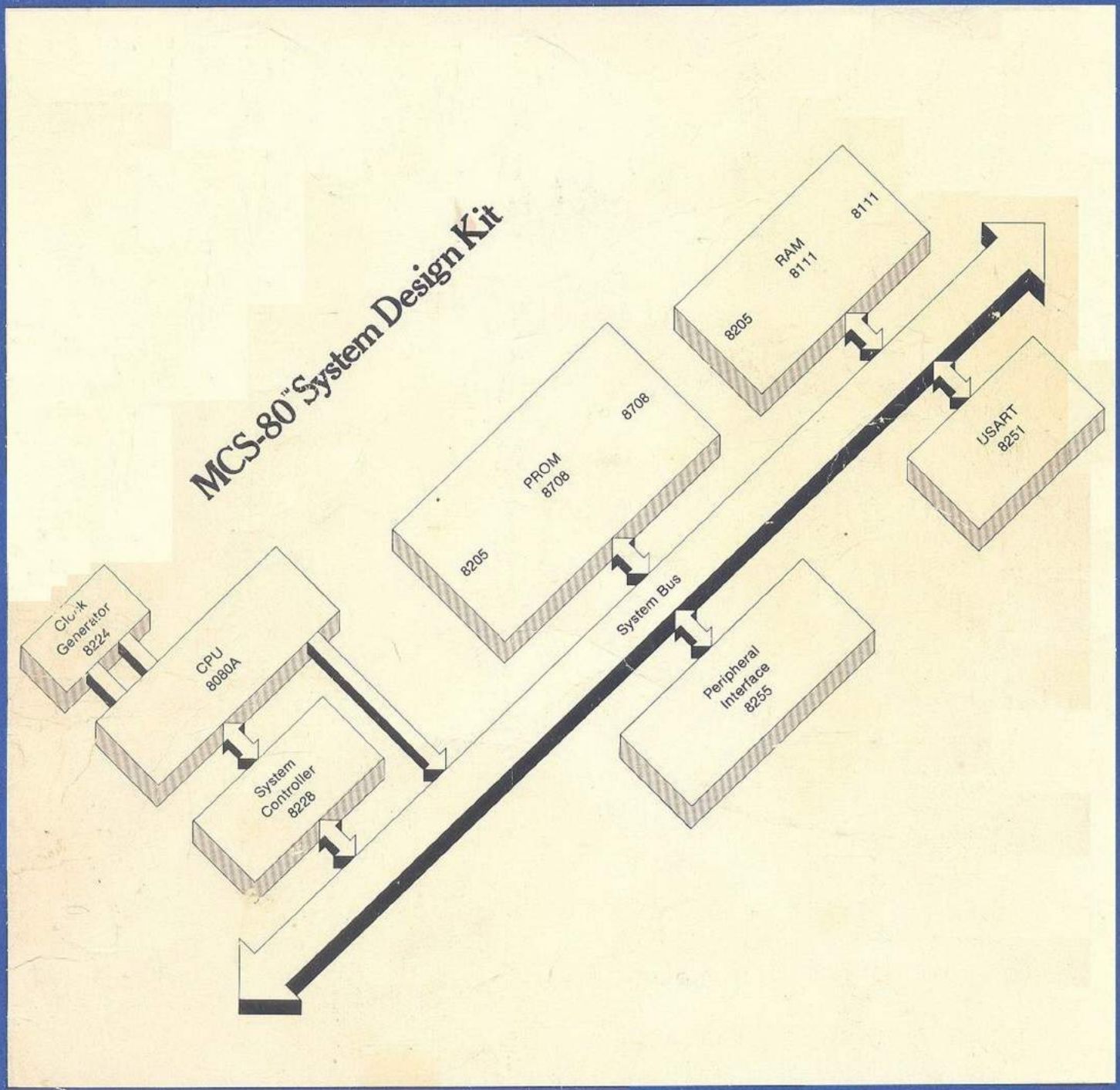
intel MCS-80™ System Design Kit User's Guide



©Intel Corporation 1976

intel® MCS-80™ System Design Kit User's Guide

CONTENTS

Data Sheet i
Specifications ii

CHAPTER 1
INTRODUCTION

General 1
Suggestions 1
Functional Definition 1
Registration Card Information 2

CHAPTER 2
HOW TO ASSEMBLE THE KIT

General 3
Tools Required 3
Optional Rotary Switch 3
Parts List 3
Assembly Procedure 6
Assembly of Area One 7
Assembly of Area Two 10
Assembly of Area Three 12

CHAPTER 3
THEORY OF OPERATION

General 15
System Busses 16
Reset Switch 16
Clock Generator and Clock Crystal 16
8080A CPU 16
System Controller 16
Address Buffers (Optional) 17
SDK-80 Memory 17
Baud Rate Generator 18
I/O Communication Interface 18
Peripheral Interfaces 18

CHAPTER 4
FINAL ASSEMBLY AND CHECKOUT

General 19
Jumper-Wiring the Board 19
Installing Integrated Circuits 21
Start-Up Procedure 23
Troubleshooting Hints 23

CHAPTER 5
SDK-80 MONITOR

Introduction 24
Monitor Operations 24

APPENDIX A

Monitor Listing 29

APPENDIX B

SDK-80 Schematics 57

APPENDIX C

Board Layout with Component Values 61

intel®

Microcomputer Systems

8080 SYSTEM DESIGN KIT (SDK-80)

- **Complete Single Board Microcomputer System Including CPU, Memory and I/O**
- **Easy to Assemble Kit-Form**
- **High-Performance (2μs Instruction Cycle)**
- **Interfaces Directly with most Terminals (75-4800 Baud)**
- **Large Wire-Wrap area for Custom Interfaces**
- **Extensive System Monitor Software in ROM**
- **PC Board Format and Power, Compatible with INTELLEC® MDS**
- **Complete MCS-80™ User's Library**

The 8080 System Design Kit (SDK-80) is a complete, single board, microcomputer system in kit form. It contains all necessary components, including resistors, caps, crystal and miscellaneous hardware to complete construction. Included is a pre-programmed ROM that contains the system monitor for general software utilities and system diagnostics.

