

# **SYSTEM 80/30 MICROCOMPUTER USER'S GUIDE**

Manual Order Number: 9800710A

The information in this document is subject to change without notice.

Intel Corporation makes no warranty of any kind with regard to this material, including, but not limited to, the implied warranties of merchantability and fitness for a particular purpose. Intel Corporation assumes no responsibility for any errors that may appear in this document. Intel Corporation makes no commitment to update nor to keep current the information contained in this document.

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

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

ICE  
INSITE  
INTEL  
INTELLEC  
ISBC

LIBRARY MANAGER  
MCS  
MEGACHASSIS  
MICROMAP  
MULTIBUS

PROMPT  
RMX  
UPI  
μSCOPE



This manual provides general information, installation instructions, operating and programming information, and principles of operation for the System 80/30 Microcomputer. Additional information is available in the following documents:

- *Intel iSBC 80/30 Single Board Computer Hardware Reference Manual*, Order No. 9800611.
- *Intel 8080/8085 Assembly Language Programming Manual*, Order No. 9800301.
- *Intel MCS-85 User's Manual*, Order No. 98-366.
- *Intel iSBC 655 System Chassis Hardware Reference Manual*, Order No. 9800709.
- *Intel iSBC 635 Power Supply Hardware Reference Manual*, Order No. 9800298.
- *Intel iSBC 604/614 Cardcage Hardware Reference Manual*, Order No. 9800708.



# CONTENTS

## CHAPTER 1

### GENERAL INFORMATION PAGE

Introduction .....	1-1
System Description .....	1-1
Monitor Program Description .....	1-2
Documentation Supplied .....	1-2
Standard Equipment and User Supplied Equipment .....	1-2
Specifications .....	1-2

## CHAPTER 2

### PREPARATION FOR USE

Introduction .....	2-1
Unpacking and Inspection .....	2-1
Installation Considerations .....	2-1
Power Requirements .....	2-1
Cooling .....	2-1
Rack Mounting .....	2-1
User Supplied Components .....	2-3
Initial Setup .....	2-3
Optional TTY Interface Module .....	2-4
Memory Protect Configuration .....	2-4

## CHAPTER 3

### OPERATING INFORMATION

Introduction .....	3-1
Front Panel Switches and Indicators .....	3-1
System 80/30 Monitor Program (iSBC 930) .....	3-1
Monitor Commands .....	3-1
Display Memory Command, D .....	3-2
Program Execute Command, G .....	3-2
Single Step Command, N .....	3-2
Insert Instruction Into Memory, I .....	3-3
Move Memory Command, M .....	3-4

Read Hexadecimal File, R .....	3-4
Substitute Memory Command, S .....	3-4
Write Hexadecimal File, W .....	3-5
Examine and Modify CPU Registers, X .....	3-5
I/O System Routines .....	3-5
Console Input — CI .....	3-6
Console Output — CO .....	3-6
Reader Input — RI .....	3-6
Punch Output — PO .....	3-6
Error Conditions .....	3-6
Invalid Characters .....	3-6
Address Value Errors .....	3-7
Peripheral Device Error .....	3-7
System Initialization (Reset) .....	3-7
Utilizing RAM Storage .....	3-7
Interrupt Processing .....	3-7
Restart Processing .....	3-8
System Programming Considerations .....	3-8

## CHAPTER 4

### PRINCIPLES OF OPERATION

Introduction .....	4-1
Functional Description .....	4-1
iSBC 655 Chassis .....	4-1
Front Panel Switches and Indicators .....	4-1
Line Voltage Select Switch, Fuse .....	4-2
Fans .....	4-3
iSBC 604 Cardcage and Backplane .....	4-3
iSBC 635 Power Supply .....	4-3
Power Fail Status .....	4-3
Output Voltage Adjustments .....	4-3
iSBC 80/30 Single Board Computer .....	4-3

## APPENDIX A

### iSBC 930 MONITOR LISTING



# TABLES

TABLE	TITLE	PAGE
1-1.	Specifications .....	1-2
2-1.	Output Power Available for Expansion Boards .....	2-3
3-1.	Interrupt Jump Table Configuration .....	3-8



# ILLUSTRATIONS

FIGURE	TITLE	PAGE
1-1.	System 80/30 Microcomputer .....	1-1
2-1.	System 80/30 Outline Dimensions .....	2-2
2-2.	Monitor ROM Socket .....	2-4
4-1.	System 80/30 Functional Block Diagram.	4-1
4-2.	System 80/30 Major Assembly Location Diagram .....	4-2





## 1.1 INTRODUCTION

The Intel System 80/30 Microcomputer is a completely packaged standalone computer for OEM applications, consisting of an iSBC 80/30 Single Board Computer, an iSBC 655 Chassis with front panel and fans, an iSBC 604 Cardcage with Backplane, and an iSBC 635 Power Supply (see figure 1-1). This chapter provides a basic system description, a specifications table and a list of necessary interfacing equipment for the System 80/30 Microcomputer.

## 1.2 SYSTEM DESCRIPTION

The Intel System 80/30 Microcomputer is housed in the 3.5 inch, rack-mountable iSBC 655 System Chassis. The chassis has removable front and rear panels, the latter providing board access. Two fans are used for cooling, one directed toward the power supply and the other toward the cardcage.

All power is furnished by an iSBC 635 modular power supply, mounted directly behind the front panel circuit board. All outputs have current limiting and overvoltage protection. An active high (+5 volt) output level is provided when input voltage falls below 90% of its nominal value.

A modular iSBC 604 cardcage/backplane resides in the chassis to house the iSBC 80/30 Single Board Computer and provide an easily accessible bus interface. The cardcage will house three additional expansion boards. The iSBC 80/30 Single Board Computer consists of an Intel 8085A central processing unit, a programmable timer, priority interrupt logic, Multibus control logic, one USART controlled serial I/O port, three parallel I/O ports and 16K bytes of dynamic random access memory. Provision is made for installation of masked or programmable read only memory and an Intel 8041 or 8741 Universal Peripheral Interface.

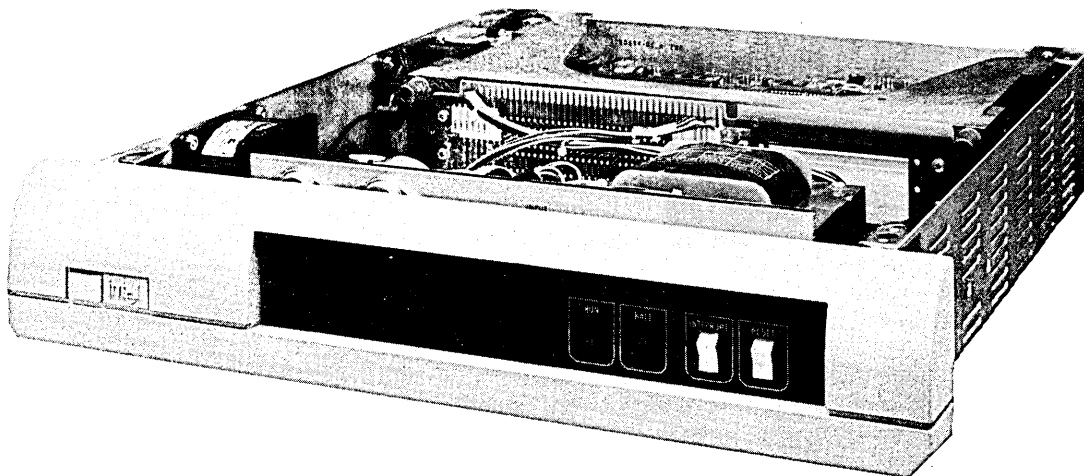


Figure 1-1. System 80/30 Microcomputer

710A-1

### 1.3 MONITOR PROGRAM DESCRIPTION

A comprehensive system monitor (iSBC 930) residing in the first 2K of ROM, is included with the system to facilitate program loading, execution and debugging. Monitor commands include reading and writing hexadecimal paper tapes, executing pre-defined program segments, executing single program instructions, break program execution or any of seven system conditions, (display, move and alter memory contents, display and alter CPU register contents, and read and write memory contents from or to paper tape). Monitor commands and resulting information may be initiated and displayed using a teletypewriter or CRT terminal.

### 1.4 DOCUMENTATION SUPPLIED

The System 80/30 Microcomputer is supplied with the System 80/30 Documentation Package. This package consists of the following publications:

1. *System 80/30 Microcomputer User's Guide*, Order No. 9800710
2. *iSBC 80/30 Single Board Computer Hardware Reference Manual*, Order No. 9800611
3. *iSBC 655 Chassis Hardware Reference Manual*, Order No. 9800709
4. *iSBC 604 Cardcage Hardware Reference Manual*, Order No. 9800708
5. *iSBC 635 Power Supply Hardware Reference Manual*, Order No. 9800298

Each of the hardware reference manuals is complete with logic and/or wiring diagrams and replacement parts listings. To locate a particular diagram or parts

listing, refer to the table of contents at the beginning of each manual.

A complete listing of the Intel 8085 CPU instruction set is given in Appendix A of the *iSBC 80/30 Hardware Reference Manual*.

### 1.5 STANDARD EQUIPMENT AND USER SUPPLIED EQUIPMENT

The System 80/30 Microcomputer is supplied with the following hardware: iSBC 80/30 Single Board Computer chassis, front panel with switches and indicators, power supply, cardcage with backplane, dual fans, a 115 volt power cord, all 115 volt and 230 volt fuses, and a RS-232-C cable (25 inch length). In addition, the 80/30 monitor ROM is supplied (one chip).

All rack mounting hardware is user supplied. Chapter 2 describes procedures for mounting and lists recommended parts. If the chassis is optioned for 230 volt operation, the power cord is not supplied. Any I/O cables, with the exception of the RS232 cable, are also user furnished. The hardware reference manual for each iSBC board will provide a table of compatible I/O connectors recommended by Intel (not all boards require I/O cables, however).

### 1.6 SPECIFICATIONS

Table 1-1 lists the system level specifications for the System 80/30 Microcomputer. Additional data are listed in the Specifications section of each Hardware Reference Manual in the Documentation Package.

Table 1-1. Specifications

<b>POWER REQUIREMENTS</b>	
Frequency:	47-63 Hz
Voltage:	Standard: 115 Vac $\pm$ 10% Optional: 230 Vac $\pm$ 10%
Current:	270 Watts maximum
<b>ENVIRONMENTAL REQUIREMENTS</b>	
Operating Temperature:	0° to 50°C (32° to 122°F)
Relative Humidity:	To 90% non-condensing
<b>PHYSICAL CHARACTERISTICS</b>	
Height:	8.90 cm (3.5 in.)
Width:	At Front Panel: 48.3 cm (19 in.) Behind Front Panel: 43.2 cm (17 in.)
Depth:	50.8 cm (20 in. with all protrusions)
Weight:	16.6 Kg (37 lb)





## 2.1 INTRODUCTION

This chapter provides instructions for unpacking, installation and initial setup of the System 80/30 microcomputer. Peripheral interfacing information is located in Chapter 2 of the *iSBC 80/30 Hardware Reference Manual*. Ideally, the System 80/30 user should be familiar with the entire *iSBC 80/30 Hardware Reference Manual* and this manual before attempting operation.

## 2.2 UNPACKING AND INSPECTION

Inspect the shipping carton immediately upon receipt for evidence of mishandling during transit. If the shipping carton is severely damaged or waterstained, request that the carrier's agent be present when the carton is opened. If the carrier's agent is not present when the carton is opened and the contents of the carton are damaged, keep the carton and packing material for the agent's inspection.

For repairs to a product damaged in shipment contact the Intel Technical Support Center to obtain a Return Authorization Number and further instructions. A purchase order should be submitted to the carrier with your claim. Instruction for contacting the Intel Technical Support Center are given in chapter 5 of the *iSBC 80/30 Hardware Reference Manual*.

It is suggested that salvageable shipping cartons and packing material be saved for future use in the event the product must be reshipped.

## 2.3 INSTALLATION CONSIDERATIONS

The System 80/30 chassis is designed for 19-inch RETMA rack mounting. Figure 2-1 illustrates all relevant outline dimensions. Before chassis installation, the user should be familiar with paragraphs 2-4 through 2-8.

## 2.4 POWER REQUIREMENTS

Maximum AC power requirements for the System 80/30 are listed in the Specifications section (Table 1-1) of Chapter 1.

The total amount of power available from the iSBC 635 Power Supply is listed at the top of Table 2-1. Since the iSBC 80/30 Single Board Computer power requirements vary by configuration, the amount of power available for expansion boards will also vary. Table 2-1 outlines the power available for expansion boards in the System 80/30 Microcomputer chassis. Notice that certain options require only one voltage for operation, and therefore do not affect the output power available from the other voltages. For example, if the iSBC 530 Teletypewriter Adapter is used with the standard iSBC 80/30 Board configuration, the only power supply voltage affected is the -12 volt output. Power consumption for the iSBC 80/30 Board is listed in the *iSBC 80/30 Single Board Computer Hardware Reference Manual*, Order No. 9800611.

## 2.5 COOLING

Without expansion boards, the system dissipates 4.0 kilogram calories of heat. Adequate cooling (<50°C) for the basic system and three expansion boards is provided by the two chassis fans. Care should be exercised during installation, to prevent obstructing chassis air flow openings.

## 2.6 RACK MOUNTING

The System is designed for installation in standard 19" RETMA racks using Chassis-Trak C-300-D-122 Pivot Slides with alternate T-bar handles, or equivalent.



When using slides other than Chassis-Trak C-300-D-122, the maximum slide width is 1.7 inches. Failure to comply may result in damage to the System.

To mount Chassis-Trak slides on the System chassis proceed as follows:

1. Remove and reinstall the front fan using four 1/4 x 7/16 inch spacers under the fan. (Spacers used should be Amatom Electronic Hardware, Part No. 9227-A140 or equivalent.)
2. Mount the slides on the chassis using mounting hardware supplied with the slide.

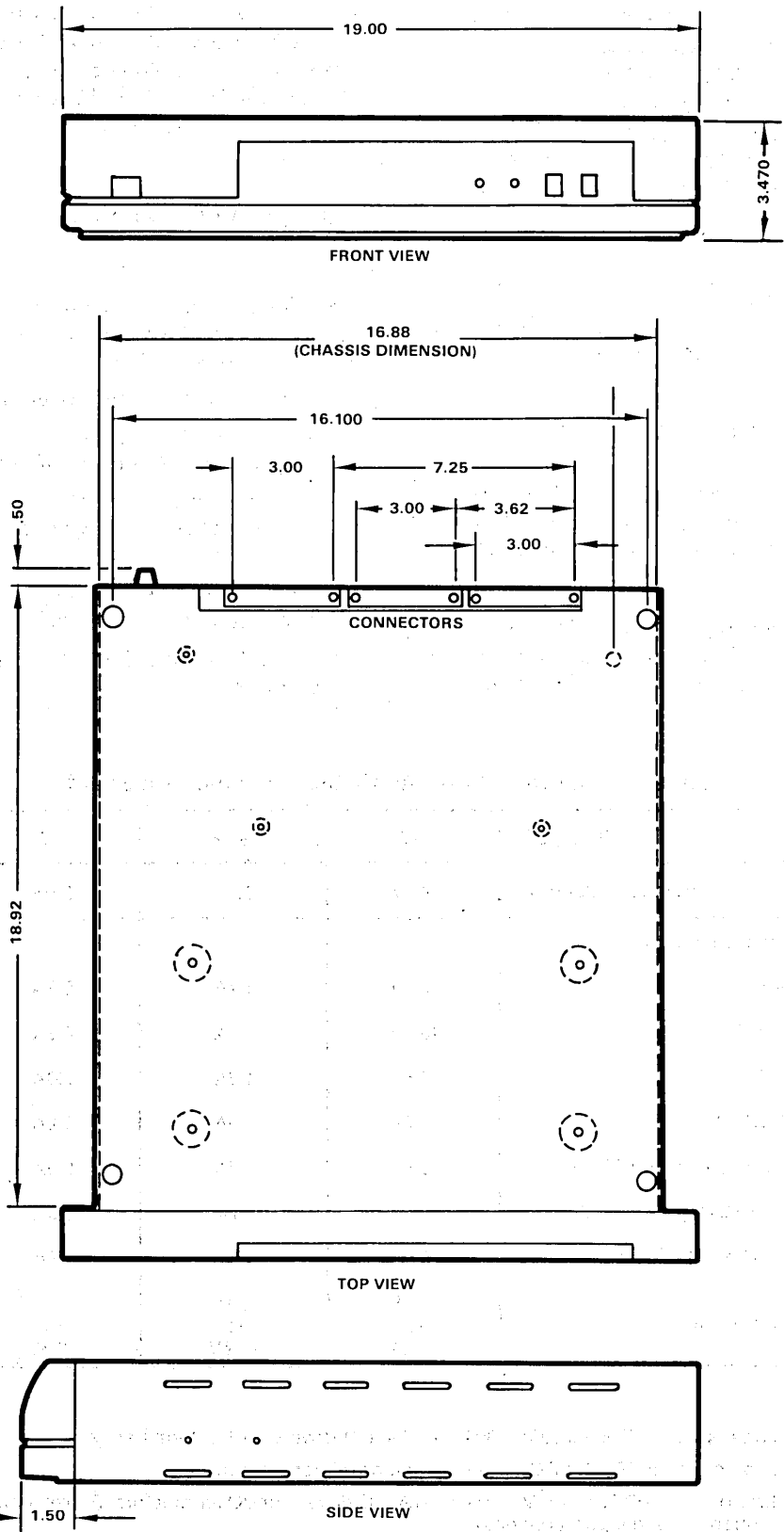


Figure 2-1. System 80/30 Outline Dimensions (Inches)

710A-2

3. When using slides other than Chassis-Track drill holes according to the manufacturer's instructions.



When drilling new holes, ensure all metal filings and chips are removed from the interior of the chassis before turn-on. Failure to comply may result in damage to the system.

4. After the System has been mounted in the cabinet, secure it in the cabinet with two No. 10-32 round head machine screws.

## 2.7 OPTIONAL COMPONENTS

Because the System is designed to satisfy a variety of applications, the user must purchase and install only those components required to satisfy his particular needs.

Instructions for installing optional ROM or an Intel Universal Peripheral Interface circuit are given in the *iSBC 80/30 Hardware Reference Manual*.

The system is shipped with an RS232C Serial I/O Cable Assembly, Part No. 4000677 that mates with most CRT terminals. For other applications, and for parallel I/O interfacing, cabling is user furnished. Refer to the *iSBC 80/30 Hardware Reference Manual* for connector information.

## 2.8 INITIAL SETUP

Once the System Microcomputer has been unpacked and inspected, the following steps should be checked or performed, as indicated:

1. Remove rear and top cover panels.
2. Inspect all internal power connectors to ensure they have not loosened during shipment.
3. Verify setting on input voltage selection switch. Verify that corresponding fuse is installed in fuse holder F1.
4. Remove the two metal circuit board retainers from either side of the cardcage.
5. Unpack the iSBC 80/30. Unpack the iSBC 930 monitor.

Table 2-1. Output Power Available for Expansion Boards

Voltages:	+5	+12	-5	-12
Maximum Current:	14.0A	2.0A	0.9A	0.8A
iSBC 80/30 Board Configuration				
Standard Board <sup>1</sup>	10.5A	1.7A	0.9A	0.7A
With 8041/8741A <sup>2</sup>	10.4A	1.7A	0.9A	0.7A
With iSBC 530 <sup>3</sup>	10.5A	1.6A	0.9A	0.6A
With 2K bytes EPROM (using 8708) <sup>4</sup>	9.6A	1.6A	0.8A	0.7A
With 2K bytes EPROM (using 2758) <sup>4</sup>	9.4A	1.7A	0.9A	0.7A
With 4K bytes EPROM (using 2716) <sup>4</sup>	9.4A	1.7A	0.9A	0.7A
With 8K bytes ROM (using 2332) <sup>4</sup>	9.4A	1.7A	0.9A	0.7A
Over-voltage protection	+5.8 to +6.6V	+14 to +16V	-5.8 to -6.6V	-14 to -16V

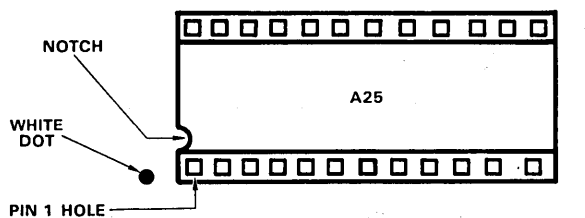
### NOTES:

1. Does not include optional EPROM/ROM, 8041/8741A, I/O drivers, or I/O terminators.
2. Does not include optional EPROM/ROM, I/O drivers, or I/O terminators.
3. Does not include optional EPROM/ROM, 8041/8741A, I/O drivers, or I/O terminators. Power for iSBC 530 is supplied through the serial port connector.
4. Includes two EPROM/ROM chips, 8041/8741A, and 220Ω/330Ω input terminators installed for 34 I/O (all terminator inputs low).

- Carefully install the 930 Monitor ROM into the A25 socket of the iSBC 80/30. Ensure that pin 1 of the ROM (notched side) corresponds to the white dot to the left of the socket (figure 2-2).

### CAUTION

All MOS devices such as the ROM monitor are extremely sensitive to transient voltages, especially static electricity discharges. Caution should be exercised in low humidity environments during device installation, to prevent static discharge. Always ground yourself before handling MOS devices to ensure any static charge which may have accumulated is discharged. After picking up the device, do not walk on carpeted floors; install the device immediately following the grounding.



710A-3

Figure 2-2. Monitor ROM Socket

- Install any other optional components; refer to Chapter 2 of the *iSBC 80/30 Hardware Reference Manual*.
- Perform any jumper modifications, if necessary. Refer to Chapter 2 of the *iSBC 80/30 Hardware Reference Manual*.
  - Install all I/O connectors and strain relief clamps. If interfacing to a teletypewriter, refer to Appendix B in the *iSBC 80/30 Hardware Reference Manual*.
  - Carefully install the computer board into cardcage slot J5 (bottom slot). Component side of the board should be facing up.
  - Install any optional boards in the remaining slots, as described in step 9.

- Replace top and rear panel covers.
- Install power cord into line filter socket on rear panel.

The System 80/30 is now ready for operator interaction, via the CRT terminal or teletypewriter. However, the user should be familiar with the operating procedures and monitor functions described in Chapter 3 before attempting operation.

## 2.9 OPTIONAL TTY INTERFACE MODULE

The optional TTY interface module (iSBC 530 Teletypewriter Adapter) converts iSBC 80/30 RS232-C signal levels to an optically isolated 20 mA current loop interface and provides signal translation for transmitted data, received data and a teletypewriter paper tape reader relay. Installation of the iSBC 530 is discussed in Appendix B of the *iSBC 80/30 Hardware Reference Manual*.

## 2.10 MEMORY PROTECT CONFIGURATION

The memory protect configuration is designed to halt the CPU and preserve RAM contents during a power failure. This is accomplished by the use of a power fail signal and backup batteries. All signals enter the iSBC 80/30 board through the auxiliary P2 connector. To implement the memory protect configuration, the following modifications must be performed:

- Connect auxiliary signal common and returns for +5V, -5V, and +12V backup batteries to P2 pins 1 and 2.
- Connect +5V battery input to P2 pins 3 and 4; -5V battery input to P2 pins 7 and 8 and +12V battery input to P2 pins 11 and 12.
- Remove jumpers W7, W8, and W9 (iSBC 80/30).
- Connect PFS/ input to P2 pin 17, and MEM PROT/ input to P2 pin 20.
- Connect PFI/ input to P2 pin 19; this signal is inverted and supplied to the priority interrupt matrix. To assign the PFI/ input as the highest priority interrupt (8085A TRAP), remove jumper 137-145 and connect jumper 134-137.

The DC characteristics for these auxiliary signals are listed in table 2-18 of the *iSBC 80/30 Hardware Reference Manual*.



## 3.1 INTRODUCTION

This chapter provides operating information for the System 80/30. An operational description of the chassis switches and indicators is given, followed by an introduction to the System 80/30 monitor program (iSBC 930). The bulk of this chapter is devoted to describing the features of the monitor program. Users should be familiar with all information presented in this chapter before attempting operation.

## 3.2 FRONT PANEL SWITCHES AND INDICATORS

The following switches and indicators are mounted on the System 80/30 front panel: a power ON/OFF indicator-switch; a momentary INTERRUPT switch; a momentary RESET switch; a RUN indicator; and a HALT indicator. The function performed by each switch and indicator is described in the following paragraphs.

*Power ON/OFF indicator-switch:* in the latched position (pushbutton toward chassis) AC power is ON and the indicator lamp will illuminate. In the unlatched position, power is OFF. Latching and unlatching is accomplished by pushbutton depression.

*INTERRUPT switch:* activating this momentary switch issues an interrupt request signal to the 8259 Interrupt Controller (A30) on the iSBC 80/30. The switch is wired to pin 42 of the Multibus which corresponds to interrupt request level INT1/. Normally this line would be jumpered to the IR1/ input on the controller. Paragraph 3-25 describes interrupt handling by the monitor.

*RESET switch:* activating this momentary switch causes the system to execute the reset routine at location 0000 in the monitor ROM. Paragraph 3-23 describes this routine.

*RUN indicator:* this indicator is on when the CPU is executing an instruction. The indicator will be off when the CPU is in the WAIT state or a HALT instruction has been executed.

*HALT indicator:* this indicator is on after the CPU has executed a HALT instruction. Only a front panel RESET or an interrupt will remove the HALT state.

Notice that when the CPU is in the WAIT state neither the RUN nor the HALT indicators are on. However, the WAIT state will be terminated by the failsafe timer timeout (approximately 10 ms).

