)
M4	M4	3	Wake-up circuit diode, anode side
M5	M5	3	Receive side of BHALT line transceiver
M6	M6	3	Wake up circuit diode, cathode side (+5 VNCR)
M7	M7	5	Socket set A, high and low byte, pin 26
M8	M8	3	BREAK request clock line
M9	M18	6	High logic level (+3 Vdc)
M10	N/A	5	Address line 14
M11	M24	6	System GND
M12	N/A	5	High logic level (+5 VNCR)
M13	N/A	5	Socket set B, high and low byte, pin 1
M14	N/A	5	Socket set A, high and low byte, pin 1
M15	M15	5	+5 Vdc power distribution to support static RAM
M16	M16	5	Battery backup +5 Vdc power source
M17	M17	6	Serial line unit (SLU) 1 BREAK detect, interrupt request output
M18	M50	2	High logic level (+3 Vdc)
M19	M19	6	60 Hz real-time clock output
M20	M20	3	Transmit side of BHALT line transceiver
M21	M21	2	Memory map select (MSB)
M22	M22	1	Start address control (TDAL 15)
M23	M23	6	Transmit side of BEVNT line transceiver
M24	M24	5	System GND
M25	M25	2	Memory map select (LSB)
M26	M26	1	Start address control (TDAL 14)
M27	M27	6	50 Hz real-time clock output
M28	M28	6	800 Hz real-time clock output
M29	M29	1	System GND
M30	M30	1	Start address control (TDAL 13)
M31	M51	3	System GND
M32	M55	1	System GND
M33	M33	5	Socket set B, high byte, pin 27
M34	N/A	5	High logic level (+5 VCR)
M35	M35	5	Socket set B, low byte, pin 23

Table H-3 Functional Equivalent Pins (Cont)

SBC-11/21 PLUS	SBC-11/21	Appendix F Schematic Sheet Number	Description
M36	N/A	5	High logic level (+5 VNCR)
M37	M37	5	Socket set A, low byte, pin 27
M39	M39	5	Socket set A, high byte, pin 27
M40	M40	5	Socket set A, low byte, pin 23
M41	N/A	5	High logic level (+5 VCR)
M42	N/A	5	Address line 15
M44	M44	5	Address line 12
M46	M46	5	High logic level for PROMs (+5 VNCR)
M48	M48	5	Socket set B, low byte, pin 27
M49	M49	7	Port B buffer direction control
M50	M50	2	High logic level (+3 Vdc)
M51	M51	7	System GND
M52	M52	7	Port A buffer direction control
M53	M53	7	Port C buffered output, to J3, pin 7
M54	M54	7	Port C PC6 output (8255A-5 pin 11)
M55	M55	5	System GND
M56	M56	5	High byte write strobe (-WHB)
M57	M57	5	Low byte write strobe (-WLB)
M58	M58	7	Port C PC4 output (8255A-5 pin 13)
M59	N/A	5	Socket set B, high and low byte, pin 28
M61	M61	5	Socket set A, high byte, pin 23
M63	M63	5	Socket set B, high byte, pin 23
M64	N/A	5	RAM, high and low byte, pin 26
M65	M65	7	Port C buffered output, to J3 pin 5
M66	N/A	5	Socket set B, high and low byte, pin 26
M74	M13	1	HALT request line

N/A means no longer available on the SBC-11/21 PLUS.

INDEX

A

AD01–AD15, 8-21
Addressing modes, 7-1
AI0–07, 8-15
Architecture, 5-1
ASPI, 8-8
Autobaud, 6-1

B

Backplane, 2-22
BBS7, 9-4, 9-20
BCLR, 8-6, 8-25
BDCOK, 8-28, 9-17
BDIN, 9-19
BDMGI, 9-8, 9-20
BDMGO, 9-10, 9-20
BDMR, 8-29, 9-17
BDOUT, 8-32, 9-18
BEVNT, 2-11, 5-5, 9-18
BHALT, 2-9, 5-5, 9-17
BINIT, 9-20
BIRQ4, 2-11, 5-5, 9-19
BKRQ, 2-11, 5-5
Bootstrap, 4-6
BPOK, 8-28, 9-17
BREAK, 6-1
BRPLY, 9-19
BSACK, 9-18
BSYNC, 9-19
Bus control, 8-33
Bus cycle, 9-3
Bus signals, 9-1, 9-3
BWTBT, 8-31, 9-19
Byte, 7-25, 8-23, 8-31