All that is required for operation are power supplies and a suitable terminal; TTY, CRT, etc., (level conversions and baud rate generation included on board).

The SDK-80 is an inexpensive, high-performance prototype system that has designed-in flexibility for simple interface to the users application.

Intel Corporation assumes no responsibility for the use of any circuitry other than circuitry embodied in an Intel product. No other circuit patent licenses are implied.

SDK-80 SPECIFICATIONS

Central Processor

CPU: 8080A
Instruction Cycle: 1.95 microsecond
Tcy: 488 ns

Memory

ROM: 2K bytes (expandable to 4K bytes) 8708/8308
RAM: 256 bytes (expandable to 1K bytes) 8111
Addressing:
ROM 0000-0FFF
RAM 1000-13FF

Input/Output

Parallel: One 8255 for 24 lines (expandable to 48 lines).
Serial: One 8251 USART.
On-board baud rate generator (jumper selectable).

Baud Rates:		
	75	1200
	110	2400
	300	4800
	600	

Interfaces

Bus: All signals TTL compatible.
Parallel I/O: All signals TTL compatible.
Serial I/O: RS232C/EIA
20 mil A. Current loop TTY
TTL (one TTL load)

Interrupts

Single level: Generates RST7 vector TTL compatible input.

DMA

Hold Request: Jumper selectable.

Software

System Monitor: Pre-programmed 8708 or 8308 ROM Address; 0000-03FF.

Features:

Display Memory Contents	(D)
Move blocks of memory	(M)
Substitute memory locations	(S)
Insert hex code	(I)
Examine Registers	(X)
Program Control	(G)

Power-up start or system reset start.

I/O: Console Device (serial I/O)

Literature

Design Library:
8080 Users Manual
8080 Assembly Language Manual
PL/M™ Programming Manual
MDS Brochure
Reference Card (Programmers)
SDK-80 User's Guide

Connectors

I/O: 25 pin female (RS232C)
PCB: MDS format

Physical Characteristics (MDS Mechanical format)

Width: 12.0 in.
Height: 6.75 in.
Depth: 0.50 in.
Weight: approx. 12 oz.

Electrical Characteristics (DC Power)

V_{CC}	5V ±5%	1.3 Amps
V_{DD}	12V ±5%	.35 Amps
V_{BB}	-10V ±5%	.20 Amps
or	-12V ±5%	

Environmental

Operating Temperature: 0-70°C

CHAPTER 1
INTRODUCTION

GENERAL

The 8080 System Design Kit (SDK-80) is a complete microcomputer system in kit form. It is simple to assemble (construction time is 6 hours) and provides an excellent training/ prototype vehicle for evaluation of the 8080 microcomputer system (MCS-80™).

The SDK-80 is an extremely flexible design and allows easy interface to an existing application or custom interface development.

An extensive system monitor is included in a pre-programmed ROM for general software utilities and system diagnostics.

The System Design Kit User's Guide will instruct the user how to assemble his kit and configure it to match the selected terminal and peripheral devices. It is suggested that the User's Guide be followed in the exact sequence that it is written to assure successful completion of the system.

SUGGESTIONS

The 8080 Microcomputer Systems User's Manual is included with the SDK-80 and it would be extremely beneficial to the user that he read and understand the operation of the 8080A and associated peripheral components prior to beginning the assembly of the SDK-80.

Every effort has been made to allow the SDK-80 to interface directly with most common terminals but with the wide array of display terminals available it is not possible to perfectly interface each one with the SDK-80 hardware and software. The user should carefully examine the requirements of his particular terminal interface and adapt the SDK-80 accordingly.

FUNCTIONAL DEFINITION

The SDK-80 is shipped with a complement of parts that allows the user to construct an operating small system with the following features:

CPU:	8080A (see 8080 User's Manual for details) 1.95 µs Instruction Cycle
RAM:	8111 (static 256 x 4) 2 included for 256 byte storage
ROM:	8708/8308 (1K x 8) 1 Pre-programmed system monitor 1 User-programmed (erasable 8708)
I/O:	8251 (Programmable Communication Interface) 1 Serial communication with terminal
	8255 (Programmable Peripheral Interface) 1 General user I/O, 24 lines
Serial Interface:	TTL 20mA current loop (TTY) RS-232 (EIA)
Baud Rate:	User-selected by jumper or switch 75,110,300 600, 1200, 2400, 4800
Interrupt:	Single level, vectored (RST-7)

The SDK-80 has many designed-in features for expandability without the necessity of cutting PC runs or adding extra logic. The maximum configuration of the SDK-80 is as follows:

RAM:	8111 (static 256 x 4) Up to 8 for 1K x 8 storage
ROM:	8708/8308 (1K x 8) Up to 4 for 4K x 8 storage
I/O:	8255 (Programmable Peripheral Interface) 2 General user I/O, 48 lines

Expanding the SDK-80 to the maximum configuration is a simple matter of purchasing the extra memory and I/O components and installing them on the board.

REGISTRATION CARD

On the back cover of the User's Guide is the Registration card for the SDK-80. Please fill it out completely and return it to INTEL upon completion of the kit. The Registration Card assures you of being updated with the latest information on the 8080 microcomputer system and any additional updates on your 8080 System Design Kit.