## 3.3 SYSTEM 80/30 MONITOR PROGRAM (iSBC 930)

System operation is facilitated through the use of the monitor program. The monitor is an Intel 8085 CPU program provided in one ROM device, occupying address space 0000 through 07FF. In addition to providing various housekeeping routines such as the power on restart program, the monitor accepts and acts upon user commands to operate the iSBC 80/30 memory and I/O ports. Specifically, the monitor program provides the following facilities:

1. Display selected areas of memory and processor registers.
2. Initiate execution of user programs.
3. Single step instruction execution.
4. Modify contents of memory and processor registers.
5. Insert instruction(s) into memory.
6. Input hexadecimal file from paper tape reader.
7. Output hexadecimal file to paper tape punch.
8. Program BREAK capability.

These facilities are described in sections 3-4 through 3-9. Throughout the discussion of the monitor, bit zero is considered to be the least significant bit. The monitor uses seven bit ASCII without parity (bit 7 = 0). All addresses are stated in hexadecimal notation.

The monitor and the operator communicate using an interactive console such as a CRT terminal. The dialogue consists of operator entered monitor commands and monitor responses, either in the form of a displayed message or an action performed. Following the reset procedure described in section 3-23 the monitor begins the dialogue by transmitting a signon message and requesting a command by displaying a period (prompt character).

## 3.4 MONITOR COMMANDS

Commands are entered in the form of a single upper case alphabetic character followed by a list of numeric or alphabetic parameter. The only command

requiring an alphabetic parameter is the "X" command. The use of alphabetic parameters will be discussed in the section explaining the "X" command. Numeric parameters are entered as hexadecimal numbers. The monitor recognizes the characters 0 through 9 and the upper case alphabetic characters A through F as legal hexadecimal digits. The valid range of numbers is from 1 to 4 hex digits (0-FFFF). If more than four digits are entered only the last four will be used.

The monitor requires each command to be terminated by a carriage return. With the exception of the "S", "N", and "X" commands, the command is not acted upon until the carriage return is sensed. Therefore, the user can abort any command, before he enters the carriage return by typing any illegal character (such as RUBOUT or any alphabetic character with the exception of A through F).

Except where indicated otherwise, a single space is synonymous with the comma for use as a delimiter. Consecutive spaces or commas, or a space or comma immediately following the command letter, will be interpreted as null parameters. Null parameters are illegal in commands except the "G", "S", and "X" command.

Items enclosed in square brackets "[" and "]" are optional. The consequences of including or omitting them are discussed in the text.

In the following paragraphs the monitor command language is discussed. Each command is described, and examples of its use are included for clarity. Error conditions which may be encountered while operating the monitor are described in paragraphs 3-19 through 3-22.

### 3.5 DISPLAY MEMORY COMMAND, D

The format for the D Command is:

D<low address>, <high address>

Selected areas of addressable memory may be accessed and displayed by the D command. The D command produces a formatted listing of the memory area between <low address> and <high address>, inclusive, on the console device. Each line of the listing begins with the address of the first memory location displayed on that line, represented as 4 hexadecimal digits, followed by up to 16 memory locations, each one represented by 2 hexadecimal digits.

The D command may be aborted during execution by typing an Escape (ESC) on the console. The command will be terminated immediately, and a new prompt issued.

*Example:*

```
D9,2A
0009 00 11 22 33 44 55 66
0010 77 88 99 AA BB CC DD EE FF 10 20 30 40 50 60 70
0020 80 90 A0 B0 C0 D0 E0 F0 01 02 03
```

### NOTE

If the <low address> parameter is equal to or greater than the <high address> parameter, only the first location defined by <low address> is printed.

### 3.6 PROGRAM EXECUTE COMMAND, G

The format of the "G" command is:

G[<start address>][<,-<break address>][,-<break address>]

Control of the CPU is transferred from the monitor to the user's program at the specified "start address". If no "start address" is specified, the monitor uses the last value of the PC register. The PC register is saved during execution of any of the following commands or instructions.

- Program BREAK set by the "G" Command.
- "N" Command
- "XP" Command
- RST 1 instruction

The <break address> parameters are 16-bit values that specify breakpoint addresses in the user program. If either is omitted, no corresponding breakpoint is set. If either breakpoint address is encountered while executing the user program, both breakpoints are cleared and control is passed back to the monitor. The current PC and the next 3 instruction bytes pointed at by the PC are displayed.

A breakpoint enables the user to temporarily suspend execution of the user program, examine the state of the program's memory and registers, make modifications if desired, and then continue the program from the point of suspension. When the breakpoint address is reached, the user program is terminated, all pertinent user data is saved, and control is returned to the monitor program. Immediately following a breakpoint, the value of the user program counter points to the memory location in which the breakpoint instruction occurred.

### 3.7 SINGLE STEP COMMAND, N

Single user instructions are executed via the N command. No Carriage return is required when executing this command. After entry of this command, all registers are restored, interrupts

are enabled, and a single instruction within the user program is executed. The instruction to be executed is assumed to be at the address of the last value of the PC register saved. The PC register is saved during execution of any of the following commands or instructions:

- a. Program BREAK set by the "G" command.
- b. "N" command
- c. "XP" command
- d. RST 1 instruction

The single step command is implemented by utilizing interval timer No. 1 connected to interrupt level 7.5. This timer is set immediately prior to exiting the monitor so a single instruction will be executed. The timer expires during execution of the first user instruction and causes an interrupt. The monitor is re-entered, all user registers are saved, the contents of the user registers, current address, and the next three bytes to be executed are displayed; and the user is prompted for a new command.

*Example:*

User Programs	Address	Code
MVI B, 3	4000	06 03
LXI H, 8000	4002	21 00 80

*Monitor Interaction*

```
.XP XXXX-4000
.N
A=XX B=03 C=XX D=XX E=XX F=XX H=XX L=XX
M=XXXX P=4002 S=7F80
4002 NI=21 00 80
```

Due to the use of a timer interrupt to provide the single step capability, care should be taken when combining single step operation with other interrupts, or when modifying the timer operation. The following should be noted:

1. If an interrupt is pending when performing a single step, the interrupt will be serviced (CALL instruction) rather than executing the next user instruction. If the interrupt is serviced by the monitor an interrupt message is displayed. However, if the interrupt is serviced by a user's program, no display or indication of the interrupt is generated.
2. On execution of an RST 1 thru 7 (user breakpoint instruction), the single step capability is suspended, and the user enters the monitor program with only the address of the next instruction displayed.
3. Due to the asynchronous nature of the refresh timing associated with the onboard RAM, the

single step command might sometimes fail to function properly. In this case, the next instruction will not be executed, and the PC and other registers will remain unchanged. When this occurs, the user may reissue the command and try again.

If any difficulty is experienced with the single step command the method of refresh may be altered by moving the jumper wire from pins 110-111 to pins 110-106. This enables the "invisible refresh" feature on the iSBC 80/30 which will cure the problem. In this mode, however, the total board power consumption will be increased from 3% - 5% with the load on the +12v supply being approximately doubled.

4. Due to delays incurred when accessing off-board RAM, the single step command will function properly only when the monitor RAM areas are on-board.

### 3.8 INSERT INSTRUCTION INTO MEMORY, I

The format of the "I" command is:

I <address>

Single or multiple instructions are entered into memory with the I command. After sensing the carriage return (terminating the command line), the monitor waits for the operator to enter a string of hexadecimal digits (0-9, A-F). Each digit in the string is converted into its binary value, and then loaded into memory, beginning at the starting address specified and continuing into sequential locations. Two hexadecimal digits are loaded into each memory location.

Separators between digits (spaces, commas, carriage returns) are ignored; illegal characters will terminate the command. The escape character, ESC (echoed as "\$") terminates the digit string. If an odd number of hex digits have been entered, a 0 will be appended to the string. As each pair of hex digits are entered they are converted to binary and stored in a memory byte. Thus, the data has been entered even if the insertion is terminated with an illegal character.

*Example 1:*

```
.I4E10
112233445566778899$
```

This command puts the following pattern into RAM:

```
4E10 11 22 33 44 55 66 77 88 99
```

*Example 2:*

```
.I4E40
123456789$
```

This command puts the following pattern into RAM:

```
4E40 12 34 56 78 90
```

Note that since an odd number of hexadecimal digits was entered a 0 was appended to the digit string.

### 3.9 MOVE MEMORY COMMAND, M

The format of the "M" command is:

```
M<low address>, <high address>, <destination>
```

The M command moves the contents of memory <low address> through <high address>, inclusive, to the area of RAM beginning at <destination>. The contents of the source field remain undisturbed, unless the receiving field overlaps the source field.

The move operation is performed on a byte-by-byte basis, beginning at <low address>. Care should be taken if <destination> is between <low address> and <high address>. For example, if location 4E10 contains 1A, the command,

```
.M4E10, 4E1F, 4E11
```

will result in locations 4E10 to 4E20 containing "1A1A1A...".

The monitor will continue to move data until the source field is exhausted, or until it reaches address FFFF. If the monitor reaches address FFFF without exhausting the source field, it will move data into this location, then stop.

*Example:*

```
.M4E00, 4E0F, 4F00
```

16 bytes of memory are moved from 4E00-4E0F to 4F00-4F0F by this command.

### NOTE

If the <low address> parameter is greater than the <high address> parameter, only the first destination address is altered.

To fill memory with a constant set the low address to the constant with the "S" command and use low address plus one for the destination address and last address minus one for the high address. The following example will set location 1000 through 10FF to C7.

*Example:*

```
.S1000-44, C7
.M1000, 10FE, 1001
```

### 3.10 READ HEXADECIMAL FILE, R

The R command reads a hexadecimal tape from the paper tape reader and loads the data into the locations specified by the address fields in the hexadecimal records. The paper tape format is described in *MCS 80/85 Absolute Object File Format Technical Specification*, Order No. 9800183. A typical R command will appear as follows:

```
.R (Turn on tape reader before executing this
command.)
```

### 3.11 SUBSTITUTE MEMORY COMMAND, S

The format of the "S" command is:

```
S<address>
```

The S command allows the user to examine and optionally modify memory locations individually. The command functions as follows:

1. Type an S, optionally followed by the hexadecimal address of the first memory location to be examined followed by a space or comma. If no address is specified, the monitor uses the last value of the PC register. The PC register is saved during execution of any of the following commands or instructions:
  1. Program BREAK set by the "G" command
  2. "N" command
  3. "XP" command
  4. RST 1 instruction
2. The contents of the location is displayed, followed by a dash (-).
3. To modify the contents of the location displayed, type in the new data, followed by a space, comma, line feed, or carriage return. If you do not wish to modify the location, type only the space, comma, line feed, or carriage return.
4. If a space or comma is typed in step 3 above, the next memory location will be displayed, followed by a dash (-). If a carriage return is typed, the S command will be terminated. If a line feed is typed, the current address minus 1 will be displayed on a new line, followed by the contents of that location.



**NOTE**

The line feed command will backup the address any number of locations even below the initial address of the S command. This allows the user to check the memory location just modified.

*Example:*

```
.S4D50 AA- BB-CC 01-13 23-24 00- (line feed)
4D53 24- (line feed)
4D52 13- (line feed)
4D51 CC- (line feed)
4D50 AA- (carriage return will terminate command)
```

In this example location 4D50 which contains AA is unchanged, location 4D51 which contained BB now contains CC, location 4D52 which contained 01 now contains 13, and location 4D53 which contained 23 now contains 24. A space was typed displaying location 4D54 then four line feeds were typed to verify the contents of locations 4D53, 4D52, 4D51 and 4D50. A carriage return then terminated the command.

**3.12 WRITE HEXADECIMAL FILE, W**

The format of the "W" command is:

W <low address>, <high address>

The W command transmits portions of memory to a paper tape punch on the teletypewriter. Data is in hexadecimal format. The paper tape format is described in *MCS 80/85 Absolute Object File Format Technical Specification*, Order No. 9800183. A leader tape consisting of 60 null characters is punched followed by the memory data specified by the low/high address parameters. An end of file record is punched automatically to terminate the tape. Following the end of file record, a trailer tape is punched consisting of 60 null characters.

An example of the Write Hexadecimal File operation is as follows:

```
.W4D00,4DAF (User must turn on tape punch
             before executing this command.)
```

This command punches out the contents of memory locations 4D00 through 4DAF.

**3.13 EXAMINE AND MODIFY CPU REGISTERS, X**

The format of the "X" command is:

.X[<register identifier>]

Displaying and modification of the CPU registers is accomplished using the X command. The X command uses <register identifier> to select the particular register to be displayed. A register identifier is a single alphabetic character denoting a register, defined as follows:

A - 8085A CPU register A  
 B - 8085A CPU register B  
 C - 8085A CPU register C  
 D - 8085A CPU register D  
 E - 8085A CPU register E  
 F - 8085A CPU flags byte, displayed in the form as it is stored by the "PUSH PSW" (hex code F5) instruction  
 H - 8085A CPU register H  
 L - 8085A CPU register L  
 M - 8085A CPU registers H and L combined  
 P - 8085A Program Counter  
 S - 8085A Stack Pointer

The command operates as follows:

1. Type an X followed by a register identifier or a carriage return.
2. The contents of the register are displayed (two hexadecimal digits for A, B, C, D, E, F, H, and L; four hexadecimal digits for M, P, and S), followed by a dash (-).
3. The register may be modified at this time by typing the new value, followed by a space, comma, or carriage return. If no modification is desired, type only the space, comma, or carriage return.
4. If a space or comma was typed in step 3, the next register in sequence (alphabetical order) will be displayed as in step 2 (unless S was just displayed in which case the command is terminated). If a carriage return was entered in step 3, the X command is terminated.
5. If a carriage return was typed in step 1 above, an annotated list of all registers and their contents is displayed.

**3.14 I/O SYSTEM ROUTINES**

The Monitor provides four I/O system routines (Device Drivers). The four routines include console character in and console character out, which the user may call upon to read and write characters from and to the console device. The other two routines allow the user to read and punch paper tapes from the teletypewriter.

The drivers interface through the Universal Synchronous/Asynchronous Receiver/Transmitter (USART). The monitor configures the USART during a power-on or reset condition to the following state:

**Mode:**

1 Stop bit at 150 thru 9600 baud  
 2 stop bits at 110 baud  
 Parity disabled  
 8 bit character length  
 Baud rate factor of 64X

**Command:**

No hunt mode  
 Request-To-Send high  
 Receiver enabled  
 Data-Terminal-Ready high at 150 through 9600 baud  
 Data-Terminal-Ready low at 110 baud (high during read)  
 Transmitter enabled

**NOTE**

Care should be exercised if modifying the USART mode and command, since the monitor depends on the configuration defined above for driver operation.

The four monitor I/O system routines may be accessed by calling the desired routine. The following paragraphs describe the routines available and their respective functions.

**3.15 CONSOLE INPUT—CI**

This routine returns an 8 bit character received from the console device to the caller in the A-register. The A-register and the CPU condition codes are affected by this operation. The entry point of this routine is 040.

*Example:*

```
CI EQU 040
...
CALL CI
STA DATA
...
```

**3.16 CONSOLE OUTPUT—CO**

This routine transmits an 8 bit character, passed from the caller in the C- register, to the console device. The A and C registers, and the CPU condition codes, are affected by this operation. The entry point of this routine is 043.

*Example:*

```
CO EQU 043
...
MVI C, .
CALL CO
```

**3.17 READER INPUT — RI**

RI returns an 8 bit character read from the teletypewriter reader in the A-register. If no character was read from the device or the End Of File was read, the CARRY condition code is set equal to 1, and the A-register is zeroed. If data is ready, the CARRY bit is zeroed. If a character is not received from the teletypewriter reader within 250 milliseconds, an End Of File is simulated and control returned to the calling program. The entry point of this routine is 046.

*Example:*

```
RI EQU 046
...
CALL RI
JC EOF ; END OF FILE SENSED
STA DATA
...
```

**3.18 PUNCH OUTPUT—PO**

PO transmits an 8 bit character from the calling program to the teletypewriter. PO is identical in format to CO, the only difference being the entry point address, 049.

*Example:*

```
PO EQU 049
...
LHLD DATADR ;DATADR-16 BIT ADDRESS
                OF DATA BYTE
MOV C,M ;TO BE PUNCHED
CALL PO
```

**3.19 ERROR CONDITIONS**

The system monitor defaults on three types of errors; Invalid Character, Address Value, and Peripheral errors. The following paragraphs detail the operation for each of these conditions.

**3.20 INVALID CHARACTERS**

The monitor checks the validity of each character as it is entered from the console. As soon as the monitor determines that the last character entered is illegal in

its context, the monitor aborts the command and issues a “#” to indicate the error.

*Examples:*

D1400,145G#

The character G was encountered in a parameter list where only hexadecimal digits and delimiters are valid.

### 3.21 ADDRESS VALUE ERRORS

Some commands require an address pair of the form <low address> <high address>. If, on these commands, the value of <low address> is greater than or equal to the value of <high address>, the action indicated by the command will be performed on the data at <low address> only.

The valid range of addresses is 0-FFFF. Thus, if a hexadecimal address greater than FFFF is entered, only the last 4 hex digits will be used.

Another type of address error may occur when the operator specifies in a command a part of memory which does not exist in his particular configuration. In general, if a nonexistent portion of memory is specified as the source field for an instruction, the data fetched will be unpredictable. If a nonexistent portion of memory is given as the destination field in a command, the command has no effect.

### 3.22 PERIPHERAL DEVICE ERROR

Peripheral devices selected by the operator which are not ready or are non-existent will cause undefined execution of the Monitor (e.g., an indefinite wait for READY status in an I/O loop). This situation may be rectified by readying the device and by re-initializing the system (refer to paragraph 3-23).

### 3.23 SYSTEM INITIALIZATION (RESET)

Following an initial power on or reset operation, the monitor program begins executing at location 0000. The following initialization and functions are performed:

1. Set timer 1 of the 8253 Interval Timer to MODE 2, so it may be used for the single step function.
2. Set the 8259 Programmable Interrupt Controller to vector into the 48 byte jump table starting at

RAM location 7FD0. Set the fixed priority mode with TRAP at the highest priority; followed by 7.5, 6.5, 5.5 and 0-7. All interrupts are set unmasked.

3. Initialize the user's stack pointer to 7F80.
4. Set timer 2 of the 8259 Interval Timer to MODE 3 then automatically determine the console terminal baud rate. The timer 2 output is used as clock for the 8251 Universal Synchronous/Asynchronous Receiver/Transmitter (USART). The USART clock is initially set at 9600 baud. Two “U” characters are used to check for baud rate. When the first “U” character is entered it is checked for 9600, 4800, 2400, and 1200 baud rate. If a match is found then that baud rate is set into the clock. If not, then a second “U” character must be entered. The second “U” character is checked for 600, 300, 150, and 110 baud. When the baud rate has been successfully determined, the sign-on message “80/30 MONITOR” will be displayed on the console. When the monitor is ready for a command, it will prompt with a period “.”.

### NOTE

After checking the first “U” the monitor will wait 3 seconds for the second “U” to be typed, if one is necessary. If the 3 second interval expires before the second “U” is typed, begin the baud rate search procedure again.

### 3.24 UTILIZING RAM STORAGE

RAM storage locations 7F80 to 7FFF are reserved for the monitor stack, register save area, and the interrupt jump table. The user's stack pointer is automatically set to 7F80 during the power on routine. RAM locations 4000 to 7F80 are available for the user's program.

### 3.25 INTERRUPT PROCESSING

All interrupts are serviced by a jump table stored in RAM at locations 7FD0 through 7FFF. By modifying the addresses in the jump table, the user can cause execution of his own interrupt service routine, stored in RAM. The interrupt jump table configuration is shown in table 3-1.

Table 3-1. Interrupt Jump Table Configuration

Hexadecimal Address	Interrupt
7FD0	TRAP
7FD4	7.5
7FD8	6.5
7FDC	5.5
7FE0	0
7FE4	1
7FE8	2
7FEC	3
7FF0	4
7FF4	5
7FF8	6
7FFC	7

Interrupts are disabled during user-monitor command interaction. Pending interrupts will not interfere with program verification. Interrupts are enabled on exiting the monitor to run a program via the "G" or "N" command.

### 3.26 RESTART PROCESSING

Entering the monitor during program execution is accomplished either by setting a breakpoint (using the "G" command) or placing an RST 1 instruction code in the program. Entering with the "G" command causes the current address and the next three bytes to be displayed. The monitor will save the state of the CPU: all registers, flags, user's program counter, and user's stack pointer. The value of these may be examined with the "X" command. Subsequently entering a "G" command will restore these values.

When an RST instruction is encountered in the user program, the monitor interrupts the user program, saves all registers and the value in the program counter. The "G" command can not be used to advance the program following an RST interrupt.

When an RST1 instruction causes an interrupt, the program counter does not advance beyond the breakpoint. Therefore it is not possible to correct the program. By using the "S" command the instruction

may be altered. If the RST1 instruction is to be kept, the "N" command may be used to step over it.

Conversly, restart instructions RST2 through RST7 will clear out any breakpoints established by the "G" command.

### 3.27 SYSTEM PROGRAMMING CONSIDERATIONS

Program development for the System 80/30 Microcomputer may be accomplished most efficiently with the aid of a development system such as the Intellec Microcomputer Development System. The Intellec's various configurations permit program development in either 8085 assembly language or the more advanced PL/M 80 and/or Fortran 80 languages. A concise introduction to the Intellec development systems is provided in the publication, *A Guide To Intellec Microcomputer Development Systems*, by Daniel McCracken. This publication is available from the Intel Literature Department and is identified by Order Number 9800558B.

Intellec control software includes a ROM based program monitor which supervises the development system CPU. Diskette equipped models include the Intel System Implementation Supervisor (ISIS) programs. This is a broad collection of development programs, including a text editor, 8085 assembler, a Library Manager and other aids.

PL/M is a high level language that is particularly well suited for use in system programming. With PL/M, programs may be created, compiled, modified, linked, relocated and debugged entirely on the Intellec system.

Fortran is a high level language that is particularly well suited to application programs. Intel's Fortran 80 compiler implements the ANSI Fortran 77 standard. In addition to the features of PL/M, Fortran has arithmetic processing capability and a variety of facilities for handling formatted input and output.



# CHAPTER 4 PRINCIPLES OF OPERATION

## 4.1 INTRODUCTION

This chapter briefly describes the fundamental operation of the System 80/30 Microcomputer. The following paragraphs describe, on a general systems level, basic operation. A more detailed description of the iSBC 80/30 Single Board Computer and circuit analysis is provided in Chapter 4 of the *iSBC 80/30 Hardware Reference Manual*.

## 4.2 FUNCTIONAL DESCRIPTION

The System 80/30 Microcomputer consists of four main components: chassis, power supply, cardcage and computer board (refer to figures 4-1 and 4-2). The following paragraphs give a summarized functional description of the System 80/30.

## 4.3 iSBC 655 CHASSIS

The iSBC 655 Chassis components include the front panel with switches and indicators, two cooling fans, and all chassis assembly sheet metal and structural hardware.

## 4.4 FRONT PANEL SWITCHES AND INDICATORS.

Three switches are associated with the front panel: the power ON/OFF indicator—switch; the RUN switch; and the HALT switch.

The power ON/OFF indicator switch (S1) is located on the left side of the panel. When power is applied, the indicator switch will illuminate. The illuminator bulb is accessed by pulling off the translucent switch cap.

The other two front panel switches are momentary rocker types, labeled RESET (S3) and INTERRUPT (S4). The RESET switch is wired to pin 14 of the backplane. When depressed, the switch generates the signal RESET/ which is synonymous with INIT/ on the backplane. The flip-flop which actually generates RESET/ is located on the front panel P.C.B. and is shown in figure 4-5 of the *iSBC 655 Hardware Reference Manual*.