C

CAS, 8-5
Clock, 8-28
Clock control, 8-29
Condition codes, 7-28
Configuration, 2-3, 2-5
Control register, 6-22
Control word, 6-22, 6-23
COUT, 8-6
CSKTA, 8-21
CSKTB, 8-21
CTMER, 2-8, 5-5
Cycle slip, 8-33

D

Data transfer, 9-3
DATI, 9-5
DATO, 9-8
DATOB, 9-8
Deferred addressing, 7-14
Direct addressing, 7-4
DMA, 5-4, 8-29, 9-8
DMA bus master, 9-8
DMA transaction, 9-8
Double operand address, 7-41

E

EIA RS-232C, 2-25
EIA RS-423, 2-25

F

Fetch/read, 8-6
Firmware, 4-1
FPLA, 8-21
Framing error, 6-5
Functional block diagram, 8-2, 8-3

H

HALT, 5-5, 8-19
Handshaking protocol, 6-25
Hardware memory stack, 5-2

I

IAK, 8-8
Initialization, 8-1, 9-14
Instruction set, 7-22
Instruction set list, 7-26
Interrupts, 2-8, 8-14
Interrupt protocol, 9-12
Interrupt set/reset, 8-8, 8-16
Interrupt vector, 5-4, 8-17
IRQ4, 8-15

J

Jumper wire, 2-1

K

KXT11-A5, 4-1

L

Loopback connector, 2-28
LSI-11 bus, 9-1

M

Macro-ODT, 4-1
Macro-ODT commands, 4-3
Maskable, 5-4
Memory maps, 2-16, 2-19, 5-6
Microcycle, 8-6
Microcycle slip, 8-17
Microprocessor, 8-1
Mode 0, 6-10
Mode 1, 6-11
Mode 2, 6-18
Mode selection, 6-22

N

Nibble, 6-11
Nonmaskable, 5-4
NOP, 8-8

O

Overrun error, 6-4

P

Parallel I/O, 6-7
Parallel I/O flowchart, 6-9
Parallel I/O handshaking, 6-25
Parallel I/O initialization, 6-25
PARQST, 5-5, 8-17
PBRQST, 5-5, 8-17
PC addressing, 7-17
PI (priority in), 8-5
Power fail (PFAIL), 8-19
Power fail routine, 8-19
Power-up/down protocol, 5-4, 8-27, 9-15
Ports A, B, C, 6-8
Priority, 5-2
Program counter (PC), 7-17, 8-1
Programmable baud rates, 6-1
Programmable baud rates select, 6-6
PSW register, 5-1
PUP, 8-4

R

RAM, 2-16, 8-21
RAS, 8-5
RCSR, 6-3
RDBR, 6-3
RDL1, 5-5, 8-17, 8-23
RDL2, 5-5, 8-17, 8-23
Read, 8-6
Read/write, 8-5, 8-31
READY, 8-5
Reply time-outs, 8-19
Restart address, 8-14
REVNT, 8-15, 8-17
ROM, 2-16, 8-21
R/—WHB, 8-5
R/—WLB, 8-5
RTI, 7-64

S

SEL0/SEL1, 8-5
Serial I/O, 2-12, 6-1, 8-23
Single operand address, 7-3
Slew rate resistor, 2-25
SLU1/SLU2, 6-1, 8-23
SLU programming, 6-1
Split speed, 6-1
Spurious halts, 4-9
Stack pointer, 5-1
Standard factory configuration, 2-7
Start address, 2-8
Status register, 5-1

T

TCSR, 6-3
TDAL, 8-1
TDBR, 6-3
Terminal problems, 4-9
TMER, 8-33
TSYNC, 8-31
Transactions, 8-6
Transmitter interrupt enable, 6-5
Tristate, 8-8

U

Unsupported options, 3-6

V

Vector address, 5-4
Verification procedure, 2-28

W

Wake-up circuit, 8-27
Write transaction, 8-8

X

XDL1, 5-5, 8-15, 8-23
XDL2, 5-5, 8-15, 8-23
XTL0/XTL1, 8-28