SDK-80 REGISTRATION CARD

(Please Print)

Name _____

Title _____

Company _____

Address _____

City _____ State _____

Mail Stop _____ Zip Code _____

Distributor _____

Location _____

GENERAL INQUIRY

How long did it take to assemble? _____ Hours

Did it work the first time? Yes ☐ No ☐

If No, please comment: _____

_____

_____

_____

General Comments: _____

_____

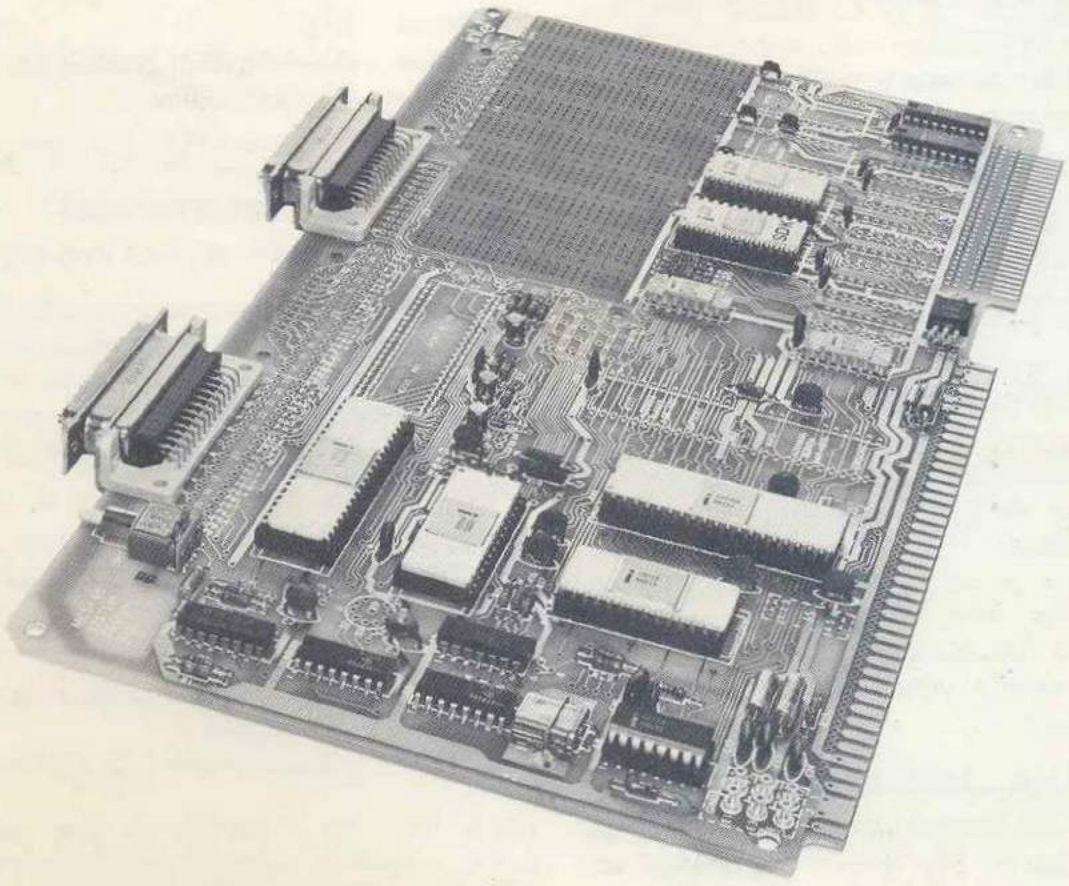
_____

_____

What make/model terminal did you use? _____

_____

Baud Rate _____



Assembled Board.

CHAPTER 2
HOW TO ASSEMBLE THE KIT

GENERAL

The MCS-80™ System Design Kit is shipped in a single cardboard carton. This chapter will take you from the point of receiving this carton to the point where you are ready to insert the IC (Integrated Circuit) chips.

Follow the instructions carefully and make a check mark in the boxes provided after you have completed each step.

Your work area should be an uncluttered, well-lighted table or desk with access to an AC wall socket or extension cord.

TOOLS REQUIRED

Before starting the project, you should make sure the proper tools are at hand and are in good operating condition. These tools will be required to assemble the MCS-80 System Design Kit:

- ☐ A pair of needle-nose pliers
- ☐ A small Phillips head screwdriver
- ☐ A 1/4" standard flat head screwdriver
- ☐ A pair of small diagonal cutters
- ☐ A 25-watt pencil-type soldering iron
- ☐ A spool of rosin core solder with 60:40 tin-lead content.

IMPORTANT
Use only rosin core solder for all electrical soldering!

- ☐ A VOM (Volt-Ohm-Milliammeter) test meter

If available, a dual-probe oscilloscope would also be helpful.

OPTIONAL ROTARY SWITCH

The Design Kit is complete for most applications; however, applications requiring multiple baud rates will need an additional rotary switch. We have allocated a position on the circuit board for this switch. One possible switch is Spectrol 87-12-19, available from Spectrol Electronics Corp., 17070 East Gale Avenue, City of Industry, CA 91744. Phone: (213) 964-6565.

PARTS LIST

With the proper tools at hand, you are now ready to take inventory of the carton.

The contents of the carton are divided into two compartments. One compartment contains the kit's documentation, the other contains the SDK-80. In addition to the User's Guide that you are now reading, the following documents are included:

- ☐ 8080 Assembly Language Programming Manual
- ☐ 8080 Assembly Language Reference Card
- ☐ 8008/8080 PL/M™ Programming Manual
- ☐ MDS Brochure
- ☐ 8080 Microcomputer Systems User's Manual

The components of the MCS-80 System Design Kit come in four packages:

- ☐ Printed wiring (PW) board, PN 1000609-01
- ☐ Miscellaneous small component bag
- ☐ SDK-80 Intel component pack

CAUTION
Do not handle the IC's until instructed to do so.