The INTERRUPT switch functions in a similar manner. When the switch is depressed, circuitry on the front panel P.C.B. generates the signal INT1/ which

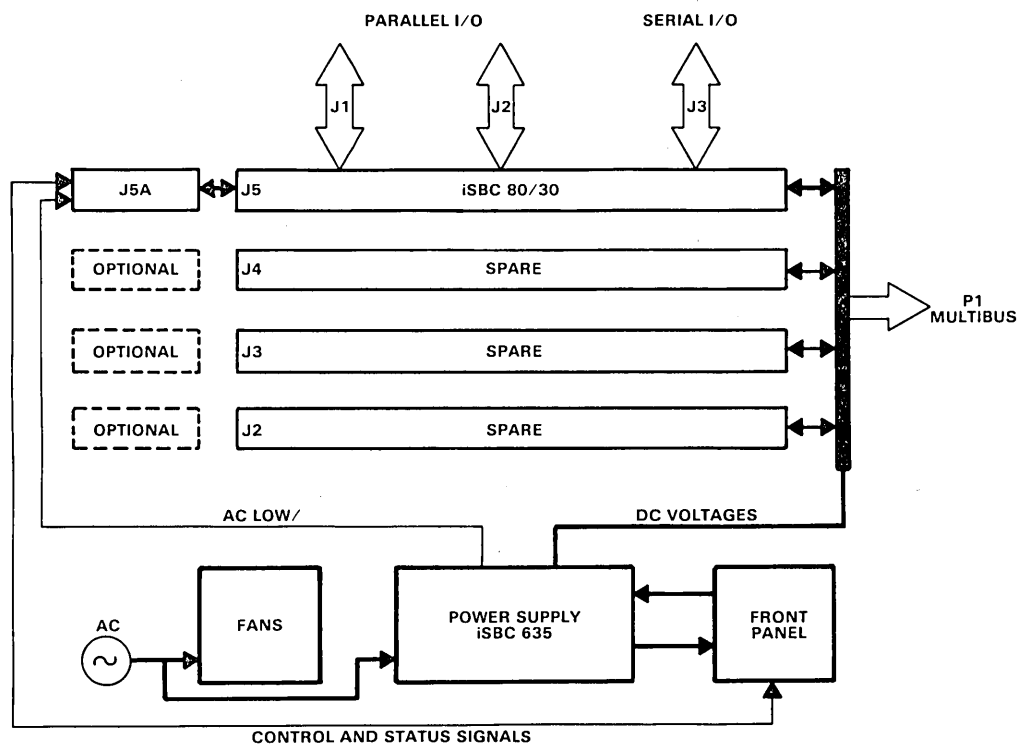


Figure 4-1. System 80/30 Functional Block Diagram

710A-4

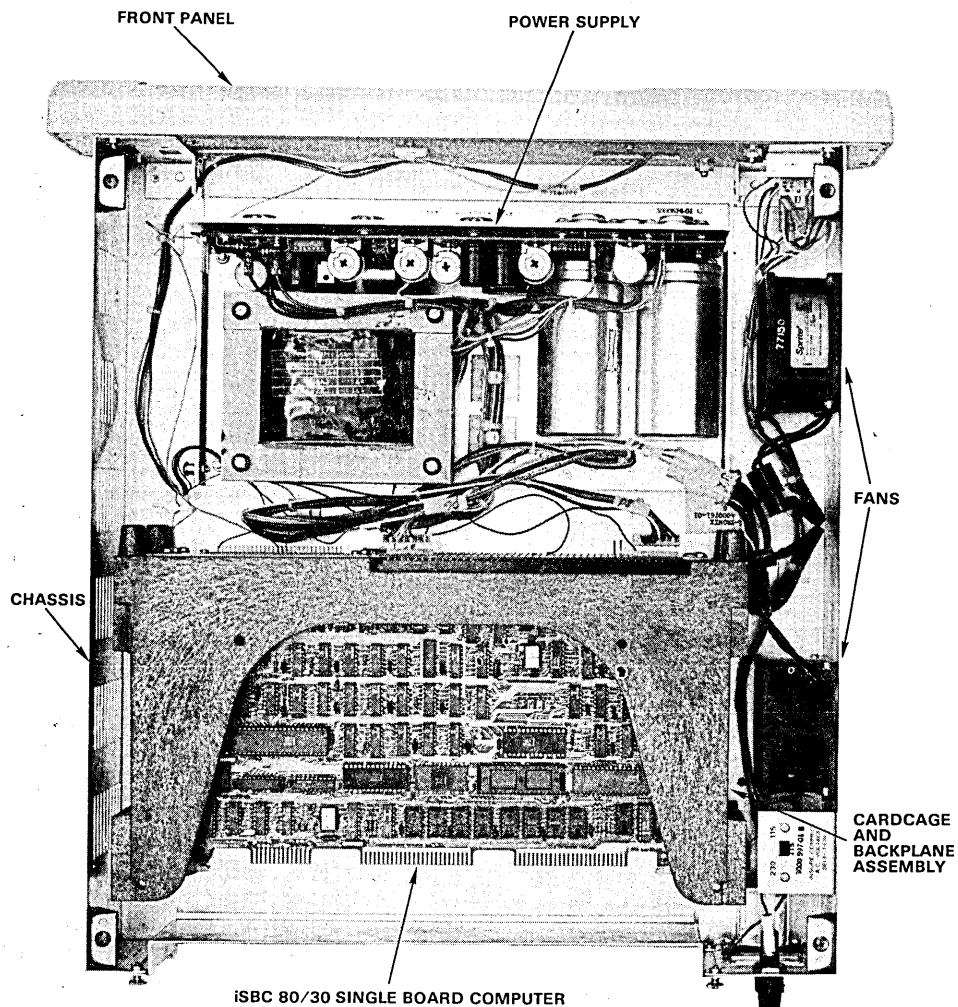


Figure 4-2. System 80/30 Major Assembly Location Diagram

is wired to pin 42 on the backplane. Refer to paragraph 3-34 of the *iSBC 80/30 Hardware Reference Manual* for a complete description of system interrupts.

The RUN and HALT indicators are actually light emitting diodes mounted on the front panel printed circuit board. The RUN and HALT indicators illuminate as a function of three iSBC 80/30 status signals: WAIT/, HALT/, and ALE. The RUN indicator will illuminate when ALE is true; and when either WAIT/ or HALT/ is false. Conversely, the HALT indicator illuminates when WAIT/ or HALT/ is true. This circuitry is shown in figure 4-5 of the *iSBC 655 Hardware Reference Manual*.

**4.5 LINE VOLTAGE SELECT SWITCH, FUSE.** The line voltage select switch is located adjacent to the cardcage fan; and is shown schematically in figure 4-2 of the *iSBC 655 Hardware Reference Manual*. The switch has two positions, corresponding to the two usable line voltages; 115 Vac and 230 Vac. A keyed switch locking plate secures the switch in one position. The switch can be set to the other position, only by loosening the two plate hold-down screws, and flipping the plate over. Each side of the plate is labeled.

Fuse F1 is located on the rear chassis panel, right side. A 2.5 ampere fuse should be used for 230 volt operation and a 5 ampere fuse is used for 115 volt operation.

The line filter is located directly below the fuse. The line filter hardware also functions as the power cord connector.

**4.6 FANS.** The chassis utilizes two fans for cooling purposes. Both are located on the power ON/OFF switch side of the chassis. Air flow is directed into the chassis, with one fan cooling the power supply and the other cooling the cardcage. Power for each fan is derived directly from the line voltage.

#### 4.7 iSBC 604 CARD CAGE AND BACKPLANE

The cardcage houses a total of four iSBC boards, including the iSBC 80/30. Considered part of the cardcage, the backplane is actually a printed circuit board with Multibus and other connectors attached. Operating voltages reach the boards via the backplane and all interboard communication occurs on the Multibus. The backplane's Multibus edge connector allows additional external cardcages to be attached.

Signal terminator resistors are located on the backplane P.C.B., and are shown schematically in figure 4-1 of the *iSBC 604/614 Hardware Reference Manual*.

An additional connector, J5A, is installed on the backplane to accommodate several iSBC 80/30 status signals and auxiliary RAM refresh power.

#### 4.8 iSBC 635 POWER SUPPLY

This power supply provides regulated DC voltage (+12, -12, +5, & -5) from 100, 115, 215 or 230 Vac power sources. Output levels are delivered through keyed connectors which mate directly to the front

panel and backplane. All outputs have current limiting and overvoltage protection. These tolerances are listed in the Specifications section of the *iSBC 635 Hardware Reference Manual*.

**4.9 POWER FAIL STATUS.** The power supply is equipped with an AC line monitor which will generate an AC low signal, PFI/, when the source falls below 90% of its nominal value. The signal PFI/ is wired to pin 19 of J5A on the backplane. This line is connected to the interrupt jumper matrix on the iSBC 80/30.

**4.10 OUTPUT VOLTAGE ADJUSTMENTS.** Each output voltage level is individually adjustable. Procedures for these adjustments are given in Chapter 3 of the *iSBC 635 Hardware Reference Manual*.

#### 4.11 iSBC 80/30 SINGLE BOARD COMPUTER

At the heart of the System 80/30 is the iSBC 80/30 Single Board Computer. The iSBC 80/30 includes an Intel 8085A CPU, 16K bytes of dynamic RAM, one serial and three parallel I/O ports, a programmable timer, priority interrupt logic, and Multibus control logic.

Two DIP sockets are provided to accommodate up to 8K bytes of ROM or EPROM, using 1K, 2K or 4K chips. An additional socket is provided for an Intel 8041/8741 Universal Peripheral Interface (UPI).

Chapters 3 and 4 of the *iSBC 80/30 Hardware Reference Manual* provide a comprehensive detailed description of board operation and programming.



**APPENDIX A**  
**iSBC 930 MONITOR LISTING**



LOC	OBJ	SEQ	SOURCE STATEMENT
		1	\$MOD85
		2	;
		3	;
		4	*****
		5	*****
		6	;
		7	80/30 MONITOR
		8	(MON830)
		9	VERSION 1.2
		10	24 APRIL 1978
		11	;
		12	*****
		13	*****
=		14	\$INCLUDE(:F1:CPYRTA.NOT)
=		15	;
=		16	;
=		17	(C) 1978 INTEL CORPORATION. ALL RIGHTS RESERVED. NO PART OF THIS
=		18	PROGRAM OR PUBLICATION MAY BE REPRODUCED, TRANSMITTED, TRANSCRIBED,
=		19	STORED IN A RETRIEVAL SYSTEM, OR TRANSLATED INTO ANY LANGUAGE OR
=		20	COMPUTER LANGUAGE, IN ANY FORM OR BY ANY MEANS, ELECTRONIC,
=		21	MECHANICAL, MAGNETIC, OPTICAL, CHEMICAL, MANUAL OR OTHERWISE,
=		22	WITHOUT THE PRIOR WRITTEN PERMISSION OF INTEL CORPORATION,
=		23	3065 BOWERS AVENUE, SANTA CLARA, CALIFORNIA 95051.
=		24	*****
=		25	*****
		26	;
		27	;
		28	ABSTRACT
		29	=====
		30	;
		31	THIS PROGRAM RUNS ON THE SBC 80/30 BOARD AND IS DESIGNED TO PROVIDE
		32	THE USER WITH A MINIMAL MONITOR. BY USING THIS PROGRAM,
		33	THE USER CAN EXAMINE AND CHANGE MEMORY OR CPU REGISTERS, LOAD
		34	A PROGRAM (IN ABSOLUTE HEX) INTO RAM, AND EXECUTE INSTRUCTIONS
		35	ALREADY IN MEMORY. THE MONITOR ALSO PROVIDES THE USER WITH
		36	ROUTINES FOR PERFORMING CONSOLE I/O AND PAPER TAPE I/O.
		37	;
		38	;
		39	PROGRAM ORGANIZATION
		40	=====
		41	;
		42	THE LISTING IS ORGANIZED IN THE FOLLOWING WAY. THE FIRST ROUTINE
		43	IS THE COMMAND RECOGNIZER, WHICH IS THE HIGHEST LEVEL
		44	ROUTINE IN THE PROGRAM. NEXT, ARE THE ROUTINES TO IMPLEMENT
		45	THE VARIOUS COMMANDS, FOLLOWED BY THE INTERRUPT HANDLERS,
		46	AND FINALLY THE UTILITY ROUTINES WHICH ACTUALLY DO THE DIRTY WORK.
		47	;
		48	WITHIN EACH SECTION, THE ROUTINES ARE ORGANIZED IN ALPHABETICAL
		49	ORDER, BY ENTRY POINT OF THE ROUTINE.
		50	;
		51	THE 80/30 MONITOR CAN RESIDE IN ONE 2716 PROM.
		52	;

LOC	OBJ	SEQ	SOURCE STATEMENT
		53	; THE PROGRAM ALSO EXPECTS THAT RAM LOCATIONS 7F80H TO 7FE0H,
		54	; INCLUSIVE, ARE RESERVED FOR THE PROGRAM'S OWN USE. THESE
		55	; LOCATIONS MAY BE ALTERED, HOWEVER, BY CHANGING THE EQU'ED
		56	; SYMBOL "DATA" AS DESIRED.
		57	;
		58	;
		59	;
		60	; *****
		61	;
		62	;
		63	MONITOR EQUATES
		64	;
		65	;
		66	; *****
		67	;
		68	;
00AF		69	B110 EQU 00AFH ; COUNT FOR 110 BAUD TIMER
0080		70	B150 EQU 0080H ; COUNT FOR 150 BAUD TIMER
0040		71	B300 EQU 0040H ; COUNT FOR 300 BAUD TIMER
0020		72	B600 EQU 0020H ; COUNT FOR 600 BAUD TIMER
0010		73	B1200 EQU 0010H ; COUNT FOR 1200 BAUD TIMER
0008		74	B2400 EQU 0008H ; COUNT FOR 2400 BAUD TIMER
0004		75	B4800 EQU 0004H ; COUNT FOR 4800 BAUD TIMER
0002		76	B9600 EQU 0002H ; COUNT FOR 9600 BAUD TIMER
0078		77	CH15 EQU 078H ; RECOGNITION CHAR FOR 150 BAUD
0066		78	CH30 EQU 066H ; RECOGNITION CHAR FOR 300 BAUD
0055		79	CH60 EQU 055H ; RECOGNITION CHAR FOR 600 BAUD
0080		80	CH12 EQU 080H ; RECOGNITION CHAR FOR 1200 BAUD
0078		81	CH24 EQU 078H ; RECOGNITION CHAR FOR 2400 BAUD
0066		82	CH48 EQU 066H ; RECOGNITION CHAR FOR 4800 BAUD
0055		83	CH96 EQU 055H ; RECOGNITION CHAR FOR 9600 BAUD
0027		84	CMD EQU 027H ; INITIALIZATION
00ED		85	CNCTL EQU 0EDH ; CONSOLE USART CONTROL PORT
00EC		86	CNIN EQU 0ECH ; CONSOLE INPUT PORT
00EC		87	CNOUT EQU 0ECH ; CONSOLE OUTPUT PORT
00ED		88	CONST EQU 0EDH ; CONSOLE STATUS INPUT PORT
000D		89	CR EQU 00DH ; CODE FOR CARRIAGE RETURN
00DD		90	CTR1 EQU 0DDH ; COUNTER #1
00DE		91	CTR2 EQU 0DEH ; COUNTER #2
0070		92	C1M0 EQU 070H ;
00B6		93	C2M3 EQU 0B6H ;
8000		94	DATA EQU 2*16384 ; END OF MONITOR 16K RAM USAGE
7FB7		95	REGS EQU DATA-73 ; START OF REGISTER SAVE AREA
001B		96	ESC EQU 01BH ; CODE FOR ESCAPE CHARACTER
0020		97	EOIC EQU 020H ; END OF INT CMD WORD
000F		98	HCHAR EQU 00FH ; MASK TO SELECT LOWER HEX CHAR FROM BYTE
007F		99	HREGS EQU HIGH REGS ; HIGH BYTE OF ADDRESS
00C3		100	JMCMD EQU 0C3H ; JUMP COMMAND FOR RAM TABLE
00DA		101	ICCP EQU 0DAH ; INT CONTROLLER COM PORT
00F6		102	ICW1 EQU 0F6H ; INT CMD WORD 1
007F		103	ICW2 EQU HREGS ; INT CMD WORD 2
0000		104	IMASK EQU 0H ; INT MASK VALUE
00FF		105	INVRT EQU 0FFH ; MASK TO INVERT HALF BYTE FLAG
000A		106	LF EQU 00AH ; CODE FOR LINE FEED
0040		107	LLOW EQU 040H ; LOWEST BYTE ADDRESS FOR SINGLE STEP

LOC	OBJ	SEQ	SOURCE STATEMENT
000F		108	LNIB EQU 00FH ; LOWER 4 BIT NIBBLE OF BYTE
004F		109	MODE EQU 04FH ; MODE SET FOR USART
00CF		110	MODE2 EQU 0CFH ; TWO STOP BITS PLEASE.
00DB		111	MSKPT EQU 0DBH ; INT CONTROLLER CMD PORT
000F		112	NEWLN EQU 00FH ; MASK FOR CHECKING MEMORY ADDR DISPLAY
000F		113	NEXCT EQU 15 ; NEXT TIMER COUNT FOR CLOCK 1.
000B		114	OCW3 EQU 0BH ; INT OPERATION CMD WORD 3
008B		115	ONEMS EQU 139 ; 1 MILLISECOND CONSTANT
007F		116	PRTY0 EQU 007FH ; MASK TO CLEAR PARITY BIT FROM CONSOLE CHAR
0002		117	RBR EQU 002H ; RECEIVER BUFFER STATUS READY
0037		118	RESURT EQU 037H ; RESET ERROR AND SET DTR.
0040		119	RSTUST EQU 040H ; USART MODE RESET COMMAND
00CF		120	RST1 EQU 0CFH ; RESTART 1 INSTRUCTION
0054		121	STM1 EQU 054H ; MODE 2 COUNTER 1
001B		122	TERM EQU 01BH ; CODE FOR ICMD TERMINATING CHARACTER (ESCAPE)
00DF		123	TMCP EQU 0DFH ; COMMAND FOR INTERVAL TIMER
000F		124	TMDIS EQU 0FH ; DISABLE ALL INTERRUPTS MASK
000B		125	TMENB EQU 0BH ; ENABLE 7.5 INTERRUPTS
0010		126	TMRST EQU 010H ; RESET 7.5 INTERRUPT
0001		127	TRDY EQU 01H ; MASK TO TEST TRANSMITTER STATUS
0037		128	TTYADV EQU 037H ; TTY READER ADVANCE COMMAND
0035		129	TTYSTP EQU 035H ; TTY READER STOP COMMAND
0004		130	TXBE EQU 04H ; CHECK FOR TRANSMITTER BUFFER EMPTY
00F0		131	UNIB EQU 0F0H ; UPPER 4 BIT NIBBLE OF BYTE
00FF		132	UPPER EQU 00FFH ; DENOTES UPPER HALF OF BYTE IN ICMD
7F80		133	USAREA EQU DATA-128 ; START OF USER STACK AREA
		134	;
		135	;
		136	;
		137	*****
		138	;
0000		139	ORG 00H
		140	;
		141	;
		142	LOK:
0000 F3		143	DI ; BETTER FILLER
0001 3E4F		144	MVI A,MODE ; USART SET UP MODE.
0003 D3ED		145	OUT CNCTL ; OUTPUT MODE
0005 C35F00		146	JMP INUST ; BRANCH TO COMPLETE USART INITIALIZATION
		147	;
		148	;
		149	*****
		150	;
		151	;
		152	RESTART ENTRY POINT
		153	;
		154	;
		155	*****
		156	;
		157	;
		158	GO:
0008 F3		159	DI ; DISABLE INTERRUPTS ON MONITOR ENTRANCE
0009 CD9106		160	CALL REGSV ; SAVE ALL USER REGISTERS
000C C32F04		161	JMP GOBK1 ; HAVE WE A BREAK ENTRY?
000F 00		162	NOF ; FILLER

LOC	OBJ	SEQ	SOURCE STATEMENT
		163 ;	
		164 ;	
0010		165	ORG 010H ;
		166 ;	
		167 ;	
0010	F3	168	DI ; STOP INTERRUPTS ON MONITOR ENTRANCE
0011	C3CC7F	169	JMP OTHER ; RST 2
0014	00	170	NOP ; FILLER
0015	00	171	NOP ; FILLER
0016	00	172	NOP ; FILLER
0017	00	173	NOP ; FILLER
		174 ;	
0018		175	ORG 018H ;
		176 ;	
0018	F3	177	DI ; STOP INTERRUPTS ON MONITOR ENTRANCE
0019	C3CC7F	178	JMP OTHER ; RST 3
001C	00	179	NOP ; FILLER
001D	00	180	NOP ; FILLER
001E	00	181	NOP ; FILLER
001F	00	182	NOP ; FILLER
		183 ;	
0020		184	ORG 020H ;
		185 ;	
0020	F3	186	DI ; STOP INTERRUPTS ON MONITOR ENTRANCE
0021	C3CC7F	187	JMP OTHER ; RST 4
		188 ;	
0024		189	ORG 024H ;
		190 ;	
0024	F3	191	DI ; STOP INTERRUPTS ON MONITOR ENTRANCE
0025	C3D07F	192	JMP TRAP ; TRAP INTERRUPT (4.5)
		193 ;	
0028		194	ORG 028H ;
		195 ;	
0028	F3	196	DI ; STOP INTERRUPTS ON MONITOR ENTRANCE
0029	C3CC7F	197	JMP OTHER ; RST 5
		198 ;	
002C		199	ORG 02CH ;
		200 ;	
002C	F3	201	DI ; STOP INTERRUPTS ON MONITOR ENTRANCE
002D	C3DC7F	202	JMP USIN1 ; OTHER 5.5 INTERRUPT
		203 ;	
0030		204	ORG 030H ;
		205 ;	
0030	F3	206	DI ; STOP INTERRUPTS ON MONITOR ENTRANCE
0031	C3CC7F	207	JMP OTHER ; RST 6
		208 ;	
0034		209	ORG 034H ;
		210 ;	
0034	F3	211	DI ; STOP INTERRUPTS ON MONITOR ENTRANCE
0035	C3D87F	212	JMP USIN2 ; RST 6.5 INTERRUPT
		213 ;	
0038		214	ORG 038H ;
		215 ;	
0038	F3	216	DI ; STOP INTERRUPTS ON MONITOR ENTRANCE
0039	C3CC7F	217	JMP OTHER ; RST 7

LOC	OBJ	SEQ	SOURCE STATEMENT
		218 ;	
003C		219	ORG 03CH ;
		220 ;	
003C F3		221	DI ; STOP INTERRUPTS ON MONITOR ENTRANCE
003D C3D47F		222	JMP USINT ; RST 7.5 INTERRUPT
		223 ;	
		224 ;	BRANCH TABLE FOR USER ACCESSIBLE ROUTINES
		225 ;	
0040		226	ORG 040H
		227	USECI:
0040 C3D804		228	JMP CI ; CONSOLE IN
		229	USECO:
0043 C3EB04		230	JMP CO ; CONSOLE OUT
		231	USERI:
0046 C3C106		232	JMP RI ; READER IN
		233	USEPO:
0049 C3EB04		234	JMP PO ; PUNCH OUT
		235 ;	
		236 ;	
		237	CPYRT:
004C 28432920		238	DB '(C) 1978 INTEL CORP'
0050 31393738			
0054 20494E54			
0058 454C2043			
005C 4F5250			
		239 ;	
		240 ;	
		241 ;	*****
		242 ;	
		243 ;	
		244 ;	
		245 ;	DESCRIPTION: INUST OUTPUTS TO THE USART THE COMMAND WORD
		246 ;	LOOKS FOR THE LETTER 'U' TO DETERMINE THE BAUD RATE OF
		247 ;	THE USERS CONSOLE. INITIALIZES THE STACK POINTER,
		248 ;	THE INTERVAL TIMER, AND THE INTERRUPT CONTROLLER.
		249 ;	
		250 ;	
		251	INUST:
005F 3E27		252	MVI A,CMD ;
0061 D3ED		253	OUT CNCTL ; OUTPUT COMMAND WORD TO USART
0063 21777F		254	LXI H,MSTAK-64 ; LOAD POINTER TO STACK
0066 22C17F		255	SHLD SSAVE ; INITIALIZE USER STACK POINTER
0069 31B77F		256	LXI SP,MSTAK ; INITIALIZE MONITOR STACK
006C 3E70		257	MVI A,C1M0 ; INITIALIZE SINGLE STEP TIMER MODE
006E D3DF		258	OUT TMCP ;
0070 3EB6		259	MVI A,C2M3 ; INITIALIZE COUNTER #2 FOR BAUD RATE
0072 D3DF		260	OUT TMCP ; OUTPUT COMMAND WORD TO INTERVAL TIMER
		261	BRSEL:
0074 210200		262	LXI H,B9600 ; LOAD HIGHEST BAUD RATE FACTOR
0077 3E37		263	MVI A,RESURT ; RESET USART STATUS ERRORS AND-
0079 D3ED		264	OUT CNCTL ; SET 'DTR'
007B 7D		265	MOV A,L ; LEAST SIGNIFICANT WORD FOR CTR2
007C D3DE		266	OUT CTR2 ; OUTPUT WORD TO CTR 2
007E 7C		267	MOV A,H ; MOST SIGNIFICANT WORD FOR CTR2
007F D3DE		268	OUT CTR2 ; OUTPUT WORD TO CTR2

LOC	OBJ	SEQ	SOURCE STATEMENT
0081	11E803	269	LXI D,1000 ; SETUP 1 SECOND TIMEOUT
		270	BRS07:
0084	CDFC04	271	CALL DELAY ; 1 MS DELAY
0087	1B	272	DCX D ; DECREMENT TIMER
0088	DBED	273	IN CONST ; INPUT USART STATUS
008A	E602	274	ANI RBR ; CHECK FOR RECEIVER BUFFER READY
008C	C29700	275	JNZ BRS08 ; NOT YET - WAIT 1 MS AND CHECK AGAIN
008F	7B	276	MOV A,E ; TEST FOR ZERO--
0090	B2	277	ORA D ; AFTER DECREMENTING
0091	C28400	278	JNZ BRS07 ; CONTINUE TO CHECK STATUS FOR 1 SEC
0094	C37400	279	JMP BRSEL ; AFTER 1 SEC REINITIALIZE BAUD RATE SEARCH
		280	BRS08:
0097	DBEC	281	IN CNIN ; READY SO GET CHARACTER
0099	4F	282	MOV C,A ; SAVE CHAR.
009A	E67F	283	ANI PRTY0 ; MASK OFF PARITY BIT
009C	FE55	284	CPI CH96 ; COMPARE FOR CORRECT CHAR.
009E	CA0E01	285	JZ IICR ; GO TO INTERRUPT INITIALIZATION
00A1	29	286	DAD H ; DOUBLE THE CLOCK RATE FOR THE CLOCK
00A2	79	287	MOV A,C ; RESTORE REG A WITH 8 BIT CHAR
00A3	FE66	288	CPI CH48 ; TEST FOR THE 4800 SHIFT CHAR
00A5	CAB400	289	JZ BRS15 ; YES IT IS 4800 BAUD.
00A8	29	290	DAD H ; DOUBLE THE CLOCK RATE FOR CLOCK
00A9	FE78	291	CPI CH24 ; TEST 2400 SHIFTED CHAR.
00AB	CAB400	292	JZ BRS15 ; YES IT IS 2400 BAUD.
00AE	29	293	DAD H ; DOUBLE THE CLOCK RATE FOR CLOCK
00AF	FE80	294	CPI CH12 ; TEST 1200 SHIFTED CHAR.
00B1	C2BD00	295	JNZ BRS20 ; NO THE ENTER SECOND CHAR SEQUENCE
		296	BRS15:
00B4	7D	297	MOV A,L ; LEAST SIGNIFICANT WORD FOR CTR2
00B5	D3DE	298	OUT CTR2 ; OUTPUT WORD TO CTR2
00B7	7C	299	MOV A,H ; MOST SIGNIFICANT BYTE FOR CTR2
00B8	D3DE	300	OUT CTR2 ; OUTPUT BYTE TO CTR2
00BA	C30E01	301	JMP IICR ; GO TO INTERRUPT INITIALIZATION VIA DELAY
		302	BRS20:
00BD	29	303	DAD H ; DOUBLE CLOCK RATE FOR 600 BAUD
00BE	7D	304	MOV A,L ; LEAST BYTE FOR CTR2
00BF	D3DE	305	OUT CTR2 ; OUTPUT TO CTR2
00C1	7C	306	MOV A,H ; MOST WORD FOR CTR2
00C2	D3DE	307	OUT CTR2 ; OUTPUT TO CTR2
00C4	0E78	308	MVI C,120 ; SET UP 120 MS TIMER
		309	BRS25:
00C6	CDFC04	310	CALL DELAY ; 1 MS DELAY
00C9	0D	311	DCR C ; DECREMENT TIMER
00CA	C2C600	312	JNZ BRS25 ; JUMP IF TIMER NOT EXPIRED
00CD	DBEC	313	IN CNIN ; CLEAR USART INPUT BUFFER
00CF	11B80B	314	LXI D,3000 ; SET UP 3 SECOND TIMEOUT
		315	BRS30:
00D2	CDFC04	316	CALL DELAY ; 1 MS DELAY
00D5	1B	317	DCX D ; DECREMENT TIMER
00D6	DBED	318	IN CONST ; INPUT USART STATUS
00D8	E602	319	ANI RBR ; CHECK FOR RECEIVER BUFFER READY
00DA	C2E500	320	JNZ BRS35 ; NOT YET - WAIT 1 MS AND CHECK AGAIN
00DD	7B	321	MOV A,E ; TEST FOR ZERO--
00DE	B2	322	ORA D ; AFTER DECREMENTING
00DF	C2D200	323	JNZ BRS30 ; CONTINUE TO CHECK STATUS FOR 1 SEC??

LOC	OBJ	SEQ	SOURCE STATEMENT
00E2	C37400	324	JMP BRSEL ; AFTER 1 SEC REINITIALIZE BAUD RATE SEARCH?
		325	BRS35:
00E5	DBEC	326	IN CNIN ; READY SO GET SECOND CHAR
00E7	4F	327	MOV C,A ; SAVE CHAR
00E8	E67F	328	ANI PRY0 ; MASK OFF PARITY BIT
00EA	FE55	329	CPI CH60 ; COMPARE FOR CORRECT CHAR.
00EC	CA0E01	330	JZ IICR ; YES 600 BAUD, GO TO INTERRUPT INITIALIZATION
00EF	29	331	DAD H ; DOUBLE CLOCK RATE
00F0	79	332	MOV A,C ; RESTORE REG A
00F1	FE66	333	CPI CH30 ; TEST FOR 300 BAUD
00F3	CAB400	334	JZ BRS15 ; YES 300 BAUD
00F6	29	335	DAD H ; DOUBLE CLOCK RATE
00F7	FE78	336	CPI CH15 ; TEST FOR 150 BAUD
00F9	CAB400	337	JZ BRS15 ; YES 150 BAUD
00FC	21AF00	338	LXI H,B110 ; GET 110 CLOCK RATE 175
00FF	3E40	339	MVI A,RSTUST ; USART RESET VALUE
0101	D3ED	340	OUT CNCTL ; RESET USART TO ACCEPT NEW MODE INST.
0103	3ECF	341	MVI A,MODE2 ; TWO STOP BITS MODE INSTRUCTION
0105	D3ED	342	OUT CNCTL ; LOAD NEW MODE INSTRUCTION
0107	3E35	343	MVI A,TTYSTP ; RESET USART STATUS ERRORS AND-
0109	D3ED	344	OUT CNCTL ; TURN OFF DTR (READER TAPE OFF)
010B	C3B400	345	JMP BRS15 ; SET UP BAUD RATE IN CLOCK
		346	IICR:
010E	0EC8	347	MVI C,200 ; SET UP 200 MS TIMER
		348	IICR5:
0110	CDFC04	349	CALL DELAY ; 1 MS DELAY
0113	0D	350	DCR C ; DECREMENT TIMER
0114	C21001	351	JNZ IICR5 ; JUMP IF TIMER NOT EXPIRED
0117	DBEC	352	IN CNIN ; CLEAR USART INPUT BUFFER
0119	3EF6	353	MVI A,ICW1 ; INITIALIZE INTERRUPT CONTROLLER
011B	D3DA	354	OUT ICCP ; OUTPUT COMMAND WORD #1
011D	3E7F	355	MVI A,ICW2 ;
011F	D3DB	356	OUT ICCP+1 ; OUTPUT COMMAND WORD #2
0121	3E00	357	MVI A,IMASK ; INTERRUPT MASK VALUE
0123	D3DB	358	OUT MSKPT ; OUTPUT MASK WORD TO CONTROLLER
0125	21807F	359	LXI H,USAREA ; INITIALIZE USER STACK POINTER
0128	22C17F	360	SHLD SSAVE ;
012B	F9	361	SPHL ; TEMP STACK POINTER
		362	;
		363	;
		364	;
		365	PRINT SIGNON MESSAGE
		366	;
		367	;
		368	;
012C	218507	369	LXI H,SGNON ; GET ADDRESS OF SIGNON MESSAGE
012F	0615	370	MVI B,LSGNON ; LENGTH OF SIGN ON MESSAGE
		371	LOK15:
0131	4E	372	MOV C,M ; FETCH NEXT CHAR TO C REG
0132	CD0705	373	CALL ECHO ; SEND IT TO THE CONSOLE
0135	23	374	INX H ; POINT TO NEXT CHARACTER
0136	05	375	DCR B ; END OF MESSAGE?
0137	C23101	376	JNZ LOK15 ; RETURN FOR NEXT CHARACTER
		377	;
013A	119A07	378	LXI D,JPTB ; LOAD START OF PROM JUMP TABLE

LOC	OBJ	SEQ	SOURCE STATEMENT
013D	21C47F	379	LXI H, RAMTB ; LOAD START OF RAM JUMP TABLE
0140	060F	380	MVI B, JPLG ; LENGTH OF TABLE IN "B"
		381	LOK20:
0142	36C3	382	MVI M, JMCMD ; PUT JUMP INTO RAM
0144	23	383	INX H
0145	1A	384	LDAX D
0146	77	385	MOV M, A
0147	23	386	INX H
0148	13	387	INX D
0149	1A	388	LDAX D
014A	77	389	MOV M, A
014B	23	390	INX H
014C	13	391	INX D
014D	3600	392	MVI M, 0 ; FOR 4 BYTES SPACING
014F	23	393	INX H
0150	05	394	DCR B
0151	C24201	395	JNZ LOK20 ; LOOP UNTIL DONE
		396 ;	
		397 ;	*****
		398 ;	
		399 ;	
		400 ;	COMMAND RECOGNIZING ROUTINE
		401 ;	
		402 ;	
		403 ;	*****
		404 ;	
		405 ;	FUNCTION* GETCM
		406 ;	INPUTS* NONE
		407 ;	OUTPUTS* NONE
		408 ;	CALLS* GETCH, ECHO, ERROR
		409 ;	DESTROYS* A, B, C, H, L, F/F'S
		410 ;	DESCRIPTION* GETCM RECEIVES AN INPUT CHARACTER FROM THE USER
		411 ;	AND ATTEMPTS TO LOCATE THIS CHARACTER IN ITS COMMAND
		412 ;	CHARACTER TABLE. IF SUCCESSFUL, THE ROUTINE
		413 ;	CORRESPONDING TO THIS CHARACTER IS SELECTED FROM
		414 ;	A TABLE OF COMMAND ROUTINE ADDRESSES, AND CONTROL
		415 ;	IS TRANSFERRED TO THIS ROUTINE. IF THE CHARACTER
		416 ;	DOES NOT MATCH ANY ENTRIES, CONTROL IS PASSED TO
		417 ;	THE ERROR HANDLER;
		418 ;	
		419	GETCM:
0154	31B77F	420	LXI SP, MSTAK ; ALWAYS WANT TO RESET STACK PTR TO MONITOR
		421	;/STARTING VALUE SO ROUTINES NEEDN'T CLEAN UP
0157	0E2E	422	MVI C, '.' ; PROMPT CHARACTER TO C
0159	CD0705	423	CALL ECHO ; SEND PROMPT CHARACTER TO USER TERMINAL
015C	CD3605	424	CALL GETCH ; GET COMMAND CHARACTER TO C
015F	CD0705	425	CALL ECHO ; ECHO CHARACTER TO USER
0162	79	426	MOV A, C ; PUT COMMAND CHARACTER INTO ACCUMULATOR
0163	010900	427	LXI B, NCMSD ; C CONTAINS LOOP AND INDEX COUNT
0166	21CC07	428	LXI H, CTAB ; HL POINTS INTO COMMAND TABLE
		429	GTC05:
0169	BE	430	CMP M ; COMPARE TABLE ENTRY AND CHARACTER
016A	CA7501	431	JZ GTC10 ; BRANCH IF EQUAL - COMMAND RECOGNIZED
016D	23	432	INX H ; ELSE, INCREMENT TABLE POINTER
016E	0D	433	DCR C ; DECREMENT LOOP COUNT



LOC	OBJ	SEQ	SOURCE STATEMENT
016F	C26901	434	JNZ GTC05 ; BRANCH IF NOT AT TABLE END
0172	C32505	435	JMP ERROR ; ELSE, COMMAND CHARACTER IS ILLEGAL
		436	GTC10:
0175	21B807	437	LXI H,CADR ; IF GOOD COMMAND, LOAD ADDRESS OF TABLE
		438	; /OF COMMAND ROUTINE ADDRESSES
0178	09	439	DAD B ; ADD WHAT IS LEFT OF LOOP COUNT
0179	09	440	DAD B ; ADD AGAIN - EACH ENTRY IN CADR IS 2-BYTES LONG
017A	7E	441	MOV A,M ; GET LSP OF ADDRESS OF TABLE ENTRY TO A
017B	23	442	INX H ; POINT TO NEXT BYTE IN TABLE
017C	66	443	MOV H,M ; GET MSP OF ADDRESS OF TABLE ENTRY TO H
017D	6F	444	MOV L,A ; PUT LSP OF ADDRESS OF TABLE ENTRY INTO L
017E	E9	445	PCHL ; NEXT INSTRUCTION COMES FROM COMMAND ROUTINE
		446 ;	
		447 ;	
		448 ;	*****
		449 ;	
		450 ;	
		451 ;	COMMAND IMPLEMENTING ROUTINES
		452 ;	
		453 ;	
		454 ;	*****
		455 ;	
		456 ;	
		457 ;	FUNCTION* DCMD
		458 ;	INPUTS* NONE
		459 ;	OUTPUTS* NONE
		460 ;	CALLS* ECHO,NMOUT,HILO,GETCM,CROUT,GETNM
		461 ;	DESTROYS* A,B,C,D,E,H,L,F/F'S
		462 ;	DESCRIPTION* DCMD IMPLEMENTS THE DISPLAY MEMORY (D) COMMAND
		463 ;	
		464	DCMD:
017F	0E02	465	MVI C,2 ; GET TWO NUMBERS FROM INPUT STREAM
0181	CD7105	466	CALL GETNM ;
0184	D1	467	POP D ; ENDING ADDRESS TO DE
0185	E1	468	POP H ; STARTING ADDRESS TO HL
		469	DCM05:
0186	CD9304	470	CALL ADRD ; DISPLAY ADDRESS
		471	DCM10:
0189	0E20	472	MVI C,' ' ;
018B	CD0705	473	CALL ECHO ; USE BLANK AS SEPARATOR
018E	7E	474	MOV A,M ; GET CONTENTS OF NEXT MEMORY LOCATION
018F	CDD405	475	CALL NMOUT ; DISPLAY CONTENTS
0192	CDAA04	476	CALL BREAK ; SEE IF USER WANTS OUT
0195	DA2A05	477	JC EXIT ; IF SO, BRANCH TO EXIT
0198	CDB605	478	CALL HILO ; SEE IF ADDRESS OF DISPLAYED LOCATION IS
		479	; /GREATER THAN OR EQUAL TO ENDING ADDRESS
019B	DA2A05	480	JC EXIT ; EXIT IF NO MORE TO DISPLAY
019E	23	481	INX H ; IF MORE TO GO, POINT TO NEXT LOC TO DISPLAY
019F	7D	482	MOV A,L ; GET LOW ORDER BITS OF NEW ADDRESS
01A0	E60F	483	ANI NEWLN ; SEE IF LAST HEX DIGIT OF ADDRESS DENOTES
		484	; /START OF NEW LINE
01A2	C28901	485	JNZ DCM10 ; NO - NOT AT END OF LINE
01A5	CDF604	486	CALL CROUT ; ECHO CARRIAGE RETURN/LINE FEED
01A8	C38601	487	JMP DCM05 ; YES - START NEW LINE WITH ADDRESS
		488 ;	

LOC	OBJ	SEQ	SOURCE STATEMENT
		489	;
		490	; *****
		491	;
		492	;
		493	; FUNCTION: GCMD
		494	; INPUTS: NONE
		495	; OUTPUTS: NONE
		496	; CALLS: ERROR,GETHX,RSTTF
		497	; DESTROYS: A,B,C,D,E,H,L,F/F'S
		498	; DESCRIPTION: GCMD IMPLEMENTS THE BEGIN EXECUTION (G) COMMAND.
		499	;
		500	;
		501	; *****
		502	;
		503	;
		504	; GO TO <ADDRESS>, OPTIONALLY SET BREAKPOINTS.
		505	;
		506	; THE G COMMAND IS USED FOR TRANSFERRING CONTROL FROM THE
		507	; MONITOR TO A USER PROGRAM. IT HAS SEVERAL MODES OF
		508	; OPERATION.
		509	;
		510	; IF ONE HEXADECIMAL PARAMETER IS ENTERED, IT IS INTERPRETED
		511	; AS THE ENTRY POINT OF THE USER PROGRAM AND A TRANSFER TO
		512	; THAT LOCATION IS EXECUTED.
		513	;
		514	; IF ADDITIONAL (UP TO 2) PARAMETERS ARE ENTERED, THESE ARE
		515	; CONSIDERED 'BREAKPOINTS', I.E., LOCATIONS WHERE CONTROL
		516	; IS TO BE RETURNED TO THE MONITOR, IF THEY ARE ENCOUNTERED.
		517	;
		518	; IF THE FIRST PARAMETER IS NOT ENTERED, THE STORED VALUE
		519	; OF THE USER'S PROGRAM COUNTER (REGISTER P) IS USED AS
		520	; THE USER PROGRAM ENTRY POINT.
		521	;
		522	GCMD:
01AB	CD3D05	523	CALL GETHX ; GET ADDRESS (IF PRESENT) FROM INPUT STREAM
01AE	D2F901	524	JNC GCM20 ; BRANCH IF NO NUMBER PRESENT
01B1	C5	525	PUSH B ; SAVE NEW PC VALUE
01B2	7A	526	MOV A,D ; GET TERMINATOR
01B3	FE0D	527	CPI CR ; SEE OF CARRIAGE
01B5	CA0D02	528	JZ GCM30 ; BRANCH IF NO OPTIONS
		529	GCM03:
01B8	0E2D	530	MVI C,'-' ; SEND PROMT FOR BREAKPOINT ADDRESS
01BA	CD0705	531	CALL ECHO ; SEND IT
01BD	CD3D05	532	CALL GETHX ; GET BREAK1
01C0	D22505	533	JNC ERROR ; NONE
01C3	C5	534	PUSH B ; MOVE TO REG H-L
01C4	E1	535	POP H ; VIA STACK
01C5	22C57F	536	SHLD BK1AD ; SAVE BREAK 1 ADDRESS
01C8	7A	537	MOV A,D ; GET TERMINATOR
01C9	FE0D	538	CPI CR ;
01CB	CAEA01	539	JZ GCM05 ; ONLY ONE BREAK
01CE	0E2D	540	MVI C,'-' ; SEND PROMPT FOR BREAKPOINT ADDRESS
01D0	CD0705	541	CALL ECHO ; SEND IT
01D3	CD3D05	542	CALL GETHX ; GET BREAK 2
01D6	D2F601	543	JNC GCM10

LOC	OBJ	SEQ	SOURCE STATEMENT
01D9	C5	544	PUSH B ; MOVE TO REG H-L
01DA	E1	545	POP H ; VIA STACK
01DB	22C97F	546	SHLD BK2AD ; SAVE BREAK 2 ADDRESS
01DE	7A	547	MOV A,D ; GET TERMINATOR
01DF	FE0D	548	CPI CR ;
01E1	C2F601	549	JNZ GCM10 ; MUST TERMINATE WITH CR CHAR
01E4	7E	550	MOV A,M ; GET BYTE AT BREAK 2
01E5	32C87F	551	STA BK2BY ; SAVE BYTE FOR BREAK 2
01E8	36CF	552	MVI M,RST1 ; RESTART 1 INSTRUCTION
		553	GCM05:
01EA	2AC57F	554	LHLD BK1AD ; BREAK 1 ADDRESS
01ED	7E	555	MOV A,M ; GET BYTE AT BREAK 1
01EE	32C47F	556	STA BK1BY ; SAVE BYTE FOR BREAK 1
01F1	36CF	557	MVI M,RST1 ; RESTART 1 INSTRUCTION
01F3	C30D02	558	JMP GCM30 ; NOW ENTER GO
		559	GCM10:
01F6	C32505	560	JMP ERROR ; EXIT CLEAR BREAK RAM LOCATIONS ON WAY
		561	GCM20:
01F9	2ABF7F	562	LHLD PSAVE ; FETCH CURRENT PC AND USE IT
01FC	E5	563	PUSH H
01FD	7A	564	MOV A,D ; IF NO STARTING ADDRESS, MAKE SURE THAT
01FE	FE2C	565	CPI ',' ; OR ALLOW A COMMA FOR BREAKPOINT ENTRY
0200	CAB801	566	JZ GCM03 ; YES ASK FOR BREAKPOINTS
0203	FE20	567	CPI ' ' ; BLANK IS ALSO GOOD
0205	CAB801	568	JZ GCM03
0208	FE0D	569	CPI CR ; /CARRIAGE RETURN TERMINATED COMMAND
020A	C22505	570	JNZ ERROR ; ERROR IF NOT
		571	GCM30:
020D	AF	572	XRA A ; RESET SINGLE STEP FLAG FOR GO CMD
020E	E1	573	POP H
020F	22BF7F	574	SHLD PSAVE ; SET UP PSAVE VALUE BASED ON GO VALUE
0212	C3FC06	575	JMP RSTTF ; RESTORE REGISTERS AND BEGIN EXECUTION
		576	GCM40:
0215	210000	577	LXI H,0 ; CLEAR REG H - L
0218	22C57F	578	SHLD BK1AD ; SAVE ADDRESS FOR BREAK 1
021B	22C97F	579	SHLD BK2AD ; SAVE ADDRESS FOR BREAK 2
021E	C9	580	RET ; RETURN
		581	;
		582	;
		583	;
		584	;
		585	*****
		586	;
		587	;
		588	FUNCTION: ICMD
		589	INPUTS: NONE
		590	OUTPUTS: NONE
		591	CALLS: ERROR,ECHO,GETCH,VALDL,VALDG,CNVBN,STHLF,GETNM,CROUT
		592	DESTROYS: A,B,C,D,E,H,L,F/F'S
		593	DESCRIPTION: ICMD IMPLEMENTS THE INSERT CODE INTO MEMORY (I) COMMAND
		594	;
		595	ICMD:
021F	0E01	596	MVI C,1 ;
0221	CD7105	597	CALL GETNM ; GET SINGLE NUMBER FROM INPUT STREAM
0224	3EFF	598	MVI A,UPPER ;

LOC	OBJ	SEQ	SOURCE STATEMENT
0226	32C37F	599	STA TEMP ; TEMP WILL HOLD THE UPPER/LOWER HALF BYTE FLAG
0229	D1	600	POP D ; ADDRESS OF START TO DE
		601	ICM05:
022A	CD3605	602	CALL GETCH ; GET A CHARACTER FROM INPUT STREAM
022D	CD0705	603	CALL ECHO ; ECHO IT
0230	79	604	MOV A,C ; PUT CHARACTER BACK INTO A
0231	FE1B	605	CPI TERM ; SEE IF CHARACTER IS A TERMINATING CHARACTER
0233	CA5F02	606	JZ ICM25 ; IF SO, ALL DONE ENTERING CHARACTERS
0236	CD6D07	607	CALL VALDL ; ELSE, SEE IF VALID DELIMITER
0239	DA2A02	608	JC ICM05 ; IF SO SIMPLY IGNORE THIS CHARACTER
023C	CD5207	609	CALL VALDG ; ELSE, CHECK TO SEE IF VALID HEX DIGIT
023F	D25902	610	JNC ICM20 ; IF NOT, BRANCH TO HANDLE ERROR CONDITION
0242	CDE204	611	CALL CNVBN ; CONVERT DIGIT TO BINARY
0245	4F	612	MOV C,A ; MOVE RESULT TO C
0246	CD3307	613	CALL STHLF ; STORE IN APPROPRIATE HALF WORD
0249	3AC37F	614	LDA TEMP ; GET HALF BYTE FLAG
024C	B7	615	ORA A ; SET F/F'S
024D	C25102	616	JNZ ICM10 ; BRANCH IF FLAG SET FOR UPPER
0250	13	617	INX D ; IF LOWER, INC ADDRESS OF BYTE TO STORE IN
		618	ICM10:
0251	EEFF	619	XRI INVRT ; TOGGLE STATE OF FLAG
0253	32C37F	620	STA TEMP ; PUT NEW VALUE OF FLAG BACK
0256	C32A02	621	JMP ICM05 ; PROCESS NEXT DIGIT
		622	ICM20:
0259	CD2807	623	CALL STHF0 ; ILLEGAL CHARACTER
025C	C32505	624	JMP ERROR ; MAKE SURE ENTIRE BYTE FILLED THEN ERROR
		625	ICM25:
025F	CD2807	626	CALL STHF0 ; HERE FOR ESCAPE CHARACTER - INPUT IS DONE
0262	C32A05	627	JMP EXIT ;
		628	;
		629	;
		630	*****
		631	;
		632	;
		633	FUNCTION: MCMD
		634	INPUTS: NONE
		635	OUTPUTS: NONE
		636	CALLS: GETCM,HILO,GETNM
		637	DESTROYS: A,B,C,D,E,H,L,F/F'S
		638	DESCRIPTION: MCMD IMPLEMENTS THE MOVE DATA IN MEMORY (M) COMMAND.
		639	;
		640	MCMD:
0265	0E03	641	MVI C,3 ;
0267	CD7105	642	CALL GETNM ; GET 3 NUMBERS FROM INPUT STREAM
026A	C1	643	POP B ; DESTINATION ADDRESS TO BC
026B	E1	644	POP H ; ENDING ADDRESS TO HL
026C	D1	645	POP D ; STARTING ADDRESS TO DE
		646	MCM05:
026D	E5	647	PUSH H ; SAVE ENDING ADDRESS
026E	62	648	MOV H,D ;
026F	6B	649	MOV L,E ; SOURCE ADDRESS TO HL
0270	7E	650	MOV A,M ; GET SOURCE BYTE
0271	60	651	MOV H,B ;
0272	69	652	MOV L,C ; DESTINATION ADDRESS TO HL
0273	77	653	MOV M,A ; MOVE BYTE TO DESTINATION