- ☐ Miscellaneous non-Intel component pack

If any of the above component packages or documents are missing, call your distributor immediately. If not, lay each of the component packages on your work table and proceed reading.

SDK-80 Intel® Component Pack

The Intel component pack contains the Intel IC's needed in the Kit. The numbers indicated in Figure 2-1 correspond to the "Item Number" in Table 2-1.

☐ Verify that all items in Table 2-1 are included. **Do not remove components from backing.**

Table 2-1. Parts List, Intel Component Pack.

Item No.	Part No.	Description	Qty.
1	52-016	IC, Intel® 8205	2
2	52-035	IC, Intel® 8251	1
3	52-045	IC, Intel® 8224	1
4	52-046	IC, Intel® 8228	1
5	52-047	IC, Intel® 8255	1
6	52-059	IC, Intel® 8708	1
7	52-058	IC, Intel® 8080A	1
8	52-062	IC, Intel® 8111	2
9	52-605	IC, Intel® SDK-80 Monitor ROM	1

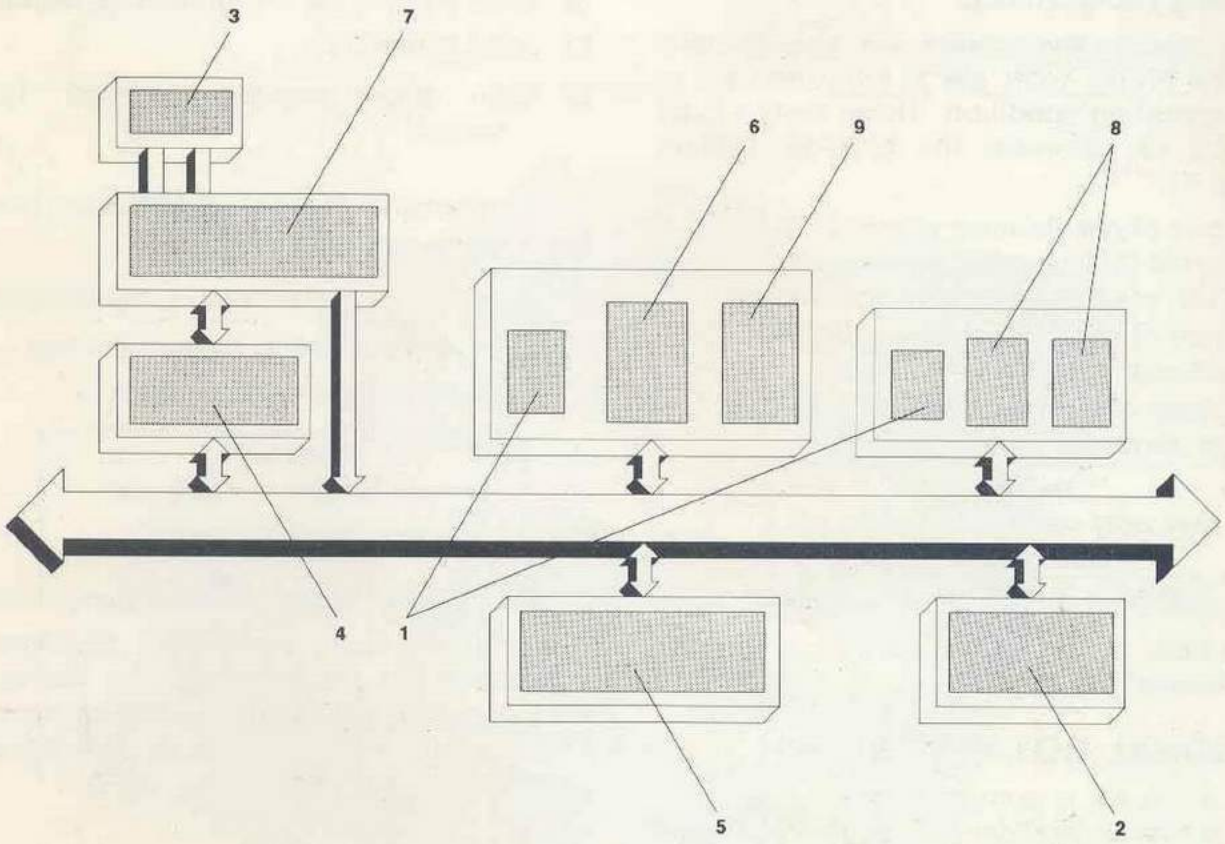


Figure 2-1. Intel® Component Pack.

Non-Intel® Component Pack

The non-Intel component pack contains miscellaneous components produced by other manufacturers. Figure 2-2 shows the arrangement of these components. The circled numbers are keyed to Table 2-2.

☐ Verify that all items in Table 2-2 are included. Do not remove components from backing.

Table 2-2. Parts List, Non-Intel Component Pack.

Item No.	Part No.	Description	Qty.
1	54-028	IC, 74161	2
2	54-092	IC, 7406	1
3	54-135	IC, 79M05AUC	1
4	54-136	IC, 93S16	1
5	68-009	Socket, 40-pin	2
6	68-007	Socket, 24-pin	2
7	68-177	Socket, 28-pin	2
8	68-102	Connect female, right angle, or 25-pin	2
9	68-077	Connect male, 25-pin	2
10	68-006	Socket, or 16-pin	1

Auxiliary Components For SDK-80 Expansion (Not Supplied With Kit)

Part No.	Description	Manufacturer	Manufacturer Part No.
66-032	Right angle push button switch SPDT	C & K	8125R2
68-102	Right angle female 25 pin connector	ITT Cannon AMP	DBC-255-AA 205858-1
68-077	Male 25 pin connector (solder)	Cinch Jones	DB-25P
——	Low profile 18 pin DIP socket	TI	C93-18-02
68-007	Low profile 24 pin DIP socket	TI	C93-24-02
68-009	Low profile 40 pin DIP socket	TI	C93-40-02
——	Small 30/60 PCB connector	CDC	97167901 (w-w) 97169001 (solder)
——	Large 43/86 PCB connector (wire-wrap)	Viking	VPB01EA3A00A-1