LOC	OBJ	SEQ	SOURCE STATEMENT
0274	03	654	INX B ; INCREMENT DESTINATION ADDRESS
0275	78	655	MOV A,B ;
0276	B1	656	ORA C ; TEST FOR DESTINATION ADDRESS OVERFLOW
0277	CA5401	657	JZ GETCM ; IF SO, CAN TERMINATE COMMAND
027A	13	658	INX D ; INCREMENT SOURCE ADDRESS
027B	E1	659	POP H ; ELSE, GET BACK ENDING ADDRESS
027C	CDB605	660	CALL HILO ; SEE IF ENDING ADDR>=SOURCE ADDR
027F	D25401	661	JNC GETCM ; IF NOT, COMMAND IS DONE
0282	C36D02	662	JMP MCM05 ; MOVE ANOTHER BYTE
		663 ;	
		664 ;	
		665 ;	*****
		666 ;	
		667 ;	
		668 ;	FUNCTION: NCMD
		669 ;	INPUTS: NONE
		670 ;	OUTPUTS: NONE
		671 ;	CALLS: CROUT
		672 ;	DESTROYS: A
		673 ;	DESCRIPTION: NCMD IMPLEMENTS THE SINGLE STEP (N) COMMAND
		674 ;	
		675	NCMD:
0285	CDF604	676	CALL CROUT ; ECHO CR/LF
0288	3EFF	677	MVI A,00EFH ; SET SINGLE STEP FLAG
028A	C3FC06	678	JMP RSTTF ; RESTORE REGISTERS AND EXECUTE NEXT INST.
		679 ;	
		680 ;	
		681 ;	*****
		682 ;	
		683 ;	.R - READ HEXADECIMAL TAPE
		684 ;	
		685 ;	FUNCTION* RCMD
		686 ;	INPUTS* NONE
		687 ;	OUTPUTS* NONE
		688 ;	CALLS* GETCH,ECHO,CO,RICH,BYTE
		689 ;	DESTROYS* A,B,C,D,E,H,L,F/F'S
		690 ;	DESCRIPTION* RCMD IMPLEMENTS THE READ HEXADECIMAL TAPE (R)
		691 ;	COMMAND.
		692 ;	
		693	RCMD:
028D	CD3605	694	CALL GETCH ; GET CARRIAGE RETURN CHARACTER
0290	CD0705	695	CALL ECHO ; ECHO IT
0293	79	696	MOV A,C ; MOVE IT TO A REGISTER
0294	FE0D	697	CPI CR ; SEE IF CARRIAGE RETURN
0296	C22505	698	JNZ ERROR ; ERROR IF NOT PROPERLY TERMINATED
		699	RCM05:
0299	CDF306	700	CALL RICH ; READ CHARACTER FROM TAPE
029C	FE3A	701	CPI ':' ; SEE IF RECORD MARK
029E	C29902	702	JNZ RCM05 ; TRY AGAIN IF NOT MARK
02A1	AF	703	XRA A ; ZERO A REGISTER
02A2	57	704	MOV D,A ; INITIALIZE D FOR HOLDING THE CHECKSUM
02A3	CDBD04	705	CALL BYTE ; READ TWO CHARACTERS FROM TAPE
02A6	CA5401	706	JZ GETCM ; IF ZERO RECORD LENGTH, ALL DONE
02A9	5F	707	MOV E,A ; OTHERWISE, PUT THE RECORD LENGTH IN E
02AA	CDBD04	708	CALL BYTE ; GET MSB OF LOAD ADDRESS

LOC	OBJ	SEQ	SOURCE STATEMENT
02AD	67	709	MOV H,A ; MOVE TO H
02AE	CDBD04	710	CALL BYTE ; GET LSB OF LOAD ADDRESS
02B1	6F	711	MOV L,A ; MOVE TO L
02B2	CDBD04	712	CALL BYTE ; GET RECORD TYPE
02B5	4B	713	MOV C,E ; MOVE RECORD LENGTH TO C
		714	RCM10:
02B6	CDBD04	715	CALL BYTE ; READ DATA FROM TAPE
02B9	77	716	MOV M,A ; PUT DATA INTO MEMORY
02BA	23	717	INX H ; INCREMENT HL FOR NEXT LOCATION
02BB	1D	718	DCR E ; DECREMENT RECORD LENGTH
02BC	C2B602	719	JNZ RCM10 ; LOOP UNTIL DONE
02BF	CDBD04	720	CALL BYTE ; READ CHECKSUM FROM TAPE
02C2	C22505	721	JNZ ERROR ; CHECKSUM ERROR IF NOT ZERO
02C5	C39902	722	JMP RCM05 ; GET ANOTHER RECORD
		723 ;	
		724 ;	
		725 ;	
		726 ;	*****
		727 ;	
		728 ;	.S - SUBSTITUTE INTO MEMORY
		729 ;	
		730 ;	FUNCTION* SCMD
		731 ;	INPUTS* NONE
		732 ;	OUTPUTS* NONE
		733 ;	CALLS* GETHX,GETCM,NMOUT,ECHO
		734 ;	DESTROYS* A,B,C,D,E,H,L,F/F'S
		735 ;	DESCRIPTION* SCMD IMPLEMENTS THE SUBSTITUTE INTO MEMORY (S) COMMAND.
		736 ;	
		737	SCMD:
02C8	2ABF7F	738	LHLD PSAVE ; ASSUME A VALUE FOR S
02CB	CD3D05	739	CALL GETHX ; GET A NUMBER, IF PRESENT, FROM INPUT
02CE	D2D302	740	JNC SCM05 ; IS NUMBER PRESENT
		741	SCM03:
02D1	C5	742	PUSH B ; ADDRESS ENTERED BY USER
02D2	E1	743	POP H ; GET NUMBER TO HL - DENOTES MEMORY LOCATION
		744	SCM05:
02D3	7A	745	MOV A,D ; GET TERMINATOR
02D4	FE20	746	CPI ' ' ; SEE IF SPACE
02D6	CADE02	747	JZ SCM10 ; YES - CONTINUE PROCESSING
02D9	FE2C	748	CPI ',' ; ELSE, SEE IF COMMA
02DB	C22505	749	JNZ ERROR ; NO - TERMINATE COMMAND
		750	SCM10:
02DE	7E	751	MOV A,M ; GET CONTENTS OF SPECIFIED LOCATION TO A
02DF	CDD405	752	CALL NMOUT ; DISPLAY CONTENTS ON CONSOLE
02E2	0E2D	753	MVI C,'-' ;
02E4	CD0705	754	CALL ECHO ; USE DASH FOR SEPARATOR
02E7	CD3D05	755	CALL GETHX ; GET NEW VALUE FOR MEMORY LOCATION, IF ANY
02EA	D2EE02	756	JNC SCM20 ; IF NO VALUE PRESENT, BRANCH
02ED	71	757	MOV M,C ; ELSE, STORE LOWER 8 BITS OF NUMBER ENTERED
		758	SCM20:
02EE	7A	759	MOV A,D ; GET TERMINATOR
02EF	FE0A	760	CPI LF ; SEE IF LINE FEED
02F1	C20D03	761	JNZ SCM25 ; NO CONTINUE
02F4	2B	762	DCX H ; YES WE WILL BACK UP ADDRESS
02F5	0E0D	763	MVI C,CR ; CARRIAGE RETURN PLEASE

LOC	OBJ	SEQ	SOURCE STATEMENT
02F7	CDEB04	764	CALL CO ; PRINT IT ONLY
02FA	0E00	765	MVI C,00 ; NULL CHAR FOR TTY DELAY TIME
02FC	CDEB04	766	CALL CO ; SEND IT
02FF	CDEB04	767	CALL CO ; TWO WILL BE JUST FINE
0302	CD0D06	768	CALL PADR ; ECHO ADDRESS
0305	0E20	769	MVI C,' ' ; SPACE FOR LOOKS PLEASE
0307	CD0705	770	CALL ECHO ; ECHO IT
030A	C3DE02	771	JMP SCM10 ; NOW WE HAVE BACKED UP ONE LETS CONTINUE
		772	SCM25:
030D	FE0D	773	CPI CR ; SEE IF CR, THE PROPER TERMINATING CHARACTER
030F	CA5401	774	JZ GETCM
0312	23	775	INX H ; NO, MUST BE ' ' OR ','
0313	C3DE02	776	JMP SCM10
		777	;
		778	;
		779	; *****
		780	;
		781	; .W - WRITE HEXADECIMAL TAPE
		782	;
		783	; FUNCTION* WCMD
		784	; INPUTS* NONE
		785	; OUTPUTS* NONE
		786	; CALLS* GETNM,LEAD,PO,PBYTE,PADR,PEOL,PEOF
		787	; DESTROYS* A,B,C,D,E,H,L,F/F'S
		788	; DESCRIPTION* WCMD IMPLEMENTS THE WRITE HEXADECIMAL TAPE (W)
		789	; COMMAND.
		790	;
		791	WCMD:
0316	0E02	792	MVI C,2 ;
0318	CD7105	793	CALL GETNM ; GET 2 NUMBERS FROM INPUT STREAM
031B	CDC805	794	CALL LEAD ; PUNCH 60 NULL CHARACTERS FOR TAPE LEADER
031E	D1	795	POP D ; ENDING ADDRESS TO DE
031F	E1	796	POP H ; STARTING ADDRESS TO HL
		797	WCM05:
0320	7D	798	MOV A,L ; MOVE L TO A
0321	C610	799	ADI 16 ; INCREMENT THE LSB OF STARTING ADDRESS BY 16
0323	4F	800	MOV C,A ; MOVE RESULT TO C
0324	7C	801	MOV A,H ; MOVE H TO A
0325	CE00	802	ACI 0 ; ADD CARRY IN FROM PREVIOUS OPERATION
0327	47	803	MOV B,A ; SAVE RESULT IN B
0328	7B	804	MOV A,E ; NOW MOVE LSB OF ENDING ADDRESS TO A
0329	91	805	SUB C ; SUBTRACT LSB OF STARTING ADDRESS
032A	4F	806	MOV C,A ; SAVE IN C
032B	7A	807	MOV A,D ; NOW GET MSB OF ENDING ADDRESS IN A
032C	98	808	SBB B ; SUBTRACT MSB OF STARTING ADDRESS
032D	DA3503	809	JC WCM10 ; BRANCH IF THE RECORD LENGTH IS NOT 16
0330	3E10	810	MVI A,16 ; OTHERWISE SET A TO RECORD LENGTH OF 16
0332	C33803	811	JMP WCM15 ; NOW BRANCH TO PUNCH THE RECORD
		812	WCM10:
0335	79	813	MOV A,C ; THIS IS THE LAST RECORD
0336	C611	814	ADI 17 ; SO SET A TO REMAINING DATA LENGTH
		815	WCM15:
0338	B7	816	ORA A ; CHECK FOR RECORD LENGTH OF ZERO
0339	CA6503	817	JZ WCM25 ; IF IT IS, ALL DONE
033C	D5	818	PUSH D ; OTHERWISE, SAVE ENDING ADDRESS

LOC	OBJ	SEQ	SOURCE STATEMENT
033D	5F	819	MOV E,A ; PUT RECORD LENGTH IN E
033E	1600	820	MVI D,0 ; INITIALIZE D FOR HOLDING CHECKSUM
0340	0E3A	821	MVI C,': ' ;
0342	CDEB04	822	CALL PO ; PUNCH RECORD MARK CHARACTER
0345	7B	823	MOV A,E ; PUT RECORD LENGTH IN A
0346	CD1606	824	CALL PBYTE ; PUNCH RECORD LENGTH
0349	CD0D06	825	CALL PADR ; PUNCH STARTING ADDRESS
034C	AF	826	XRA A ; ZERO A
034D	CD1606	827	CALL PBYTE ; PUNCH RECORD TYPE
		828	WCM20:
0350	7E	829	MOV A,M ; GET DATA TO BE PUNCHED FROM MEMORY
0351	CD1606	830	CALL PBYTE ; PUNCH IT
0354	23	831	INX H ; INCREMENT MEMORY ADDRESS
0355	1D	832	DCR E ; DECREMENT LENGTH COUNT
0356	C25003	833	JNZ WCM20 ; LOOP UNTIL ALL DATA PUNCHED
0359	AF	834	XRA A ;
035A	92	835	SUB D ; PUNCH CHECKSUM
035B	CD1606	836	CALL PBYTE ;
035E	D1	837	POP D ; RESTORE ENDING ADDRESS
035F	CD4B06	838	CALL PEOL ; PUNCH CARRIAGE RETURN AND LINE FEED
0362	C32003	839	JMP WCM05 ;
		840	WCM25:
0365	CD2D06	841	CALL PEOF ; PUNCH END OF FILE RECORD
0368	C32A05	842	JMP EXIT ; ALL DONE
		843	;
		844	;
		845	*****
		846	;
		847	.X - EXAMINE REGISTERS AND CHANGE
		848	;
		849	FUNCTION* XCMD
		850	INPUTS* NONE
		851	OUTPUTS* NONE
		852	CALLS* GETCH,ECHO,REGDS,GETCM,ERROR,RGADR,NMOUT,CROUT,GETHX
		853	DESTROYS* A,B,C,D,E,H,L,F/F'S
		854	DESCRIPTION* XCMD IMPLEMENTS THE REGISTER EXAMINE AND CHANGE (X)
		855	COMMAND.
		856	;
		857	XCMD:
036B	CD3605	858	CALL GETCH ; GET REGISTER IDENTIFIER
036E	CD0705	859	CALL ECHO ; ECHO IT
0371	79	860	MOV A,C ;
0372	FE0D	861	CPI CR ;
0374	C27D03	862	JNZ XCM05 ; BRANCH IF NOT CARRIAGE RETURN
0377	CD6006	863	CALL REGDS ; ELSE, DISPLAY REGISTER CONTENTS
037A	C35401	864	JMP GETCM ; THEN TERMINATE COMMAND
		865	XCM05:
037D	4F	866	MOV C,A ; GET REGISTER IDENTIFIER TO C
037E	CDAA06	867	CALL RGADR ; CONVERT IDENTIFIER INTO RTAB TABLE ADDR
0381	C5	868	PUSH B ;
0382	E1	869	POP H ; PUT POINTER TO REGISTER ENTRY INTO HL
0383	0E20	870	MVI C, ' ' ;
0385	CD0705	871	CALL ECHO ; ECHO SPACE TO USER
0388	79	872	MOV A,C ;
0389	32C37F	873	STA TEMP ; PUT SPACE INTO TEMP AS DELIMITER



LOC	OBJ	SEQ	SOURCE STATEMENT
		874	XCM10:
038C	3AC37F	875	LDA TEMP ; GET TERMINATOR
038F	FE20	876	CPI ' ' ; SEE IF A BLANK
0391	CA9903	877	JZ XCM15 ; YES - GO CHECK POINTER INTO TABLE
0394	FE2C	878	CPI ',' ; NO - SEE IF COMMA
0396	C25401	879	JNZ GETCM ; NO - MUST BE CARRIAGE RETURN TO END COMMAND
		880	XCM15:
0399	7E	881	MOV A,M ;
039A	B7	882	ORA A ; SET F/F'S
039B	CA2A05	883	JZ EXIT ; BRANCH IF AT END OF TABLE
039E	E5	884	PUSH H ; PUT POINTER ON STACK
039F	5E	885	MOV E,M ;
03A0	167F	886	MVI D,HREGS ; FETCH ADDRESS OF SAVE LOCATION FROM
03A2	23	887	INX H ; /TABLE
03A3	46	888	MOV B,M ; FETCH LENGTH FLAG FROM TABLE
03A4	D5	889	PUSH D ; SAVE ADDRESS OF SAVE LOCATION
03A5	D5	890	PUSH D ;
03A6	E1	891	POP H ; MOVE ADDRESS TO HL
03A7	C5	892	PUSH B ; SAVE LENGTH FLAG
03A8	7E	893	MOV A,M ; GET 8 BITS OF REGISTER FROM SAVE LOCATION
03A9	CDD405	894	CALL NMOUT ; DISPLAY IT
03AC	F1	895	POP PSW ; GET BACK LENGTH FLAG
03AD	F5	896	PUSH PSW ; SAVE IT AGAIN
03AE	B7	897	ORA A ; SET F/F'S
03AF	CAB703	898	JZ XCM20 ; IF 8 BIT REGISTER, NOTHING MORE TO DISPLAY
03B2	2B	899	DCX H ; ELSE, FOR 16 BIT REGISTER, GET LOWER 8 BITS
03B3	7E	900	MOV A,M ;
03B4	CDD405	901	CALL NMOUT ; DISPLAY THEM
		902	XCM20:
03B7	0E2D	903	MVI C,'-'
03B9	CD0705	904	CALL ECHO ; USE DASH AS SEPARATOR
03BC	CD3D05	905	CALL GETHX ; SEE IF THERE IS A VALUE TO PUT INTO REGISTER
03BF	D2D703	906	JNC XCM35 ; NO - GO CHECK FOR NEXT REGISTER
03C2	7A	907	MOV A,D ;
03C3	32C37F	908	STA TEMP ; ELSE, SAVE THE TERMINATOR FOR NOW
03C6	F1	909	POP PSW ; GET BACK LENGTH FLAG
03C7	E1	910	POP H ; PUT ADDRESS OF SAVE LOCATION INTO HL
03C8	B7	911	ORA A ; SET F/F'S
03C9	CACE03	912	JZ XCM25 ; IF 8 BIT REGISTER, BRANCH
03CC	70	913	MOV M,B ; SAVE UPPER 8 BITS
03CD	2B	914	DCX H ; POINT TO SAVE LOCATION FOR LOWER 8 BITS
		915	XCM25:
03CE	71	916	MOV M,C ; STORE ALL OF 8 BIT OR LOWER 1/2 OF 16 BIT REG
		917	XCM30:
03CF	110300	918	LXI D,RTABS ; SIZE OF ENTRY IN RTAB TABLE
03D2	E1	919	POP H ; POINTER INTO REGISTER TABLE RTAB
03D3	19	920	DAD D ; ADD ENTRY SIZE TO POINTER
03D4	C38C03	921	JMP XCM10 ; DO NEXT REGISTER
		922	XCM35:
03D7	7A	923	MOV A,D ; GET TERMINATOR
03D8	32C37F	924	STA TEMP ; SAVE IN MEMORY
03DB	D1	925	POP D ; CLEAR STACK OF LENGTH FLAG AND ADDRESS
03DC	D1	926	POP D ; /OF SAVE LOCATION
03DD	C3CF03	927	JMP XCM30 ; GO INCREMENT REGISTER TABLE POINTER
		928	;

```
LOC OBJ      SEQ      SOURCE STATEMENT
          929 ;
          930 ; *****
          931 ;
          932 ;
          933 ;           INTERRUPT SERVICE ROUTINES
          934 ;
          935 ;
          936 ; *****
          937 ;
          938 ;
          939 ; *****
          940 ;
          941 ;
          942 ; FUNCTION: INTIN
          943 ; INPUTS: NONE
          944 ; OUTPUTS: NONE
          945 ; CALLS: REGSV,ECHO,NMOUT,REGDS
          946 ; DESTROYS: A,B,C
          947 ; DESCRIPTION: INTIN HANDLES INTERRUPTS CAUSED BY ACTIVE SIGNALS ON
          948 ; TRAP, RST 6.5, AND RST 5.5, IF THEY ARE NOT HANDLED BY
          949 ; THE USER. IT PRINTS THE INTERRUPT MASK, NEXT INSTRUCTION
          950 ; AND REGISTER VALUES.
          951 ;
          952 ;
          953 INTIN:
03E0 CD9106  954 CALL    REGSV    ; SAVE ALL USERS REGISTERS
03E3 0E49   955 MVI     C,'I'    ;
03E5 CD0705 956 CALL    ECHO     ; OUTPUT INTERRUPT MESSAGE 'I85M=#'
03E8 0E38   957 MVI     C,'8'    ;
03EA CD0705 958 CALL    ECHO     ;
03ED 0E35   959 MVI     C,'5'    ;
03EF CD0705 960 CALL    ECHO     ;
03F2 0E3D   961 MVI     C,'='    ;
03F4 CD0705 962 CALL    ECHO     ;
03F7 20     963 RIM                     ; GET 8085 MASK BYTE
03F8 CDD405 964 CALL    NMOUT   ;
03FB 3E0F   965 MVI     A,TMDIS ; RESET ALL MASK INTERRUPTS
03FD 30     966 SIM                     ; RESET ALL MASK INTERRUPTS
03FE C32904 967 JMP     FND20   ; FINISH EXIT
          968 ;
          969 ;
          970 ; *****
          971 ; FUNCTION: INTIN9
          972 ; INPUTS: NONE
          973 ; OUTPUTS: NONE
          974 ; CALLS: REGSV,ECHO,NMOUT,REGDS
          975 ; DESTROYS: A,B,C
          976 ; DESCRIPTION: INTIN9 HANDLES ANY INTERRUPT REQUESTED BY THE 8259 IF
          977 ; IT IS NOT HANDLED BY THE USER. IT PRINTS THE INTERRUPT
          978 ; MASK, NEXT INSTRUCTION, AND REGISTER VALUES.
          979 ;
          980 ;
          981 INTIN9:
0401 CD9106 982 CALL    REGSV    ; SAVE ALL USERS REGISTERS
0404 0E49   983 MVI     C,'I'    ;
```

LOC	OBJ	SEQ	SOURCE STATEMENT
0406	CD0705	984	CALL ECHO ; OUTPUT INTERRUPT MESSAGE 'I=#'
0409	0E3D	985	MVI C,'=' ;
040B	CD0705	986	CALL ECHO ;
040E	3E0B	987	MVI A,OCW3 ; READ INTERRUPT 'IN SERVICE' REGISTER
0410	D3DA	988	OUT ICCP ;
0412	DBDA	989	IN ICCP ;
0414	0608	990	MVI B,8 ; SET UP TO FIND INTERRUPT NUMBER
0416	0E00	991	MVI C,0 ;
		992	FINTN:
0418	1F	993	RAR ; ROTATE TO CHECK INTERRUPT 'IS' BIT
0419	DA2104	994	JC FNDI ; EXIT IF # FOUND
041C	0C	995	INR C ;
041D	05	996	DCR B ; TRY AGAIN
041E	C21804	997	JNZ FINTN ;
		998	FNDI:
0421	3E20	999	MVI A,EOIC ; END OF INTERRUPT
0423	D3DA	1000	OUT ICCP ;
0425	79	1001	MOV A,C ; MOVE FOR OUTPUT CONVERSION
0426	CDD405	1002	CALL NMOUT ; PRINT INTERRUPT #
		1003	FND20:
0429	CDF604	1004	CALL CROUT ; CARRIAGE RETURN - LINE FEED
042C	C37F04	1005	JMP STP05 ; FINISH AND EXIT
		1006 ;	
		1007 ;	
		1008 ;	*****
		1009 ;	
		1010 ;	
		1011 ;	
		1012	GOBK1:
042F	2AC57F	1013	LHLD BK1AD ; GET BREAK 1 ADDRESS
0432	7C	1014	MOV A,H ;
0433	B5	1015	ORA L ; TEST FOR ZERO
0434	CA3B04	1016	JZ GOBK10 ; NO BREAK ONE
0437	3AC47F	1017	LDA BK1BY ; GET BYTE SAVED
043A	77	1018	MOV M,A ; RESTORE USER RAM
		1019	GOBK10:
043B	2AC97F	1020	LHLD BK2AD ; GET BREAK 2 ADDRESS
043E	7C	1021	MOV A,H ;
043F	B5	1022	ORA L ; TEST FOR ZERO
0440	CA4704	1023	JZ GOBK20 ; NO BREAK TWO
0443	3AC87F	1024	LDA BK2BY ; GET BYTE SAVED
0446	77	1025	MOV M,A ; RESTORE USER RAM
		1026	GOBK20:
0447	CD1502	1027	CALL GCM40 ; CLEAR BREAK ADDRESS LOCATIONS
044A	2ABF7F	1028	LHLD PSAVE ; GET USER P REG
044D	2B	1029	DCX H ; DEC IT PLEASE
044E	22BF7F	1030	SHLD PSAVE ; NOW IT SHOULD BE CORRECT
0451	0E23	1031	MVI C,'#' ;
0453	CD0705	1032	CALL ECHO ; PRINT # CHAR
0456	C38204	1033	JMP STP10 ; GO AND DISPLAY ADDRESS AT BREAK
		1034 ;	
		1035 ;	*****
		1036 ;	
		1037 ;	
		1038 ;	FUNCTION* STEPIN

LOC	OBJ	SEQ	SOURCE STATEMENT
		1039	; INPUTS* NONE
		1040	; OUTPUTS* NONE
		1041	; CALLS* REGSV,REGDS,NXTIN
		1042	; DESTROYS* A,F/F'S
		1043	; DESCRIPTION* STEPIN OUTPUTS DATA AFTER SINGLE STEP TIMER INTERRUPT
		1044	;
		1045	;
		1046	STEPIN:
0459	CD9106	1047	CALL REGSV ; SAVE ALL REGISTERS ON ENTRY
045C	3E0F	1048	MVI A,TMDIS ; STOP INTERRUPTS
045E	30	1049	SIM ; RESET ALL MASK INTERRUPTS
045F	3AC07F	1050	LDA PSAVE+1 ; TEST FOR SINGLE STEP INTO BREAKPOINT
0462	A7	1051	ANA A ;
0463	C27F04	1052	JNZ STP05 ; PC HIGH=0 FOR BREAKPOINT ADDRESS
0466	3ABF7F	1053	LDA PSAVE ;
0469	FE40	1054	CPI LLOW ; PC LOW < RST FOR BREAKPOINT ADDRESS
046B	D27F04	1055	JNC STP05 ; CONTINUE IF NO USER BREAKPOINT
046E	2AC17F	1056	LHLD SSAVE ; GET USER STACK POINTER
0471	5E	1057	MOV E,M ; RESTORE ADDRESS OF USER BREAKPOINT
0472	23	1058	INX H ;
0473	56	1059	MOV D,M ;
0474	23	1060	INX H ;
0475	22C17F	1061	SHLD SSAVE ; UPDATE USER STACK POINTER
0478	EB	1062	XCHG ; GET BREAKPOINT ADDRESS INTO HL
0479	22BF7F	1063	SHLD PSAVE ; UPDATE USER P COUNTER
047C	C39C04	1064	JMP ADROUT ; PRINT BREAKPOINT ENTRY
		1065	STP05:
047F	CD6006	1066	CALL REGDS ; OUTPUT REGISTERS
		1067	STP10:
0482	2ABF7F	1068	LHLD PSAVE ; LOAD USER P COUNTER
0485	CD9304	1069	CALL ADRD ; DISPLAY ADDRESS
0488	0E20	1070	MVI C,' ' ; SPACE
048A	CD0705	1071	CALL ECHO ; PRINT IT
048D	CDE705	1072	CALL NXTIN ; OUTPUT 3 BYTES FOR NEXT INSTRUCTION
0490	C35401	1073	JMP GETCM ;
		1074	;
		1075	;
		1076	; *****
		1077	;
		1078	;
		1079	UTILITY ROUTINES
		1080	;
		1081	;
		1082	; *****
		1083	;
		1084	;
		1085	; *****
		1086	;
		1087	;
		1088	; FUNCTION* ADRD
		1089	; INPUTS* HL - ADDRESS TO BE DISPLAYED
		1090	; OUTPUTS* NONE
		1091	; CALLS* NMOUT
		1092	; DESTROYS* A
		1093	; DESCRIPTION* ADRD OUTPUTS TO THE CONSOLE THE ADDRESS