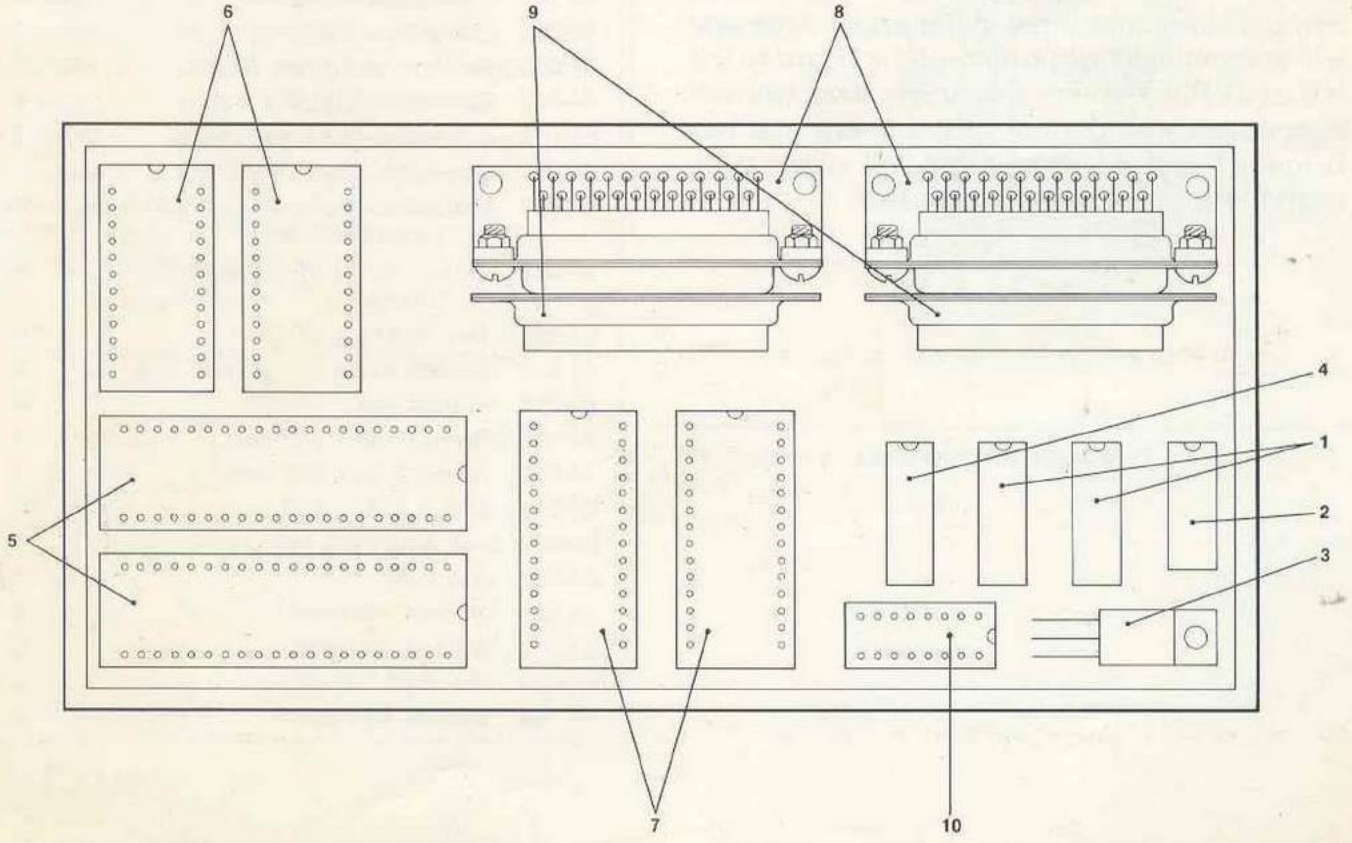


Figure 2-2. Non-Intel® Component Pack.

Small Component Bag

The small component bag contains the miscellaneous resistors, capacitors, screws, etc. needed to support the integrated circuits. Proceed as follows:

- ☐ Open the bag and spread the components in front of you.
- ☐ Separate the components into groups; i.e., resistors in one group, capacitors in another, screws and nuts and washers in another, etc.
- ☐ Verify that all parts listed in Table 2-3 are included.

ASSEMBLY PROCEDURE

Now that you have verified that all of the parts are included, you are ready to begin assembling the board.

This section of the text will be supplemented with drawings that illustrate the area of the board under construction. Components to be installed later will be shown as unshaded. Components being installed will be shown as shaded. Components that were installed earlier in the procedure will be shown blackened.

We shall assemble the board components by area, where the board may be thought of as being divided into three major areas. **Area one** will encompass that portion of the board to the left of J1 Pin 1 on the silkscreen. **Area two** will encompass that portion of the board that lies below J1 and J2. **Area three** will encompass everything to the right of area two.

Table 2-3. Parts List, Small Component Bag

Part No.	Description	Qty.
62-008	Crystal, 18.432 MHz	1
56-044	Resistor, 10 KΩ, 1/4W (brown-black-orange)	1
56-024	Resistor, 1 KΩ, 1/4W (brown-black-red)	16
56-033	Resistor, 2.7 KΩ, 1/4W (red-violet-red)	1
56-006	Resistor, 47 Ω, 1/4W (yellow-violet-black)	2
56-017	Resistor, 390 Ω, 1/4W (orange-white-brown)	1
56-106	Resistor, 430 Ω, 1W	1
56-038	Resistor, 4.7 KΩ, 1/4W (yellow-violet-red)	1
56-209	Resistor, 560 KΩ, 1/4W (green-blue-yellow)	1
56-039	Resistor, 5.1 KΩ, 1/4W (green-brown-red)	2
56-112	Resistor, 150 Ω, 1/4W (brown-green-brown)	1
56-020	Resistor, 510 Ω, 1/4W (green-brown-brown)	1
56-210	Resistor, 430 Ω, 1.5W	1
56-213	Resistor, 130 Ω, 1/4W (brown-orange-brown)	1
60-003	Diode, 1N914	2
58-006	Transistor, 2N2222	2
58-003	Transistor, 2N2907	1
66-032	Switch, right angle, SPDT	1
64-042	Capacitor, 1 uf, 50V tant	2
64-012	Capacitor, 22 uf, 15V tant	3
64-052	Capacitor, 10 pF, mica	1
64-050	Capacitor, .1 uf, 50V monolithic (bright-colored)	13
64-022	Capacitor, .01 uf, ceramic disc (orange)	16
82-015	Terminal lugs, 2010B	6
82-072	Spacer, nylon 1/4" x 7/16" o.d.	5
82-069	Rubber feet	5
84-010	Screw, 4-40 x 3/8" pan	1
84-073	Screw, 2-56 x 3/8" pan	4
84-016	Screw, 6-32 x 3/4" pan	5
84-069	Nut, 4-40 plain hex	1
84-042	Nut, 2-56	4
84-068	Washer, #2 nylon	4
84-027	Washer, #6 nylon	5
84-070	Nut, 6-32	5
84-059	Washer, #4 nylon	2

Assembly of Area One

Proceed as follows:

- ☐ Lay the PW board on your work area, silkscreened side up, so that the edge connectors are directly in front of you.
- ☐ Your first step will be to install the five rubber feet. There will be one foot in each corner and one in the middle of the board. At each of these locations, place a nylon spacer and rubber foot underneath the board and insert a screw (5-32x3/4) through them from the bottom. Then attach the screw at the top using a #6 nylon washer and a 6-32 hex nut.
- ☐ Referring to Table 2-4, solder resistors R1-R4 and R21-R24 in place. Figure 2-3 shows this area of the board.

HINT
Save all scrap resistor leads for later use as jumper wires.

Table 2-4. Construction Table #1.

Resistor	Description
R1	130, 1/4W (brown-orange-brown)
R2	1K, 1/4W (brown-black-red)
R3	1K, 1/4W (brown-black-red)
R4	560K, 1/4W (green-blue-yellow)
R21	1K, 1/4W (brown-black-red)
R22	5.1K, 1/4W (green-brown-red)
R23	1K, 1/4W (brown-black-red)
R24	1K, 1/4W (brown-black-red)

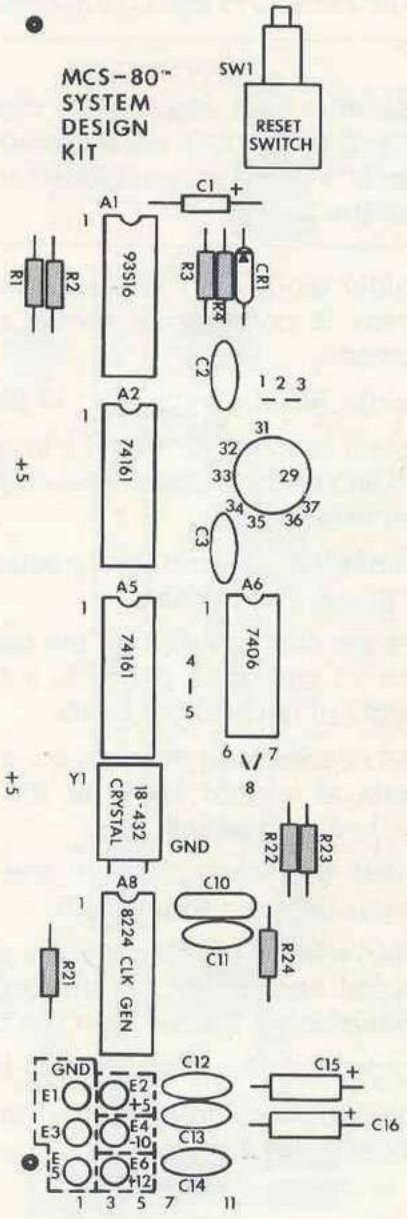


Figure 2-3. Construction Figure #1.

☐ Solder capacitors C1-C3 and C10-C16 in place (Figure 2-4). C1 is a 1 uf, 50V tant. C2, C3, and C11-C14 0.01 uf ceramic. C10 is a 10 pF mica. C15 and C16 are 22 uf, 15V tant.

IMPORTANT
Be sure that electrolytic capacitors C1, C15 and C16 are installed so that their "+" ends are positioned as shown on the board.

☐ Solder diode CR1 in place. Make sure the arrow is pointing as shown on the silk-screen.

☐ Solder Reset Switch SW1 in place.

☐ Solder terminal lugs E1-E6 in place. Figure 2-5 shows the installation completed in the last three steps.

☐ Solder the Spectrol rotary baud rate switch in place, if applicable.

☐ Set the clock crystal on the board at location Y1 and use a pencil to mark the bend points of each of the leads.

☐ Use needle-nose pliers to bend each of the leads at a right angle at the points you marked with pencil.

☐ Insert the leads through the board and solder them on the bottom.

☐ Strap a piece of scrap resistor wire over the crystal, pushing each end through one of the drilled holes. Solder from the bottom.

☐ Solder the 16-pin socket into location A8.

You have now completed area one. Compare your board with Figure 2-6.