LOC	OBJ	SEQ	SOURCE STATEMENT
		1094 ;	CONTAINED IN THE H,L REGISTERS.
		1095 ;	
		1096	ADRD:
0493	7C	1097	MOV A,H ; DISPLAY FIRST HALF OF ADDRESS
0494	CDD405	1098	CALL NMOUT ;
0497	7D	1099	MOV A,L ; DISPLAY SECOND HALF OF ADDRESS
0498	CDD405	1100	CALL NMOUT ;
049B	C9	1101	RET ; RETURN TO CALLING ROUTINE
		1102 ;	
		1103 ;	
		1104 ;	*****
		1105 ;	
		1106 ;	
		1107 ;	FUNCTION* ADROUT
		1108 ;	INPUTS* USER REGISTERS ON THE STACK
		1109 ;	OUTPUTS* NOTHING
		1110 ;	CALLS* ECHO,ADRD
		1111 ;	DESTROYS* A,B,C,D,E,H,L,F/F'S
		1112 ;	DESCRIPTION* ADROUT OUTPUTS THE USER P COUNTER TO THE CONSOLE
		1113 ;	AFTER AN RST 1 INSTRUCTION.
		1114 ;	
		1115	ADROUT:
049C	0E23	1116	MVI C,'#' ;
049E	CD0705	1117	CALL ECHO ; OUTPUT '#'
04A1	2ABF7F	1118	LHLD PSAVE ; LOAD USER P COUNTER
04A4	CD9304	1119	CALL ADRD ; DISPLAY ADDRESS
04A7	C32A05	1120	JMP EXIT ; GET NEW COMMAND
		1121 ;	
		1122 ;	
		1123 ;	*****
		1124 ;	
		1125 ;	
		1126 ;	FUNCTION: BREAK
		1127 ;	INPUTS: NONE
		1128 ;	OUTPUTS: CARRY - 1 IF EXCAPE CHARACTER INPUT
		1129 ;	- 0 IF ANY OTHER CHARACTER OR NO CHAR PENDING
		1130 ;	CALLS: NOTHING
		1131 ;	DESTROYS: A,F/F'S
		1132 ;	DESCRIPTION: BREAK IS USED TO SENSE AN ESCAPE CHARACTER FROM
		1133 ;	THE USER. IF NO CHARACTER IS PENDING, OR IF THE
		1134 ;	PENDING CHARACTER IS NOT THE EXCAPE, THEN A FAILURE
		1135 ;	RETURN (CARRY=0) IS TAKEN. IN THIS CASE, THE
		1136 ;	PENDING CHARACTER (IF ANY) IS LOST. IF THE PENDING
		1137 ;	CHARACTER IS AN EXCAPE CHARACTER, BREAK TAKES A SUCCESS
		1138 ;	RETURN (CARRY-1).
		1139 ;	
		1140	BREAK:
04AA	DBED	1141	IN CONST ; GET CONSOLE STATUS
04AC	E602	1142	RBR ; SEE IF CHARACTER PENDING
04AE	CA3305	1143	JZ FRET ; NO - TAKE FAILURE RETURN
04B1	DBEC	1144	IN CNIN ; YES - PICK UP CHARACTER
04B3	E67F	1145	ANI PRTY0 ; STRIP OFF PARITY BIT
04B5	FE1B	1146	CPI ESC ; SEE IF BREAK CHARACTER
04B7	CA2607	1147	JZ SRET ; YES - SUCCESS RETURN
04BA	C33305	1148	JMP FRET ; NO - FAILURE RETURN - CHARACTER LOST

LOC	OBJ	SEQ	SOURCE STATEMENT
		1149	;
		1150	;
		1151	;
		1152	;
		1153	*****
		1154	;
		1155	;
		1156	; FUNCTION* BYTE
		1157	; INPUTS* D - CURRENT VALUE OF CHECKSUM
		1158	; OUTPUTS* A - HEXADECIMAL CHARACTER
		1159	; D - UPDATED VALUE OF CHECKSUM
		1160	; Z FLAG - SET IF (D) = 0, CLEARED IF (D) <> 0
		1161	; CALLS* RICH,CNVBN
		1162	; DESTROYS* A,B,C,D,F/F'S
		1163	; DESCRIPTION* BYTE READS 2 ASCII CHARACTERS FROM THE TELETYPEWRITER
		1164	; AND CONVERTS THE CHARACTERS TO ONE HEXADECIMAL CHARACTER.
		1165	; THE A REGISTER CONTAINS THE FINAL CHARACTER AND THE
		1166	; D REGISTER CONTAINS THE UPDATED VALUE OF
		1167	; THE CHECKSUM.
		1168	;
		1169	BYTE:
04BD	C5	1170	PUSH B ; SAVE BC
04BE	CDF306	1171	CALL RICH ; READ ASCII CHARACTER FROM TAPE
04C1	4F	1172	MOV C,A ;
04C2	CDE204	1173	CALL CNVBN ; CONVERT CHARACTER TO HEXADECIMAL
04C5	07	1174	RLC ; POSITION VALUE INTO UPPER 4 BITS
04C6	07	1175	RLC ;
04C7	07	1176	RLC ;
04C8	07	1177	RLC ;
04C9	47	1178	MOV B,A ; SAVE RESULTS IN B
04CA	CDF306	1179	CALL RICH ; GET ANOTHER CHARACTER FROM TAPE
04CD	4F	1180	MOV C,A ;
04CE	CDE204	1181	CALL CNVBN ; CONVERT IT
04D1	B0	1182	ORA B ; OR IN THE UPPER 4 BITS
04D2	4F	1183	MOV C,A ; SAVE
04D3	82	1184	ADD D ; INCREMENT CHECKSUM
04D4	57	1185	MOV D,A ;
04D5	79	1186	MOV A,C ; RESTORE HEX DATA TO A REGISTER
04D6	C1	1187	POP B ; RESTORE BC
04D7	C9	1188	RET ;
		1189	;
		1190	;
		1191	*****
		1192	;
		1193	; FUNCTION* CI
		1194	; INPUTS* NONE
		1195	; OUTPUTS* A - CHARACTER FROM CONSOLE (8-BITS)
		1196	; CALLS* DELAY
		1197	; DESTROYS* A,F/F'S
		1198	; DESCRIPTION* CI WAITS UNTIL A CHARACTER HAS BEEN ENTERED AT THE
		1199	; CONSOLE AND THEN RETURNS THE CHARACTER, VIA THE A
		1200	; REGISTER, TO THE CALLING ROUTINE. THIS ROUTINE
		1201	; IS CALLED BY THE USER VIA A JUMP TABLE IN RAM.
		1202	;
		1203	CI:

```

LOC  OBJ          SEQ          SOURCE STATEMENT
04D8  DBED        1204          IN          CONST    ; GET STATUS OF CONSOLE
04DA  E602        1205          ANI         RBR      ; CHECK FOR RECEIVER BUFFER READY
04DC  CAD804      1206          JZ          CI       ; NOT YET - WAIT
04DF  DBEC        1207          IN          CNIN    ; READY SO GET CHARACTER
04E1  C9          1208          RET
1209 ;
1210 ; *****
1211 ;
1212 ;
1213 ; FUNCTION: CNVBN
1214 ; INPUTS: C - ASCII CHARACTER '0'-'9' OR 'A'-'F'
1215 ; OUTPUTS: A - 0 TO F HEX
1216 ; CALLS: NOTHING
1217 ; DESTROYS: A,F/F'S
1218 ; DESCRIPTION: CNVBN CONVERTS THE ASCII REPRESENTATION OF A HEX
1219 ; CHARACTER INTO ITS CORRESPONDING BINARY VALUE. CNVBN
1220 ; DOES NOT CHECK THE VALIDITY OF ITS INPUT.
1221 ;
1222 CNVBN:
04E2  79          1223          MOV         A,C     ;
04E3  D630       1224          SUI         '0'    ; SUBTRACT CODE FOR '0' FROM ARGUMENT
04E5  FE0A       1225          CPI         10     ; WANT TO TEST FOR RESULT OF 0 TO 9
04E7  F8         1226          RM          ; IF SO, THEN ALL DONE
04E8  D607       1227          SUI         7      ; ELSE, RESULT BETWEEN 17 AND 23 DECIMAL
04EA  C9         1228          RET          ; SO RETURN AFTER SUBTRACTING BIAS OF 7
1229 ;
1230 ;
1231 ;
1232 ; *****
1233 ;
1234 ;
1235 ; FUNCTION: PO
1236 ; INPUTS* C - CHARACTER TO BE PUNCHED
1237 ; OUTPUTS* NONE
1238 ; CALLS* CO
1239 ; DESTROYS* NOTHING
1240 ; DESCRIPTION* PO PUNCHES THE CHARACTER SUPPLIED IN TH C REGISTER TO
1241 ; THE USER TELETYPEWRITER.
1242 ;
1243 PO:
1244 ; THIS WILL NOW BE THE SAME AS THE CALL CONSOLE.
1245 ;
1246 ; *****
1247 ;
1248 ;
1249 ; FUNCTION* CO
1250 ; INPUTS* C - CHARACTER TO OUTPUT TO CONSOLE
1251 ; OUTPUTS* C - CHARACTER OUTPUT TO CONSOLE
1252 ; CALLS* SPDLY,MKDLY,DELAY
1253 ; DESTROYS* A,F/F'S
1254 ; DESCRIPTION* CO SENDS THE INPUT ARGUMENT TO THE CONSOLE.
1255 ; AND THEN SENDS THE INPUT ARGUMENT TO THE CONSOLE.
1256 ;
1257 CO:
04EB  DBED       1258          IN          CONST    ; GET STATUS OF CONSOLE

```

LOC	OBJ	SEQ	SOURCE STATEMENT
04ED	E601	1259	ANI TRDY ; SEE IF TRANSMITTER READY
04EF	CAEB04	1260	JZ CO ; NO - WAIT
04F2	79	1261	MOV A,C ; ELSE, MOVE CHARACTER TO A REG FOR OUTPUT
04F3	D3EC	1262	OUT CNOUT ; SEND TO CONSOLE
04F5	C9	1263	RET
		1264	;
		1265	*****
		1266	;
		1267	;
		1268	; FUNCTION: CROUT
		1269	; INPUTS: NONE
		1270	; OUTPUTS: NONE
		1271	; CALLS: ECHO
		1272	; DESTROYS: A,B,C,F/F'S
		1273	; DESCRIPTION: CROUT SENDS A CARRIAGE RETURN (AND HENCE A LINE
		1274	FEED) TO THE CONSOLE.
		1275	;
		1276	CROUT:
04F6	0E0D	1277	MVI C,CR ;
04F8	CD0705	1278	CALL ECHO ; OUTPUT CARRIAGE RETURN TO USER TERMINAL
04FB	C9	1279	RET ;
		1280	;
		1281	*****
		1282	;
		1283	; FUNCTION* DELAY
		1284	; INPUTS* NONE
		1285	; OUTPUTS* NONE
		1286	; CALLS* NOTHING
		1287	; DESTROYS* NOTHING
		1288	; DESCRIPTION* DELAY PROVIDES A PROGRAMMED DELAY OF 1 MILLISECOND
		1289	;
		1290	DELAY:
04FC	C5	1291	PUSH B ; SAVE BC REGISTERS
04FD	068B	1292	MVI B,ONEMS ; LOAD 1 MILLISECOND CONSTANT
		1293	DEL1:
04FF	05	1294	DCR B ; DECREMENT COUNTER
0500	00	1295	NOP ; EXTRA TIMMING FOR 8085 TIMMING
0501	00	1296	NOP ;
0502	C2FF04	1297	JNZ DEL1 ; JUMP IF NOT DONE
0505	C1	1298	POP B ; RESTORE BC REGISTERS
0506	C9	1299	RET ; RETURN TO CALLING ROUTINE
		1300	;
		1301	*****
		1302	;
		1303	; FUNCTION: ECHO
		1304	; INPUTS: C - CHARACTER TO ECHO TO TERMINAL
		1305	; OUTPUTS: C - CHARACTER ECHOED TO TERMINAL
		1306	; CALLS: CO
		1307	; DESTROYS: A,F/F'S
		1308	; DESCRIPTION: ECHO TAKES A SINGLE CHARACTER AS INPUT AND, VIA
		1309	THE MONITOR, SENDS THAT CHARACTER TO THE USER
		1310	TERMINAL; A CARRIAGE RETURN IS ECHOED AS A CARRIAGE
		1311	RETURN LINE FEED, AND AN ESCAPE CHARACTER IS ECHOED AS \$
		1312	;
		1313	ECHO:



LOC	OBJ	SEQ	SOURCE STATEMENT
0507	C5	1314	PUSH B ; SAVE ARGUMENTT
0508	3E1B	1315	MVI A,ESC ;
050A	B9	1316	CMP C ; SEE IF ECHOING AN ESCAPE CHARACTER
050B	C21005	1317	JNZ ECH05 ; NO - BRANCH
050E	0E24	1318	MVI C,'\$' ; YES - ECHO AS \$
		1319	ECH05:
0510	CDEB04	1320	CALL CO ; DO OUTPUT THROUGH MONITOR
0513	3E0D	1321	MVI A,CR ;
0515	B9	1322	CMP C ; SEE IF CHARACTER ECHOED WAS A CARRIAGE RE...
0516	C22305	1323	JNZ ECH10 ; NO - NO NEED TO TAKE SPECIAL ACTION
0519	0E00	1324	MVI C,00 ; NULL FOR AUTO LINE FEED CRT
051B	CDEB04	1325	CALL CO ; ALLOW ANOTHER CHARACTER DELAY FOR TTY
051E	0E0A	1326	MVI C,LF ; YES - WANT TO ECHO LINE FEED, TOO
0520	CDEB04	1327	CALL CO ;
		1328	ECH10:
0523	C1	1329	POP B ; RESTORE ARGUMENT
0524	C9	1330	RET ;
		1331	;
		1332	;
		1333	*****
		1334	;
		1335	;
		1336	; FUNCTION: ERROR
		1337	; INPUTS: NONE
		1338	; OUTPUTS: NONE
		1339	; CALLS: GCM40,ECHO,CROUT,GETCM
		1340	; DESTROYS: A,B,C,F/F'S
		1341	; DESCRIPTION: ERROR PRINTS THE ERROR CHARACTER (CURRENTLY A CHECK
		1342	ON THE CONSOLE, FOLLOWED BY A CARRIAGE RETURN-LINE FEED,
		1343	AND THEN RETURNS CONTROL TO THE COMMAND RECOGNIZER.
		1344	;
		1345	ERROR:
0525	0E23	1346	MVI C,'#' ;
0527	CD0705	1347	CALL ECHO ; SEND # TO CONSOLE
		1348	EXIT:
052A	CDF604	1349	CALL CROUT ; SKIP TO BEGINNING OF NEXT LINE
052D	CD1502	1350	CALL GCM40 ; IN CASE OF ERROR, CLEAR BREAK RAM ADDRESSES
0530	C35401	1351	JMP GETCM ; TRY AGAIN FOR ANOTHER COMMAND
		1352	;
		1353	;
		1354	*****
		1355	;
		1356	;
		1357	; FUNCTION: FRET
		1358	; INPUTS: NONE
		1359	; OUTPUTS: CARRY - ALWAYS 0
		1360	; CALLS: NOTHING
		1361	; DESTROYS: CARRY
		1362	; DESCRIPTION: FRET IS JUMPED TO BY ANY ROUTINE THAT WISHES TO
		1363	INDICATE FAILURE ON RETURN. FRET SETS THE CARRY
		1364	FALSE, DENOTING FAILURE, AND THEN RETURNS TO THE
		1365	CALLER OF THE ROUTINE INVOKING FRET.
		1366	;
		1367	FRET:
0533	37	1368	STC ; FIRST SET CARRY TRUE

LOC	OBJ	SEQ	SOURCE STATEMENT
0534	3F	1369	CMC ; THEN COMPLEMENT IT TO MAKE IT FALSE
0535	C9	1370	RET ; RETURN APPROPRIATELY
		1371 ;	
		1372 ;	
		1373 ;	*****
		1374 ;	
		1375 ;	
		1376 ;	FUNCTION: GETCH
		1377 ;	INPUTS: NONE
		1378 ;	OUTPUTS: C - NEXT CHARACTER IN INPUT STREAM
		1379 ;	CALLS: CI
		1380 ;	DESTROYS: A,C,F/F'S
		1381 ;	DESCRIPTION: GETCH RETURNS THE NEXT CHARACTER IN THE INPUT STREAM
		1382 ;	TO THE CALLING PROGRAM.
		1383 ;	
		1384	GETCH:
0536	CDD804	1385	CALL CI ; GET CHARACTER FROM TERMINAL
0539	E67F	1386	ANI PRTY0 ; TURN OFF PARITY BIT IN CASE SET BY CONSOLE
053B	4F	1387	MOV C,A ; PUT VALUE IN C REGISTER FOR RETURN
053C	C9	1388	RET ;
		1389 ;	
		1390 ;	
		1391 ;	*****
		1392 ;	
		1393 ;	
		1394 ;	FUNCTION: GETHX
		1395 ;	INPUTS: NONE
		1396 ;	OUTPUTS: BC - 16 BIT INTEGER
		1397 ;	D - CHARACTER WHICH TERMINATED THE INTEGER
		1398 ;	CARRY - 1 IF FIRST CHARACTER NOT DELIMITER
		1399 ;	- 0 IF FIRST CHARACTER IS DELIMITER
		1400 ;	CALLS: GETCH,ECHO,VALDL,VALDG,CNVBN,ERROR
		1401 ;	DESTROYS: A,B,C,D,E,F/F'S
		1402 ;	DESCRIPTION: GETHX ACCEPTS A STRING OF HEX DIGITS FROM THE INPUT
		1403 ;	STREAM AND RETURNS THEIR VALUE AS A 16 BIT BINARY
		1404 ;	INTEGER. IF MORE THAN 4 HEX DIGITS ARE ENTERED,
		1405 ;	ONLY THE LAST 4 ARE USED. THE NUMBER TERMINATES WHEN
		1406 ;	A VALID DELIMITER IS ENCOUNTERED. THE DELIMITER IS
		1407 ;	ALSO RETURNED AS AN OUTPUT OF THE FUNCTION. ILLEGAL
		1408 ;	CHARACTERS (NOT HEX DIGITS OR DELIMITERS) CAUSE AN
		1409 ;	ERROR INDICATION. IF THE FIRST (VALID) CHARACTER
		1410 ;	ENCOUNTERED IN THE INPUT STREAM IS NOT A DELIMITER,
		1411 ;	GETHX WILL RETURN WITH THE CARRY BIT SET TO 1
		1412 ;	OTHERWISE, THE CARRY BIT IS SET TO 0 AND THE CONTENTS
		1413 ;	OF BC ARE UNDEFINED.
		1414 ;	
		1415	GETHX:
053D	E5	1416	PUSH H ; SAVE HL
053E	210000	1417	LXI H,0 ; INITIALIZE RESULT
0541	1E00	1418	MVI E,0 ; INITIALIZE DIGIT FLAG TO FALSE
		1419	GHX05:
0543	CD3605	1420	CALL GETCH ; GET A CHARACTER
0546	CD0705	1421	CALL ECHO ; ECHO THE CHARACTER
0549	CD6D07	1422	CALL VALDL ; SEE IF DELIMITER
054C	D25B05	1423	JNC GHX10 ; NO - BRANCH

LOC	OBJ	SEQ	SOURCE STATEMENT
054F	51	1424	MOV D,C ; YES - ALL DONE, BUT WANT TO RETURN DELIMITER
0550	E5	1425	PUSH H ;
0551	C1	1426	POP B ; MOVE RESULT TO BC
0552	E1	1427	POP H ; RESTORE HL
0553	7B	1428	MOV A,E ; GET FLAG
0554	B7	1429	ORA A ; SET F/F'S
0555	C22607	1430	JNZ SRET ; IF FLAG NON-0, A NUMBER HAS BEEN FOUND
0558	CA3305	1431	FRET ; ELSE, DELIMITER WAS FIRST CHARACTER
		1432	GHX10:
055B	CD5207	1433	CALL VALDG ; IF NOT DELIMITER, SEE IF DIGIT
055E	D22505	1434	JNC ERROR ; ERROR IF NOT A VALID DIGIT, EITHER
0561	CDE204	1435	CALL CNVBN ; CONVERT DIGIT TO ITS BINARY VALUE
0564	1EFF	1436	MVI E,00FFH ; SET DIGIT FLAG NON-0
0566	29	1437	DAD H ; *2
0567	29	1438	DAD H ; *4
0568	29	1439	DAD H ; *8
0569	29	1440	DAD H ; *16
056A	0600	1441	MVI B,0 ; CLEAR UPPER 8 BITS OF BC PAIR
056C	4F	1442	MOV C,A ; BINARY VALUE OF CHARACTER INTO C
056D	09	1443	DAD B ; ADD THIS VALUE TO PARTIAL RESULT
056E	C34305	1444	JMP GHX05 ; GET NEXT CHARACTER
		1445 ;	
		1446 ;	
		1447 ;	*****
		1448 ;	
		1449 ;	
		1450 ;	FUNCTION* GETNM
		1451 ;	INPUTS* C - COUNT OF NUMBERS TO FIND IN INPUT STREAM
		1452 ;	OUTPUTS* TOP OF STACK - NUMBERS FOUND IN REVERSE ORDER (LAST ON T
		1453 ;	OF STACK)
		1454 ;	CALLS* GETHX,HILO,ERROR
		1455 ;	DESTROYS* A,B,C,D,E,H,L,F/F'S
		1456 ;	DESCRIPTION* GETNM FINDS A SPECIFIED COUNT OF NUMBERS, BETWEEN 1
		1457 ;	AND 3, INCLUSIVE, IN THE INPUT
		1458 ;	STREAM AND RETURNS THEIR VALUES ON THE STACK. IF 2
		1459 ;	OR MORE NUMBERS ARE REQUESTED, THEN THE FIRST MUST BE
		1460 ;	LESS THAN OR EQUAL TO THE SECOND, OR THE FIRST AND
		1461 ;	SECOND NUMBERS WILL BE SET EQUAL. THE LAST NUMBER
		1462 ;	REQUESTED MUST BE TERMINATED BY A CARRIAGE RETURN
		1463 ;	OR AN ERROR INDICATION WILL RESULT.
		1464 ;	
		1465	GETNM:
0571	2E03	1466	MVI L,3 ; PUT MAXIMUM ARGUMENT COUNT INTO L
0573	79	1467	MOV A,C ; GET THE ACTUAL ARGUMENT COUNT
0574	E603	1468	ANI 3 ; FORCE TO MAXIMUM OF 3
0576	C8	1469	RZ ; IF 0, DON'T BOTHER TO DO ANYTHING
0577	67	1470	MOV H,A ; ELSE, PUT ACTUAL COUNT INTO H
		1471	GNM05:
0578	CD3D05	1472	CALL GETHX ; GET A NUMBER FROM INPUT STREAM
057B	D22505	1473	JNC ERROR ; ERROR IF NOT THERE - TOO FEW NUMBERS
057E	C5	1474	PUSH B ; ELSE, SAVE NUMBER ON STACK
057F	2D	1475	DCR L ; DECREMENT MAXIMUM ARGUMENT COUNT
0580	25	1476	DCR H ; DECREMENT ACTUAL ARGUMENT COUNT
0581	CA8D05	1477	JZ GNM10 ; BRANCH IF NO MORE NUMBERS WANTED
0584	7A	1478	MOV A,D ; ELSE, GET NUMBER TERMINATOR TO A