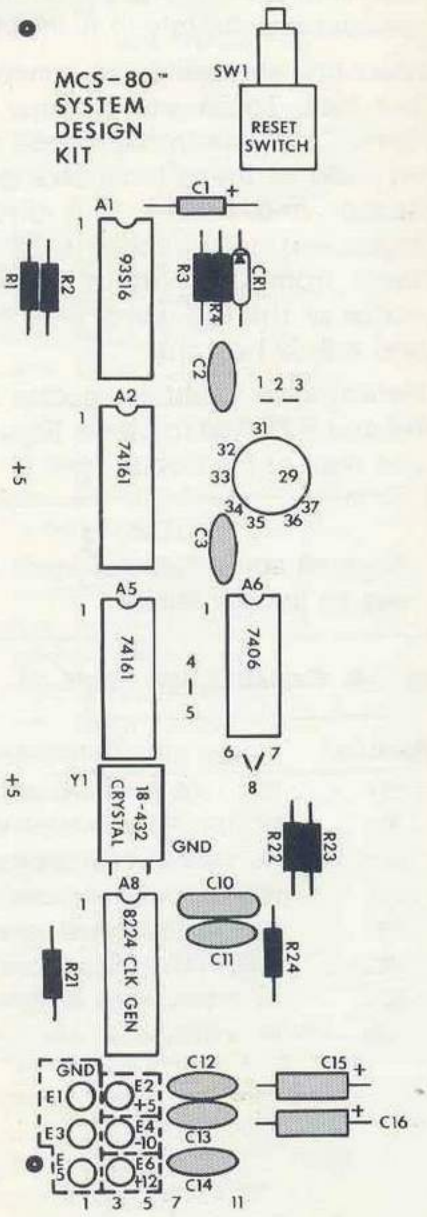


Figure 2-4. Construction Figure #2.

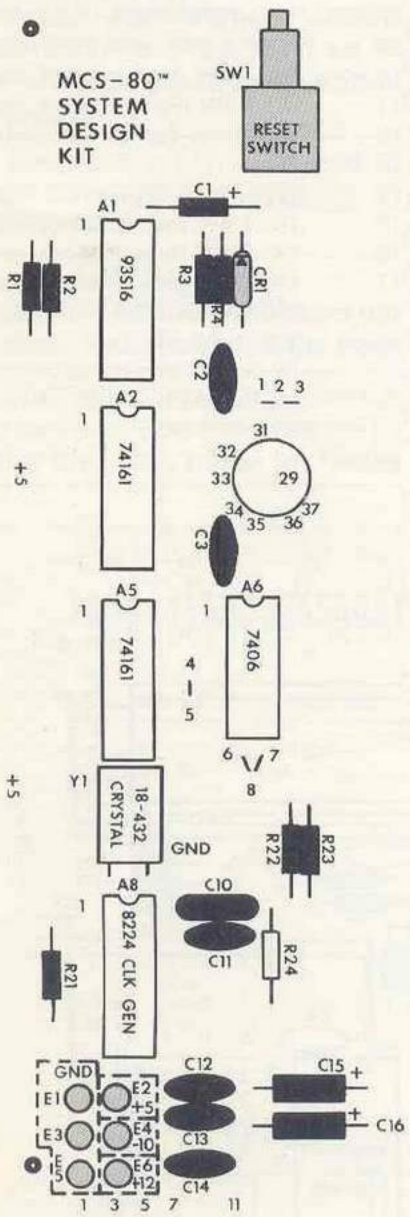


Figure 2-5. Construction Figure #3.

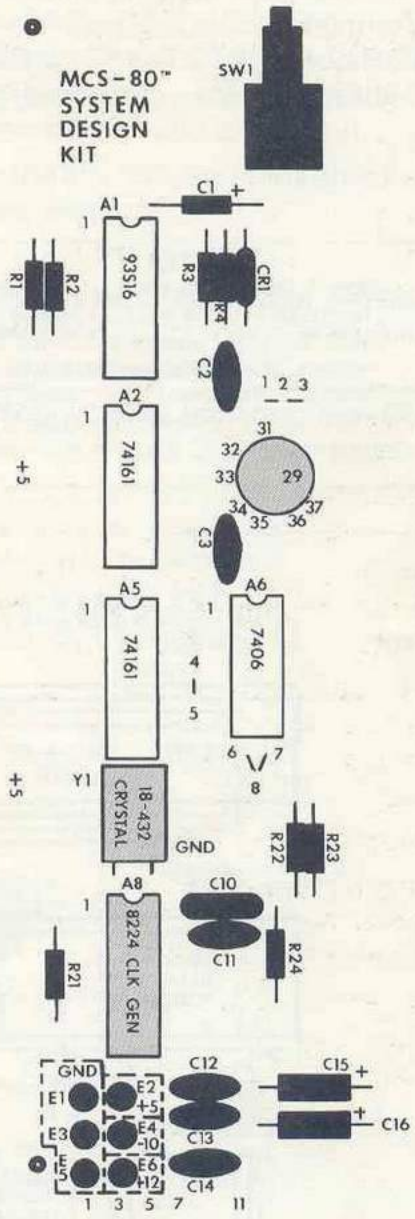


Figure 2-6. Construction Figure #4.

Assembly of Area Two

Board **area two** will contain the 8080A and its related logic. The assembly procedure is as follows:

- ☐ Referring to Table 2-5, solder resistors R5-R17, R19, R20, R25, and R26 in place. Figure 2-7 shows this area of the board.

Note 1. (From Table 2-5)
The resistor to be installed in R13 depends upon what negative voltage level your power supply delivers. For -10V, use a 390Ω, 1/4W resistor (orange-white-brown). For -12V or -15V, use a 510Ω 1/4W resistor (green-brown-brown).

Table 2-5. Construction Table #2.