LOC	OBJ	SEQ	SOURCE STATEMENT
0585	FE0D	1479	CPI CR ; SEE IF CARRIAGE RETURN
0587	CA2505	1480	JZ ERROR ; ERROR IF SO - TOO FEW NUMBERS
058A	C37805	1481	JMP GNM05 ; ELSE, PROCESS NEXT NUMBER
		1482	GNM10:
058D	7A	1483	MOV A,D ; WHEN COUNT 0, CHECK LAST TERMINATOR
058E	FE0D	1484	CPI CR ;
0590	C22505	1485	JNZ ERROR ; ERROR IF NOT CARRIAGE RETURN
0593	01FFFF	1486	LXI B,0FFFFH; HL GETS LARGEST NUMBER
0596	7D	1487	MOV A,L ; GET WHAT'S LEFT OF MAXIMUM ARG COUNT
0597	B7	1488	ORA A ; CHECK FOR 0
0598	CAA005	1489	JZ GNM20 ; IF YES, 3 NUMBERS WERE INPUT
		1490	GNM15:
059B	C5	1491	PUSH B ; IF NOT, FILL REMAINING ARGUMENTS WITH 0FFFFH
059C	2D	1492	DCR L ;
059D	C29B05	1493	JNZ GNM15 ;
		1494	GNM20:
05A0	C1	1495	POP B ; GET THE 3 ARGUMENTS OUT
05A1	D1	1496	POP D ;
05A2	E1	1497	POP H ;
05A3	CDB605	1498	CALL HILO ; SEE IF FIRST >= SECOND
05A6	D2AB05	1499	JNC GNM25 ; NO - BRANCH
05A9	54	1500	MOV D,H ;
05AA	5D	1501	MOV E,L ; YES - MAKE SECOND EQUAL TO THE FIRST
		1502	GNM25:
05AB	E3	1503	XTHL ; PUT FIRST ON STACK - GET RETURN ADDR
05AC	D5	1504	PUSH D ; PUT SECOND ON STACK
05AD	C5	1505	PUSH B ; PUT THIRD ON STACK
05AE	E5	1506	PUSH H ; PUT RETURN ADDRESS ON STACK
		1507	GNM30:
05AF	3D	1508	DCR A ; DECREMENT RESIDUAL COUNT
05B0	F8	1509	RM ; IF NEGATIVE, PROPER RESULTS ON STACK
05B1	E1	1510	POP H ; ELSE, GET RETURN ADDR
05B2	E3	1511	XTHL ; REPLACE TOP RESULT WITH RETURN ADDR
05B3	C3AF05	1512	JMP GNM30 ; TRY AGAIN
		1513	;
		1514	;
		1515	*****
		1516	;
		1517	;
		1518	; FUNCTION* HILO
		1519	; INPUTS* DE - 16 BIT INTEGER
		1520	; HL - 16 BIT INTEGER
		1521	; OUTPUTS* CARRY - 0 IF HL<DE
		1522	; - 1 IF HL>=DE
		1523	; CALLS* NOTHING
		1524	; DESTROYS* A,F/F'S
		1525	; DESCRIPTION* HILO COMPARES THE 2 16 BIT INTEGERS IN HL AND DE. THE
		1526	; INTEGERS ARE TREATED AS UNSIGNED NUMBERS. THE CARRY
		1527	; BIT IS SET ACCORDING TO THE RESULT OF THE COMPARISON.
		1528	;
		1529	HILO:
05B6	C5	1530	PUSH B ; SAVE BC
05B7	47	1531	MOV B,A ; SAVE A REGISTER
05B8	23	1532	INX H ; INCREMENT HL BY 1
05B9	7C	1533	MOV A,H ; WANT TO TEST FOR 0 RESULT AFTER

LOC	OBJ	SEQ	SOURCE STATEMENT
05BA	B5	1534	ORA L ; /INCREMENTING
05BB	2B	1535	DCX H ; RESTORE HL
05BC	37	1536	STC ; SET CARRY
05BD	CAC505	1537	JZ HIL05 ; IF SO, CARRY IS SET PROPERLY
05C0	7D	1538	MOV A,L ; IF NOT, MOVE L TO A
05C1	93	1539	SUB E ; SUBTRACT E
05C2	7C	1540	MOV A,H ; MOVE H TO A
05C3	9A	1541	SBB D ; SUBTRACT D WITH BORROW
05C4	3F	1542	CMC ; COMPLIMENT CARRY FOR CORRECT CARRY BIT VALUE
		1543	HIL05:
05C5	78	1544	MOV A,B ; RESTORE A
05C6	CJ	1545	POP B ; RESTORE BC
05C7	C9	1546	RET ; EXIT
		1547	;
		1548	;
		1549	*****
		1550	;
		1551	;
		1552	FUNCTION* LEAD
		1553	INPUTS* NONE
		1554	OUTPUTS* NONE
		1555	CALLS* PO
		1556	DESTROYS* B,C,F/F'S
		1557	DESCRIPTION* LEAD OUTPUTS 60 NULL CHARACTERS TO PAPER TAPE TO FORM A
		1558	LEADER.
		1559	;
		1560	LEAD:
05C8	063C	1561	MVI B,60 ; LOAD B WITH A COUNT OF 60
		1562	LE05:
05CA	0E00	1563	MVI C,0 ;
05CC	CDEB04	1564	CALL PO ; PUNCH NULL CHARACTER
05CF	05	1565	DCR B ; DECREMENT COUNT
05D0	C2CA05	1566	JNZ LE05 ; DO IT AGAIN IF NOT DONE
05D3	C9	1567	RET ;
		1568	;
		1569	;
		1570	*****
		1571	;
		1572	;
		1573	FUNCTION* NMOUT
		1574	INPUTS* A - 8 BIT INTEGER
		1575	OUTPUTS* NONE
		1576	CALLS* ECHO,PRVAL
		1577	DESTROYS* A,B,C,F/F'S
		1578	DESCRIPTION* NMOUT CONVERTS THE 8 BIT, UNSIGNED INTEGER IN THE
		1579	A REGISTER INTO 2 ASCII CHARACTERS. THE ASCII CHARACTERS
		1580	ARE THE ONES REPRESENTING THE 8 BITS. THESE TWO
		1581	CHARACTERS ARE SENT TO THE CONSOLE AT THE CURRENT PRINT
		1582	POSITION OF THE CONSOLE.
		1583	;
		1584	NMOUT:
05D4	F5	1585	PUSH PSW ; SAVE ARGUMENT
05D5	0F	1586	RRC ;
05D6	0F	1587	RRC ;
05D7	0F	1588	RRC ;

LOC	OBJ	SEQ	SOURCE STATEMENT
05D8	0F	1589	RRC ; GET UPPER 4 BITS TO LOW 4 BIT POSITIONS
05D9	CD5606	1590	CALL PRVAL ; CONVERT LOWER 4 BITS TO ASCII
05DC	CD0705	1591	CALL ECHO ; SEND TO TERMINAL
05DF	F1	1592	POP PSW ; GET BACK ARGUMENT
05E0	CD5606	1593	CALL PRVAL ;
05E3	CD0705	1594	CALL ECHO ;
05E6	C9	1595	RET ;
		1596 ;	
		1597 ;	
		1598 ; *****	
		1599 ;	
		1600 ;	
		1601 ; FUNCTION* NXTIN	
		1602 ; INPUTS* NONE	
		1603 ; OUTPUTS* NONE	
		1604 ; CALLS* ECHO,NMOUT,CROUT	
		1605 ; DESTROYS* A,F/F'S,C,D,H,L	
		1606 ; DESCRIPTION* NXTIN PRINTS 3 BYTES OF NEXT INSTRUCTION ON THE CONSOLE	
		1607 ;	
		1608 ;	
		1609 NXTIN:	
05E7	0E4E	1610	MVI C,'N' ; OUTPUT 'NI='
05E9	CD0705	1611	CALL ECHO ;
05EC	0E49	1612	MVI C,'I' ;
05EE	CD0705	1613	CALL ECHO ;
05F1	0E3D	1614	MVI C,'=' ;
05F3	CD0705	1615	CALL ECHO ;
05F6	1603	1616	MVI D,3 ; OUTPUT 3 BYTES
05F8	2ABF7F	1617	LHLD PSAVE ; GET LAST PC
		1618 NXT05:	
05FB	7E	1619	MOV A,M ;
05FC	CDD405	1620	CALL NMOUT ; OUTPUT BYTE
05FF	0E20	1621	MVI C,' ' ; USE SPACE FOR DELIMITER
0601	CD0705	1622	CALL ECHO ;
0604	15	1623	DCR D ; DECREMENT COUNT
0605	23	1624	INX H ; INCREMENT PC ADDRESS
0606	C2FB05	1625	JNZ NXT05 ; DO NEXT BYTE
0609	CDF604	1626	CALL CROUT ;
060C	C9	1627	RET ; RETURN
		1628 ;	
		1629 ;	
		1630 ; *****	
		1631 ;	
		1632 ;	
		1633 ; FUNCTION* PADR	
		1634 ; INPUTS* HL - ADDRESS TO BE PUNCHED	
		1635 ; OUTPUTS* NONE	
		1636 ; CALLS* PBYTE	
		1637 ; DESTROYS* A	
		1638 ; DESCRIPTION* PADR PUNCHES ON THE TELETYPEWRITER THE ADDRESS	
		1639 ; CONTAINED IN THE H,L REGISTERS.	
		1640 ;	
		1641 PADR:	
060D	7C	1642	MOV A,H ; PUNCH FIRST HALF OF ADDRESS
060E	CD1606	1643	CALL PBYTE ;

LOC	OBJ	SEQ	SOURCE STATEMENT
0611	7D	1644	MOV A,L ; PUNCH SECOND HALF OF ADDRESS
0612	CD1606	1645	CALL PBYTE ;
0615	C9	1646	RET ; RETURN TO CALLING ROUTINE
		1647	;
		1648	;
		1649	*****
		1650	;
		1651	;
		1652	; FUNCTION* PBYTE
		1653	; INPUTS* A - CHARACTER TO BE PUNCHED
		1654	; D - CURRENT VALUE OF CHECKSUM
		1655	; OUTPUTS* D - UPDATED VALUE OF CHECKSUM
		1656	; CALLS* PRVAL,PO
		1657	; DESTROYS* A,F/F'S
		1658	; DESCRIPTION* PBYTE CONVERTS THE HEXADECIMAL VALUE IN THE A REGISTER
		1659	; INTO TWO ASCII CHARACTERS AND PUNCHES THESE CHARACTERS
		1660	; ON PAPER TAPE. THE CHECKSUM CONTAINED IN D IS UPDATED.
		1661	;
		1662	PBYTE:
0616	F5	1663	PUSH PSW ; SAVE A,F/F'S
0617	0F	1664	RRC ; POSITION UPPER 4 BITS INTO LOWER 4 BITS
0618	0F	1665	RRC ;
0619	0F	1666	RRC ;
061A	0F	1667	RRC ;
061B	CD5606	1668	CALL PRVAL ; CONVERT UPPER 4 BITS JUST ROTATED TO ASCII
061E	CDEB04	1669	CALL PO ; PUNCH CHARACTER
0621	F1	1670	POP PSW ; RESTORE A,F/F'S
0622	F5	1671	PUSH PSW ; SAVE A AGAIN
0623	CD5606	1672	CALL PRVAL ; CONVERT LOWER 4 BITS TO ASCII CHARACTER
0626	CDEB04	1673	CALL PO ; PUNCH CHARACTER
0629	F1	1674	POP PSW ; RESTORE A
062A	82	1675	ADD D ; ADD VALUE TO CHECKSUM
062B	57	1676	MOV D,A ; UPDATE D REGISTER WITH NEW CHECKSUM
062C	C9	1677	RET ; RETURN TO CALLING ROUTINE
		1678	;
		1679	;
		1680	*****
		1681	;
		1682	;
		1683	; FUNCTION* PEOF
		1684	; INPUTS* NONE
		1685	; OUTPUTS* NONE
		1686	; CALLS* PO,PBYTE,PADR,LEAD
		1687	; DESTROYS* A,C,D,H,L,F/F'S
		1688	; DESCRIPTON* PEOF PUNCHES THE END OF FILE RECORD CONSISTING OF A RECO
		1689	; MARK, A LOAD ADDRESS OF 0, THE RECORD TYPE, AND THE
		1690	; RECORD CHECKSUM.
		1691	;
		1692	PEOF:
062D	0E3A	1693	MVI C,';' ;
062F	CDEB04	1694	CALL PO ; PUNCH RECORD MARK
0632	AF	1695	XRA A ; ZERO CHECKSUM
0633	57	1696	MOV D,A ; SAVE IN D REGISTER
0634	CD1606	1697	CALL PBYTE ; PUNCH RECORD LENGTH
0637	210000	1698	LXI H,0 ; LOAD HL WITH ZERO ADDRESS

LOC	OBJ	SEQ	SOURCE STATEMENT
063A	CD0D06	1699	CALL PADR ; PUNCH IT
063D	3E01	1700	MVI A,1 ; LOAD A WITH RECORD TYPE
063F	CD1606	1701	CALL PBYTE ; PUNCH IT
0642	AF	1702	XRA A ; ZERO A
0643	92	1703	SUB D ; COMPUTE CHECKSUM
0644	CD1606	1704	CALL PBYTE ; PUNCH IT
0647	CDC805	1705	CALL LEAD ; PUNCH TRAILER
064A	C9	1706	RET ;
		1707 ;	
		1708 ;	
		1709 ;	*****
		1710 ;	
		1711 ;	
		1712 ;	FUNCTION* PEOL
		1713 ;	INPUTS* NONE
		1714 ;	OUTPUTS* NONE
		1715 ;	CALLS* PO
		1716 ;	DESTROYS* C
		1717 ;	DESCRIPTION* PEOL PUNCHES A CARRIAGE RETURN AND LINE FEED ONTO
		1718 ;	PAPER TAPE.
		1719 ;	
		1720	PEOL:
064B	0E0D	1721	MVI C,CR ;
064D	CDEB04	1722	CALL PO ; PUNCH CARRIAGE RETURN CHARACTER
0650	0E0A	1723	MVI C,LF ;
0652	CDEB04	1724	CALL PO ; PUNCH LINE FEED CHARACTER
0655	C9	1725	RET ;
		1726 ;	
		1727 ;	
		1728 ;	*****
		1729 ;	
		1730 ;	
		1731 ;	
		1732 ;	FUNCTION* PRVAL
		1733 ;	INPUTS* A - INTEGER, RANGE 0 TO F
		1734 ;	OUTPUTS* A - ASCII CHARACTER
		1735 ;	CALLS* NOTHING
		1736 ;	DESTROYS* NOTHING
		1737 ;	DESCRIPTION* PRVAL CONVERTS A NUMBER IN THE RANGE 0 TO F HEX TO
		1738 ;	THE CORRESPONDING ASCII CHARACTER, 0-9,A-F. PRVAL
		1739 ;	DOES NOT CHECK THE VALIDITY OF ITS INPUT ARGUMENT.
		1740 ;	
		1741	PRVAL:
0656	E60F	1742	ANI HCHAR ; MASK OUT UPPER 4 BITS - WANT 1 HEX CHAR
0658	C690	1743	ADI 090H ; SET UP A SO THAT A-F CAUSE A CARRY
065A	27	1744	DAA ; ADJUST CONTENTS OF A REGISTER
065B	CE40	1745	ACI 040H ; ADD IN CARRY AND ADJUST UPPER 4 BITS
065D	27	1746	DAA ; ADJUST CONTENTS OF A REGISTER AGAIN
065E	4F	1747	MOV C,A ; MOVE ASCII CHARACTER TO C
065F	C9	1748	RET ; ALL DONE
		1749 ;	
		1750 ;	*****
		1751 ;	
		1752 ;	
		1753 ;	FUNCTION* REGDS



LOC	OBJ	SEQ	SOURCE STATEMENT
		1754	; INPUTS* NONE
		1755	; OUTPUTS* NONE
		1756	; CALLS* ECHO,NMOUT,ERROR,CROUT
		1757	; DESTROYS* A,B,C,D,E,H,L,F/F'S
		1758	; DESCRIPTION* REGDS DISPLAYS THE CONTENTS OF THE REGISTER SAVE
		1759	LOCATIONS, IN FORMATTED FORM, ON THE CONSOLE. THE
		1760	DISPLAY IS DRIVEN FROM A TABLE, RTAB, WHICH CONTAINS
		1761	THE REGISTER'S PRINT SYMBOL, SAVE LOCATION ADDRESS,
		1762	AND LENGTH (8 OR 16 BITS).
		1763	;
		1764	REGDS:
0660	21D507	1765	LXI H,RTAB ; LOAD HL WITH ADDRESS OF START OF TABLE
		1766	REG05:
0663	4E	1767	MOV C,M ; GET PRINT SYMBOL OF REGISTER
0664	79	1768	MOV A,C ;
0665	B7	1769	ORA A ; TEST FOR 0 - END OF TABLE
0666	C26D06	1770	JNZ REG10 ; IF NOT END, BRANCH
0669	CDF604	1771	CALL CROUT ; ELSE, CARRIAGE RETURN/LINE FEED TO END
066C	C9	1772	RET ; /DISPLAY
		1773	REG10:
066D	CD0705	1774	CALL ECHO ; ECHO CHARACTER
0670	0E3D	1775	MVI C,'=' ;
0672	CD0705	1776	CALL ECHO ; OUTPUT EQUALS SIGN, I.E. A=
0675	23	1777	INX H ; POINT TO START OF SAVE LOCATION ADDRESS
0676	5E	1778	MOV E,M ; GET LSP OF SAVE LOCATION ADDRESS TO E
0677	167F	1779	MVI D,HREGS ; PUT MSP OF SAVE LOC ADDRESS INTO D
0679	23	1780	INX H ; POINT TO LENGTH FLAG
067A	1A	1781	LDAX D ; GET CONTENTS OF SAVE ADDRESS
067B	CDD405	1782	CALL NMOUT ; DISPLAY ON CONSOLE
067E	7E	1783	MOV A,M ; GET LENGTH FLAG
067F	B7	1784	ORA A ; SET SIGN F/F
0680	CA8806	1785	JZ REG15 ; IF 0, REGISTER IS 8 BITS
0683	1B	1786	DCX D ; ELSE, 16 BIT REGISTER SO MORE TO DISPLAY
0684	1A	1787	LDAX D ; GET LOWER 8 BITS
0685	CDD405	1788	CALL NMOUT ; DISPLAY THEM
		1789	REG15:
0688	0E20	1790	MVI C,' ' ;
068A	CD0705	1791	CALL ECHO ; OUTPUT BLANK CHARACTER
068D	23	1792	INX H ; POINT TO START OF NEXT TABLE ENTRY
068E	C36306	1793	JMP REG05 ; DO NEXT REGISTER
		1794	;
		1795	;
		1796	; *****
		1797	;
		1798	;
		1799	; FUNCTION* REGSV
		1800	; INPUTS* NONE
		1801	; OUTPUTS* NONE
		1802	; CALLS* NONE
		1803	; DESTROYS* H,SP
		1804	; DESCRIPTION* REGSV SAVES THE USER REGISTERS ON INTERRUPT
		1805	;
		1806	REGSV:
0691	22BD7F	1807	SHLD LSAVE ; SAVE HL REGISTERS
0694	E1	1808	POP H ; GET CALLING ADDRESS

LOC	OBJ	SEQ	SOURCE STATEMENT
0695	E3	1809	XTHL ; EXCHANGE CALLER ADDR. WITH INT. PC
0696	22BF7F	1810	SHLD PSAVE ; ASSUME THIS IS THE LAST PROG COUNTER
0699	F5	1811	PUSH PSW ; SAVE A,F/F'S
069A	210400	1812	LXI H,4 ; SET HL TO 4 TO SAVE STACK POINTER CORRECTLY
069D	39	1813	DAD SP ; GET STACK POINTER VALUE
069E	22C17F	1814	SHLD SSAVE ; SAVE USERS STACK POINTER
06A1	F1	1815	POP PSW ; RESTORE A,F/F'S
06A2	E1	1816	POP H ; CALLERS RETURN POINT
06A3	31BD7F	1817	LXI SP,ASAVE+1 ; NEW VALUE FOR STACK POINTER
06A6	F5	1818	PUSH PSW ; SAVE THE REST OF THE REGISTERS
06A7	C5	1819	PUSH B ;
06A8	D5	1820	PUSH D ;
06A9	E9	1821	PCHL ; RETURN
		1822 ;	
		1823 ;	
		1824 ;	*****
		1825 ;	
		1826 ;	
		1827 ;	FUNCTION* RGADR
		1828 ;	INPUTS* C - CHARACTER DENOTING REGISTER
		1829 ;	OUTPUTS* BC - ADDRESS OF ENTRY IN RTAB CORRESPONDING TO REGISTER
		1830 ;	CALLS* ERROR
		1831 ;	DESTROYS* A,B,C,D,E,H,L,F/F'S
		1832 ;	DESCRIPTION* RGADR TAKES A SINGLE CHARACTER AS INPUT. THIS CHARACTER
		1833 ;	DENOTES A REGISTER. RGADR SEARCHES THE TABLE RTAB
		1834 ;	FOR A MATCH ON THE INPUT ARGUMENT. IF ONE OCCURS,
		1835 ;	RGADR RETURNS THE ADDRESS OF THE ADDRESS OF THE
		1836 ;	SAVE LOCATION CORRESPONDING TO THE REGISTER. THIS
		1837 ;	ADDRESS POINTS INTO RTAB. IF NO MATCH OCCURS, THEN
		1838 ;	THE REGISTER IDENTIFIER IS ILLEGAL AND CONTROL IS
		1839 ;	PASSED TO THE ERROR ROUTINE.
		1840 ;	
		1841	RGADR:
06AA	21D507	1842	LXI H,RTAB ; HL GETS ADDRESS OF TABLE START
06AD	110300	1843	LXI D,RTABS ; DE GET SIZE OF A TABLE ENTRY
		1844	RGA05:
06B0	7E	1845	MOV A,M ; GET REGISTER IDENTIFIER
06B1	B7	1846	ORA A ; CHECK FOR TABLE END (IDENTIFIER IS 0)
06B2	CA2505	1847	JZ ERROR ; IF AT END OF TABLE, ARGUMENT IS ILLEGAL
06B5	B9	1848	CMP C ; ELSE, COMPARE TABLE ENTRY AND ARGUMENT
06B6	CABD06	1849	JZ RGA10 ; IF EQUAL, WE'VE FOUND WHAT WE'RE LOOKING FOR
06B9	19	1850	DAD D ; ELSE, INCREMENT TABLE POINTER TO NEXT ENTRY
06BA	C3B006	1851	JMP RGA05 ; TRY AGAIN
		1852	RGA10:
06BD	23	1853	INX H ; IF A MATCH, INCREMENT TABLE POINTER TO
06BE	44	1854	MOV B,H ; /SAVE LOCATION ADDRESS
06BF	4D	1855	MOV C,L ; RETURN THIS VALUE
06C0	C9	1856	RET ;
		1857 ;	
		1858 ;	
		1859 ;	*****
		1860 ;	
		1861 ;	
		1862 ;	FUNCTION* RI
		1863 ;	INPUTS* NONE

LOC	OBJ	SEQ	SOURCE STATEMENT
		1864	; OUTPUTS* A - ZERO, CARRY - 1 IF END OF FILE
		1865	; A - CHARACTER, CARRY - 0 IF VALID CHARACTER
		1866	; CALLS* DELAY
		1867	; DESTROYS* A,F/F'S
		1868	; DESCRIPTION* RI READS A CHARACTER FROM THE TTY TAPE READER.
		1869	;
		1870	RI:
06C1	C5	1871	PUSH B ; SAVE BC
		1872	RI05:
06C2	DBED	1873	IN CNCTL ; READ IN USART STATUS
06C4	E604	1874	ANI TXBE ; CHECK FOR TRANSMITTER BUFFER EMPTY
06C6	CAC205	1875	JZ RI05 ; TRY AGAIN IF NOT EMPTY
06C9	3E37	1876	MVI A,TTYADV ; ADVANCE THE TAPE
06CB	D3ED	1877	OUT CNCTL ; OUTPUT THE ADVANCE COMMAND
06CD	0628	1878	MVI B,40 ; INITIALIZE TIMER FOR 40 MS.
		1879	RI07:
06CF	CDFC04	1880	CALL DELAY ; DELAY FOR 1 MILLISECOND
06D2	05	1881	DCR B ; DECREMENT TIMER
06D3	C2CF06	1882	JNZ RI07 ; JUMP IF TIMER NOT EXPIRED
06D6	3E35	1883	MVI A,TTYSTP ; STOP THE READER COMMAND
06D8	D3ED	1884	OUT CNCTL ; OUTPUT STOP COMMAND
06DA	06FA	1885	MVI B,250 ; INITIALIZE TIMER FOR 250 MS.
		1886	RI10:
06DC	DBED	1887	IN CONST ; INPUT READER STATUS
06DE	E602	1888	ANI RBR ; CHECK FOR RECEIVER BUFFER READY
06E0	C2EE06	1889	JNZ RI15 ; YES - DATA IS READY
06E3	CDFC04	1890	CALL DELAY ; DELAY 1 MS
06E6	05	1891	DCR B ; DECREMENT TIMER
06E7	C2DC06	1892	JNZ RI10 ; JUMP IF TIMER NOT EXPIRED
06EA	AF	1893	XRA A ; ZERO A
06EB	37	1894	STC ; SET CARRY INDICATING EOF
06EC	C1	1895	POP B ; RESTORE BC
06ED	C9	1896	RET ; RETURN TO CALLING ROUTINE
		1897	RI15:
06EE	DBEC	1898	IN CNIN ; INPUT DATA CHARACTER
06F0	B7	1899	ORA A ; CLEAR CARRY
06F1	C1	1900	POP B ; RESTORE BC
06F2	C9	1901	RET ; RETURN TO CALLING ROUTINE
		1902	;
		1903	; *****
		1904	;
		1905	; FUNCTION* RICH
		1906	; INPUTS* NONE
		1907	; OUTPUTS* A - ZERO, CARRY - 1 IF END OF FILE
		1908	; A - CHARACTER, CARRY - 0 IF VALID CHARACTER
		1909	; CALLS* RI
		1910	; DESTROYS* A,F/F'S
		1911	; DESCRIPTION* RICH TESTS FOR AN END OF FILE CONDITION.
		1912	;
		1913	RICH:
06F3	CDC106	1914	CALL RI ; READ A CHARACTER FROM TAPE
06F6	DA2505	1915	JC ERROR ; JUMP IF READER TIMEOUT ERROR
06F9	E67F	1916	ANI PRTY0 ; REMOVE PARITY BIT
06FB	C9	1917	RET ; RETURN TO CALLING ROUTINE
		1918	;

LOC	OBJ	SEQ	SOURCE STATEMENT
		1919	;
		1920	; *****
		1921	;
		1922	;
		1923	; FUNCTION* RSTTF
		1924	; INPUTS* A = 0, NOT SINGLE STEP EXECUTION
		1925	; A = 0FFH, SINGLE STEP EXECUTION
		1926	; OUTPUTS* NONE
		1927	; CALLS* NOTHING
		1928	; DESTROYS* A,B,C,D,E,H,L,F/F'S
		1929	; DESCRIPTION* RSTTF RESTORES ALL CPU REGISTER, FLIP/FLOPS, STACK
		1930	POINTER AND PROGRAM COUNTER FROM THEIR RESPECTIVE
		1931	SAVE LOCATIONS IN MEMORY. THE ROUTINE THEN TRANSFERS
		1932	CONTROL TO THE LOCATION SPECIFIED BY THE PROGRAM
		1933	COUNTER (I.E. THE RESTORED VALUE). THE ROUTINE
		1934	EXITS WITH THE INTERRUPTS ENABLED.
		1935	;
		1936	RSTTF:
06FC	F3	1937	DI ; DISABLE INTERRUPTS
06FD	31B77F	1938	LXI SP,MSTAK ; SET MONITOR STACK POINTER TO START
		1939	; /OF STACK
0700	D1	1940	POP D ; START ALSO END OF REGISTER SAVE AREA
0701	C1	1941	POP B ;
0702	2AC17F	1942	LHLD SSAVE ; RESTORE USER STACK POINTER
0705	F9	1943	SPHL ;
0706	2ABF7F	1944	LHLD PSAVE
0709	E5	1945	PUSH H
070A	2ABB7F	1946	LHLD FSAVE ; GET A,F/F'S
070D	E5	1947	PUSH H ; SAVE THEM
070E	2ABD7F	1948	LHLD LSAVE ; RESTORE HL REG
0711	A7	1949	ANA A ; CHECK FOR SINGLE STEP
0712	CA2307	1950	JZ RST05 ; NO, DONE
0715	3E54	1951	MVI A,STM1 ; STOP TIMER
0717	D3DF	1952	OUT T MCP ; SEND COMMAND
0719	3E10	1953	MVI A,TMRST ; RESET 7.5 INTERRUPT
071B	30	1954	SIM ; SEND IT
071C	3E0F	1955	MVI A,NEXCT ; COUNT WILL INTERRUPT ON NEXT INSTRUCTION
071E	D3DD	1956	OUT CTRL ; SET TIME VALUE
0720	3E0B	1957	MVI A, TMENB ; ENABLE 7.5
0722	30	1958	SIM ; NOW IT STARTS
		1959	RST05:
0723	F1	1960	PCP PSW ; RESTORE A,F/F'S
0724	FB	1961	EI ; ENABLE ALL (5) INTERRUPTS.
0725	C9	1962	RET ; JUMP TO RESTORED PC LOCATION
		1963	;
		1964	;
		1965	; *****
		1966	;
		1967	;
		1968	; FUNCTION* SRET
		1969	; INPUTS* NONE
		1970	; OUTPUTS* CARRY = 1
		1971	; CALLS* NOTHING
		1972	; DESTROYS* CARRY
		1973	; DESCRIPTION* SRET IS JUMPED TO BY ROUTINES WISHING TO RETURN SUCCESS.

LOC	OBJ	SEQ	SOURCE STATEMENT
		1974 ;	SRET SETS THE CARRY TRUE AND THEN RETURNS TO THE
		1975 ;	CALLER OF THE ROUTINE INVOKING SRET.
		1976 ;	
		1977 SRET:	
0726	37	1978 STC	; SET CARRY TRUE
0727	C9	1979 RET	; RETURN APPROPRIATELY
		1980 ;	
		1981 ;	
		1982 ;	*****
		1983 ;	
		1984 ;	
		1985 ;	FUNCTION* STHF0
		1986 ;	INPUTS* DE - 16 BIT ADDRESS OF BYTE TO BE STORED INTO
		1987 ;	OUTPUTS* NONE
		1988 ;	CALLS* NOTHING
		1989 ;	DESTROYS* A,B,C,H,L,F/F'S
		1990 ;	DESCRIPTION* STHF0 CHECKS THE HALF BYTE FLAG IN TEMP TO SEE IF
		1991 ;	IT IS SET TO LOWER. IF SO, STHF0 STORES A 0 TO
		1992 ;	PAD OUT THE LOWER HALF OF THE ADDRESSED BYTE
		1993 ;	OTHERWISE, THE ROUTINE TAKES NO ACTION.
		1994 ;	
		1995 STHF0:	
0728	3AC37F	1996 LDA TEMP	; GET HALF BYTE FLAG
072B	B7	1997 ORA A	; SET F/F'S
072C	C0	1998 RNZ	; IF SET TO UPPER, DON'T DO ANYTHING
072D	0E00	1999 MVI C,0	; ELSE, WANT TO STORE THE VALUE 0
072F	CD3307	2000 CALL STHLF	; DO IT
0732	C9	2001 RET	
		2002 ;	
		2003 ;	
		2004 ;	*****
		2005 ;	
		2006 ;	
		2007 ;	FUNCTION* STHLF
		2008 ;	INPUTS* C - 4 BIT VALUE TO BE STORED IN HALF BYTE
		2009 ;	DE - 16 BIT ADDRESS OF BYTE TO BE STORED INTO
		2010 ;	OUTPUTS* NONE
		2011 ;	CALLS* NOTHING
		2012 ;	DESTROYS* A,B,C,H,L,F/F'S
		2013 ;	DESCRIPTION* STHLF TAKES THE 4 BIT VALUE IN C AND STORES IT IN
		2014 ;	HALF OF THE BYTE ADDRESSED BY REGISTERS DE. THE
		2015 ;	HALF BYTE USED (EITHER UPPER OR LOWER) IS DENOTED
		2016 ;	BY THE VALUE OF THE FLAG IN TEMP. STHLF ASSUMES
		2017 ;	THAT THIS FLAG HAS BEEN PREVIOUSLY SET
		2018 ;	(NOMINALLY BY ICMD).
		2019 ;	
		2020 STHLF:	
0733	D5	2021 PUSH D	
0734	E1	2022 POP H	; MOVE ADDRESS OF BYTE INTO HL
0735	79	2023 MOV A,C	; GET VALUE
0736	E60F	2024 ANI LNIB	; FORCE TO 4 BIT LENGTH
0738	4F	2025 MOV C,A	; PUT VALUE BACK
0739	3AC37F	2026 LDA TEMP	; GET HALF BYTE FLAG
073C	B7	2027 ORA A	; CHECK FOR LOWER HALF
073D	C24607	2028 JNZ STH05	; BRANCH IF NOT

LOC	OBJ	SEQ	SOURCE STATEMENT
0740	7E	2029	MOV A,M ; ELSE, GET BYTE
0741	E6F0	2030	ANI UNIB ; CLEAR LOWER 4 BITS
0743	B1	2031	ORA C ; OR IN VALUE
0744	77	2032	MOV M,A ; PUT BYTE BACK
0745	C9	2033	RET ;
		2034	STH05:
0746	7E	2035	MOV A,M ; IF UPPER HALF, GET BYTE
0747	E60F	2036	ANI LNIB ; CLEAR UPPER 4 BITS
0749	47	2037	MOV B,A ; SAVE BYTE IN B
074A	79	2038	MOV A,C ; GET VALUE
074B	0F	2039	RRC ;
074C	0F	2040	RRC ;
074D	0F	2041	RRC ;
074E	0F	2042	RRC ; ALIGN TO UPPER 4 BITS
074F	B0	2043	ORA B ; OR IN ORIGINAL LOWER 4 BITS
0750	77	2044	MOV M,A ; PUT NEW CONFIGURATION BACK
0751	C9	2045	RET ;
		2046 ;	
		2047 ;	
		2048 ;	*****
		2049 ;	
		2050 ;	
		2051 ;	FUNCTION* VALDG
		2052 ;	INPUTS* C - ASCII CHARACTER
		2053 ;	OUTPUTS* CARRY - 1 IF CHARACTER REPRESENTS VALID HEX DIGIT
		2054 ;	- 0 OTHERWISE
		2055 ;	CALLS* NOTHING
		2056 ;	DESTROYS* A,F/F'S
		2057 ;	DESCRIPTION* VALDG RETURNS SUCCESS IF ITS INPUT ARGUMENT IS
		2058 ;	AN ASCII CHARACTER REPRESENTING A VALID HEX DIGIT
		2059 ;	(0-9,A-F), AND FAILURE OTHERWISE.
		2060 ;	
		2061	VALDG:
0752	79	2062	MOV A,C ;
0753	FE30	2063	CPI '0' ; TEST CHARACTER AGAINST '0'
0755	FA3305	2064	JM FRET ; IF ASCII CODE LESS, CANNOT BE VALID DIGIT
0758	FE39	2065	CPI '9' ; ELSE, SEE IF IN RANGE '0'-'9'
075A	FA2607	2066	JM SRET ; CODE BETWEEN '0' AND '9'
075D	CA2607	2067	JZ SRET ; CODE EQUAL '9'
0760	FE41	2068	CPI 'A' ; NOT A DIGIT - TRY FOR A LETTER
0762	FA3305	2069	JM FRET ; NO - CODE BETWEEN '9' AND 'A'
0765	FE47	2070	CPI 'G' ;
0767	F23305	2071	JP FRET ; NO - CODE GREATER THAN 'F'
076A	C32607	2072	JMP SRET ; OKAY - CODE IS 'A' TO 'F', INCLUSIVE
		2073 ;	
		2074 ;	
		2075 ;	*****
		2076 ;	
		2077 ;	
		2078 ;	FUNCTION* VALDL
		2079 ;	INPUTS* C - CHARACTER
		2080 ;	OUTPUTS* CARRY - 1 IF INPUT ARGUMENT VALID DELIMITER
		2081 ;	- 0 OTHERWISE
		2082 ;	CALLS* NOTHING
		2083 ;	DESTROYS* A,F/F'S

LOC	OBJ	SEQ	SOURCE STATEMENT
		2084	; DESCRIPTION* VALDL RETURNS SUCCESS IF ITS INPUT ARGUMENT IS A VALID
		2085	; DELIMITER CHARACTER (SPACE, COMMA, CARRIAGE RETURN,
		2086	; AND LINE FEED) FAILURE OTHERWISE.
		2087	;
		2088	VALDL:
076D	79	2089	MOV A,C ;
076E	FE2C	2090	CPI ',' ; CHECK FOR COMMA
0770	CA2607	2091	JZ SRET ;
0773	FE0D	2092	CPI CR ; CHECK FOR CARRIAGE RETURN
0775	CA2607	2093	JZ SRET ;
0778	FE0A	2094	CPI LF ; CHECK FOR LINE FEED
077A	CA2607	2095	JZ SRET ;
077D	FE20	2096	CPI ' ' ; CHECK FOR SPACE
077F	CA2607	2097	JZ SRET ;
0782	C33305	2098	JMP FRET ; ERROR IF NONE OF THE ABOVE
		2099	;
		2100	;
		2101	; *****
		2102	;
		2103	;
		2104	;
		2105	MONITOR TABLES
		2106	;
		2107	; *****
		2108	;
		2109	;
		2110	SGNON: ; SIGNON MESSAGE
0785	0D	2111	DB CR,'80/30 MONITOR, V1.2',CR
0786	38302F33		
078A	30204D4F		
078E	4E49544F		
0792	522C2056		
0796	312E32		
0799	0D		
0015		2112	LSGNON EQU \$-SGNON ; LENGTH OF SIGNON MESSAGE
		2113	;
		2114	JPTB:
079A	0000	2115	DW 0 ; BREAK 1 ADDRESS SAVE
079C	0000	2116	DW 0 ; BREAK 2 ADDRESS SAVE
079E	0800	2117	DW GO ; RST 2-7 SERVICE ROUTINE
07A0	E003	2118	DW INTIN ; JUMP TO SERVICE ROUTINE FOR TRAP
07A2	5904	2119	DW STEPIN ; --- FOR LEVEL 7.5
07A4	E003	2120	DW INTIN ; JUMP TO SERVICE ROUTINE FOR LEVEL 6.5
07A6	E003	2121	DW INTIN ; --- FOR LEVEL 5.5
07A8	0104	2122	DW INTIN9 ; --- FOR LEVEL 0
07AA	0104	2123	DW INTIN9 ; --- FOR LEVEL 1
07AC	0104	2124	DW INTIN9 ; --- FOR LEVEL 2
07AE	0104	2125	DW INTIN9 ; --- FOR LEVEL 3
07B0	0104	2126	DW INTIN9 ; --- FOR LEVEL 4
07B2	0104	2127	DW INTIN9 ; --- FOR LEVEL 5
07B4	0104	2128	DW INTIN9 ; --- FOR LEVEL 6
07B6	0104	2129	DW INTIN9 ; --- FOR LEVEL 7
		2130	;
000F		2131	JPLG EQU (\$-JPTB)/2
		2132	;

LOC	OBJ	SEQ	SOURCE STATEMENT	
		2133	;	
		2134	CADR:	; TABLE OF ADDRESSES OF COMMAND ROUTINES
07B8	0000	2135	DW	0 ; DUMMY
07BA	8502	2136	DW	NCMD ;
07BC	6B03	2137	DW	XCMD ;
07BE	C802	2138	DW	SCMD ;
07C0	6502	2139	DW	MCMD ;
07C2	1F02	2140	DW	ICMD ;
07C4	AB01	2141	DW	GCMD ;
07C6	7F01	2142	DW	DCMD ;
07C8	8D02	2143	DW	RCMD ;
07CA	1603	2144	DW	WCMD ;
		2145	;	
		2145	CTAB:	; TABLE OF VALID COMMAND CHARACTERS
07CC	57	2147	DB	'W' ;
07CD	52	2148	DB	'R' ;
07CE	44	2149	DB	'D' ;
07CF	47	2150	DB	'G' ;
07D0	49	2151	DB	'I' ;
07D1	4D	2152	DB	'M' ;
07D2	53	2153	DB	'S' ;
07D3	58	2154	DB	'X' ;
07D4	4E	2155	DB	'N' ;
0009		2156	NCMDS EQU	\$-CTAB ; NUMBER OF VALID COMMANDS
		2157	;	
		2158	;	
		2159	RTAB:	; TABLE OF REGISTER INFORMATION
07D5	41	2160	DB	'A' ; REGISTER IDENTIFIER
07D6	BC	2161	DB	LOW(ASAVE) ; ADDRESS OF REGISTER SAVE LOCATION
07D7	00	2162	DB	0 ; LENGTH FLAG - 0=8 BITS, 1=16 BITS
0003		2163	RTABS EQU	\$-RTAB ; SIZE OF AN ENTRY IN THIS TABLE
07D8	42	2164	DB	'B' ;
07D9	BA	2165	DB	LOW(BSAVE) ;
07DA	00	2166	DB	0 ;
07DB	43	2167	DB	'C' ;
07DC	B9	2168	DB	LOW(CSAVE) ;
07DD	00	2169	DB	0 ;
07DE	44	2170	DB	'D' ;
07DF	B8	2171	DB	LOW(DSAVE) ;
07E0	00	2172	DB	0 ;
07E1	45	2173	DB	'E' ;
07E2	B7	2174	DB	LOW(ESAVE) ;
07E3	00	2175	DB	0 ;
07E4	46	2176	DB	'F' ;
07E5	BB	2177	DB	LOW(FSAVE) ;
07E6	00	2178	DB	0 ;
07E7	48	2179	DB	'H' ;
07E8	BE	2180	DB	LOW(HSAVE) ;
07E9	00	2181	DB	0 ;
07EA	4C	2182	DB	'L' ;
07EB	BD	2183	DB	LOW(LSAVE) ;
07EC	00	2184	DB	0 ;
07ED	4D	2185	DB	'M' ;
07EE	BE	2186	DB	LOW(HSAVE) ;
07EF	01	2187	DB	1 ;



LOC	OBJ	SEQ	SOURCE STATEMENT		
07F0	50	2188	DB	'P'	;
07F1	C0	2189	DB	LOW(PSAVE+1)	;
07F2	01	2190	DB	1	;
07F3	53	2191	DB	'S'	;
07F4	C2	2192	DB	LOW(SSAVE+1)	;
07F5	01	2193	DB	1	;
07F6	00	2194	DB	0	; END OF TABLE MARKERS
07F7	00	2195	DB	0	
		2196		;	
		2197		;	
		2198		*****	
		2199		;	
		2200		;	
7FB7		2201	ORG	REGS	; ORG TO REGISTER SAVE - STACK GOES IN HERE
		2202		;	
7FB7		2203	MSTAK	EQU	\$ ; START OF MONITOR STACK
		2204	ESAVE:		
0001		2205	DS	1	; E REGISTER SAVE LOCATION
		2206	DSAVE:		
0001		2207	DS	1	; D REGISTER SAVE LOCATION
		2208	CSAVE:		
0001		2209	DS	1	; C REGISTER SAVE LOCATION
		2210	BSAVE:		
0001		2211	DS	1	; B REGISTER SAVE LOCATION
		2212	FSAVE:		
0001		2213	DS	1	; FLAGS SAVE LOCATION
		2214	ASAVE:		
0001		2215	DS	1	; A REGISTER SAVE LOCATION
		2216	LSAVE:		
0001		2217	DS	1	; L REGISTER SAVE LOCATION
		2218	HSAVE:		
0001		2219	DS	1	; H REGISTER SAVE LOCATION
		2220	PSAVE:		
0002		2221	DS	2	; PGM COUNTER SAVE LOCATION
		2222	SSAVE:		
0002		2223	DS	2	; USER STACK POINTER SAVE LOCATION
		2224	TEMP:		
0001		2225	DS	1	; TEMPORARY MONITOR CELL
		2226		;	
		2227		;	
		2228	RAMTB:		
		2229	BK1BY:		
0001		2230	DS	1	; BYTE SAVE FOR BREAK 1
		2231	BK1AD:		
0003		2232	DS	3	; ADDRESS SAVE FOR BREAK 1
		2233	BK2BY:		
0001		2234	DS	1	; BYTE SAVE FOR BREAK 2
		2235	BK2AD:		
0003		2236	DS	3	; ADDRESS SAVE FOR BREAK 2
		2237	OTHER:		
0004		2238	DS	4	; THIS WILL ALLOW USER TO CHANGE RST 2-7
		2239	TRAP:		
0004		2240	DS	4	; TRAP INTERRUPT
		2241	USINT:		
0004		2242	DS	4	; ALSO SPECIAL FOR 7.5 FOR NEXT COMMAND

LOC	OBJ	SEQ	SOURCE STATEMENT
		2243	USIN2:
0004		2244	DS 4 ; I/O INTERRUPT 6.5
		2245	USIN1:
0004		2246	DS 4 ; I/O INTERRUPT 5.5
0020		2247	DS 32 ; INTERRUPT VECTOR RAM TABLE FOR LEVELS 0-7
		2248	END

PUBLIC SYMBOLS

EXTERNAL SYMBOLS

USER SYMBOLS

ADRD A 0493	ADROUT A 049C	ASAVE A 7FBC	B110 A 00AF	B1200 A 0010	B150 A 0080	B2400 A 0008
B300 A 0040	B4800 A 0004	B600 A 0020	B9600 A 0002	BK1AD A 7FC5	BK1BY A 7FC4	BK2AD A 7FC9
BK2BY A 7FC8	BREAK A 04AA	BRS07 A 0084	BRS08 A 0097	BRS15 A 00B4	BRS20 A 00BD	BRS25 A 00C6
BRS30 A 00D2	BRS35 A 00E5	BRSEL A 0074	BSAVE A 7FBA	BYTE A 04BD	C1M0 A 0070	C2M3 A 00B6
CADR A 07B8	CH12 A 0080	CH15 A 0078	CH24 A 0078	CH30 A 0066	CH48 A 0066	CH60 A 005
CH96 A 0055	CI A 04D8	CMD A 0027	CNCTL A 00ED	CNIN A 00EC	CNOUT A 00EC	CNVBN A 04E
CO A 04EB	CONST A 00ED	CPVRT A 004C	CR A 000D	CROUT A 04F6	CSAVE A 7FB9	CTAB A 07CC
CTR1 A 00DD	CTR2 A 00DE	DATA A 8000	DCM05 A 0186	DCM10 A 0189	DCMD A 017F	DEL1 A 04FF
DELAY A 04FC	DSAVE A 7FB8	ECH05 A 0510	ECH10 A 0523	ECHO A 0507	EOIC A 0020	ERROR A 0525
ESAVE A 7FB7	ESC A 001B	EXIT A 052A	FINTN A 0418	FND20 A 0429	FNDI A 0421	FRET A 0533
FSAVE A 7FBB	GCM03 A 01B8	GCM05 A 01EA	GCM10 A 01F6	GCM20 A 01F9	GCM30 A 020D	GCM40 A 0215
GCMD A 01AB	GETCH A 0536	GETCM A 0154	GETHX A 053D	GETNM A 0571	GHX05 A 0543	GHX10 A 055B
GNM05 A 0578	GNM10 A 058D	GNM15 A 059B	GNM20 A 05A0	GNM25 A 05AB	GNM30 A 05AF	GO A 0008
GOBK1 A 042F	GOBK10 A 043B	GOBK20 A 0447	GTC05 A 0169	GTC10 A 0175	HCHAR A 000F	HIL05 A 05C5
HILO A 05B6	HREGS A 007F	HSAVE A 7FBE	ICCP A 00DA	ICM05 A 022A	ICM10 A 0251	ICM20 A 0259
ICM25 A 025F	ICMD A 021F	ICW1 A 00F6	ICW2 A 007F	IICR A 010E	IICR5 A 0110	IMASK A 0000
INTIN A 03E0	INTIN9 A 0401	INUST A 005F	INVRT A 00FF	JMCMD A 00C3	JPLG A 000F	JPTB A 079A
LE05 A 05CA	LEAD A 05C8	LF A 000A	LLow A 0040	LNIB A 000F	LOK A 0000	LOK15 A 0131
LOK20 A 0142	LSAVE A 7FBD	LSGNON A 0015	MCM05 A 026D	MCMD A 0265	MODE A 004F	MODE2 A 00CF
MSKPT A 00DB	MSTAK A 7FB7	NCMD A 0285	NCMDS A 0009	NEWLN A 000F	NEXCT A 000F	NMOUT A 05D
NXT05 A 05FB	NXTIN A 05E7	OCW3 A 000B	ONEMS A 008B	OTHER A 7FCC	PADR A 060D	PBYTE A 061
PEOF A 062D	PEOL A 064B	PO A 04EB	PRTY0 A 007F	PRVAL A 0656	PSAVE A 7FBF	RAMTB A 7FC
RBR A 0002	RCM05 A 0299	RCM10 A 02B6	RCMD A 028D	REG05 A 0663	REG10 A 066D	REG15 A 0688
REGDS A 0660	REGS A 7FB7	REGSV A 0691	RESURT A 0037	REGA05 A 06B0	RGA10 A 06BD	RGADR A 06AA
RI A 06C1	RI05 A 06C2	RI07 A 06CF	RI10 A 06DC	RI15 A 06EE	RICH A 06F3	RST05 A 0723
RST1 A 00CF	RSTTF A 06FC	RSTUST A 0040	RTAB A 07D5	RTABS A 0003	SCM03 A 02D1	SCM05 A 02D3
SCM10 A 02DE	SCM20 A 02EE	SCM25 A 030D	SCMD A 02C8	SGNON A 0785	SRET A 0726	SSAVE A 7FC1
STEPIN A 0459	STH05 A 0746	STHF0 A 0728	STHLF A 0733	STM1 A 0054	STP05 A 047F	STP10 A 0482
TEMP A 7FC3	TERM A 001B	TMCP A 00DF	TMDIS A 000F	TMENB A 000B	TMRST A 0010	TRAP A 7FD0
TRDY A 0001	TTYADV A 0037	TTYSTP A 0035	TXBE A 0004	UNIB A 00F0	UPPER A 00FF	USAREA A 7F80
USECI A 0040	USECO A 0043	USEPO A 0049	USERI A 0046	USIN1 A 7FDC	USIN2 A 7FD8	USINT A 7FD4
VALDG A 0752	VALDL A 076D	WCM05 A 0320	WCM10 A 0335	WCM15 A 0338	WCM20 A 0350	WCM25 A 0365
WCMD A 0316	XCM05 A 037D	XCM10 A 038C	XCM15 A 0399	XCM20 A 03B7	XCM25 A 03CE	XCM30 A 03CF
XCM35 A 03D7	XCMD A 036B					

ASSEMBLY COMPLETE, NO ERRORS



### REQUEST FOR READER'S COMMENTS

The Microcomputer Division Technical Publications Department attempts to provide documents that meet the needs of all Intel product users. This form lets you participate directly in the documentation process.

Please restrict your comments to the usability, accuracy, readability, organization, and completeness of this document.

1. Please specify by page any errors you found in this manual.

---

---

---

---

---

---

2. Does the document cover the information you expected or required? Please make suggestions for improvement.

---

---

---

---

---

3. Is this the right type of document for your needs? Is it at the right level? What other types of documents are needed?

---

---

---

---

---

---

4. Did you have any difficulty understanding descriptions or wording? Where?

---

---

---

---

5. Please rate this document on a scale of 1 to 10 with 10 being the best rating. \_\_\_\_\_

NAME \_\_\_\_\_ DATE \_\_\_\_\_  
TITLE \_\_\_\_\_  
COMPANY NAME/DEPARTMENT \_\_\_\_\_  
ADDRESS \_\_\_\_\_  
CITY \_\_\_\_\_ STATE \_\_\_\_\_ ZIP CODE \_\_\_\_\_

Please check here if you require a written reply.





INTEL CORPORATION, 3065 Bowers Avenue, Santa Clara, CA 95051 (408) 987-8080

Printed in U.S.A.