Resistor	Description
R5	5.1K, 1/4W (green-brown-red)
R6	4.7K, 1/4W (yellow-violet-red)
R7	1K, 1/4W (brown-black-red)
R8	10K, 1/4W (brown-black-orange)
R9	2.7K, 1/4W (red-violet-red)
R10	1K, 1/4W (brown-black-red)
R11	47, 1/4W (yellow-violet-black)
R12	150, 1/4W (brown-green-brown)
R13	(Note 1)
R14	47, 1/4W (yellow-violet-black)
R15	1K, 1/4W (brown-black-red)
R16	1K, 1/4W (brown-black-red)
R17	1K, 1/4W (brown-black-red)
R19	430, 1W (RS-1A)
R20	430, 1.5W (G-2)
R25	1K, 1/4W (brown-black-red)
R26	1K, 1/4W (brown-black-red)

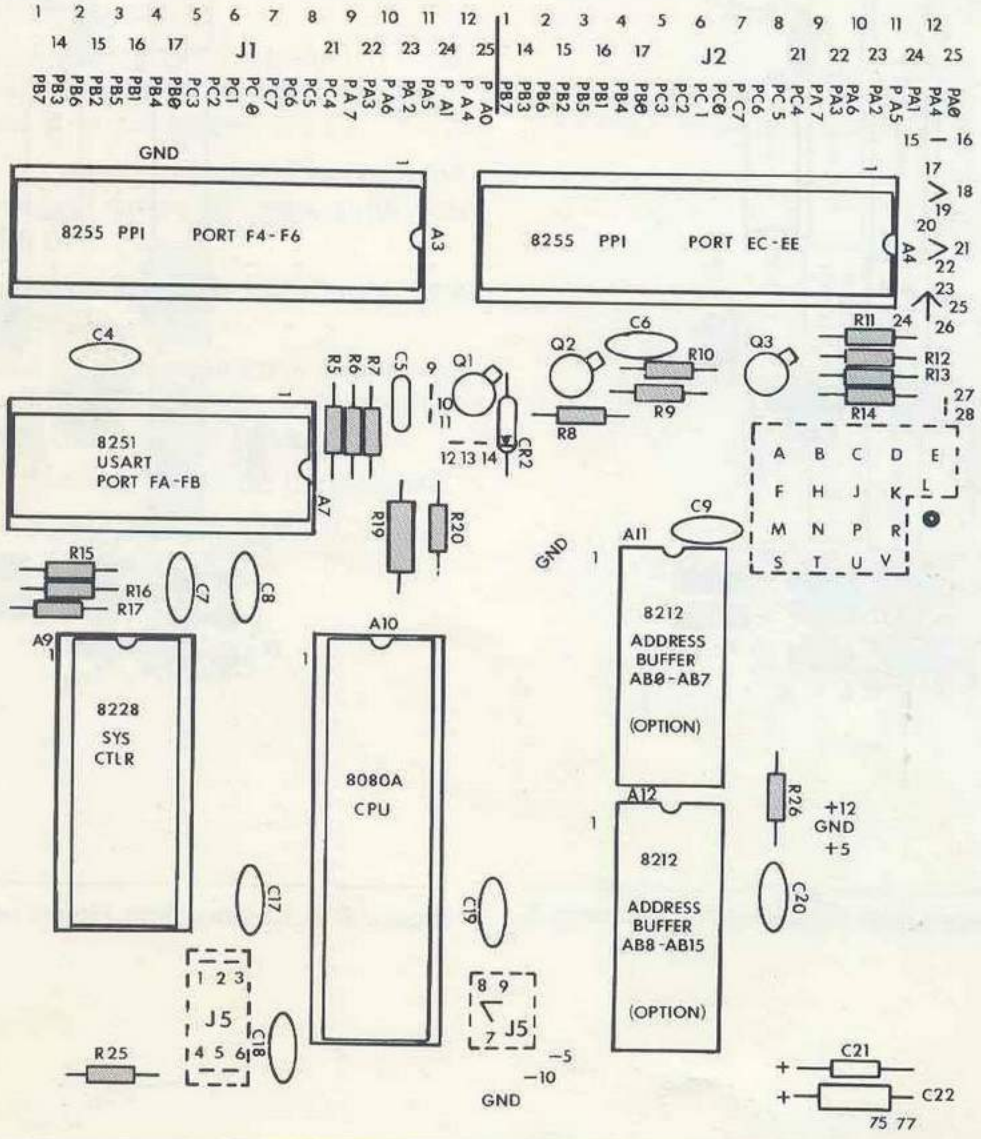


Figure 2-7. Construction Figure #5.

☐ Solder capacitors C4-C9 and C17-C22 into place. C4, C6-C9, and C17-C20 are 0.01 uf ceramic. C5 is 0.1 uf, 50V mono. C21 is 1 uf, 50V tant. C22 is 22 uf, 15V tant.

IMPORTANT

Be sure that electrolytic capacitors C21 and C22 are positioned so that their "+" ends are positioned as shown on the board.

☐ Solder transistors Q1, Q2, and Q3 into place. Q1 and Q2 are 2N2222. Q3 is a 2N2907.

IMPORTANT

The metal tabs on the transistors must be positioned as shown on the board.

☐ Solder diode CR2 into place. Make sure the arrow is as shown on the board. Figure 2-8 shows the progress made in the last three steps.

☐ Solder 40-pin sockets into locations A3 and A10.

☐ Solder 28-pin sockets into locations A7 and A9.

☐ Insert a 25-pin female connector into location J1. At each of two locations, place a 2-56x3/8 screw through the connector from the top and secure it at the bottom with a #2 nylon washer and a 2-56 nut.

☐ Solder the 25 connector pins onto the board from the bottom.

NOTE

The 25-pin male connector provided can be used to interface your hardware to connector J1.

Area two should now be complete. Compare your board with Figure 2-9 (next page).

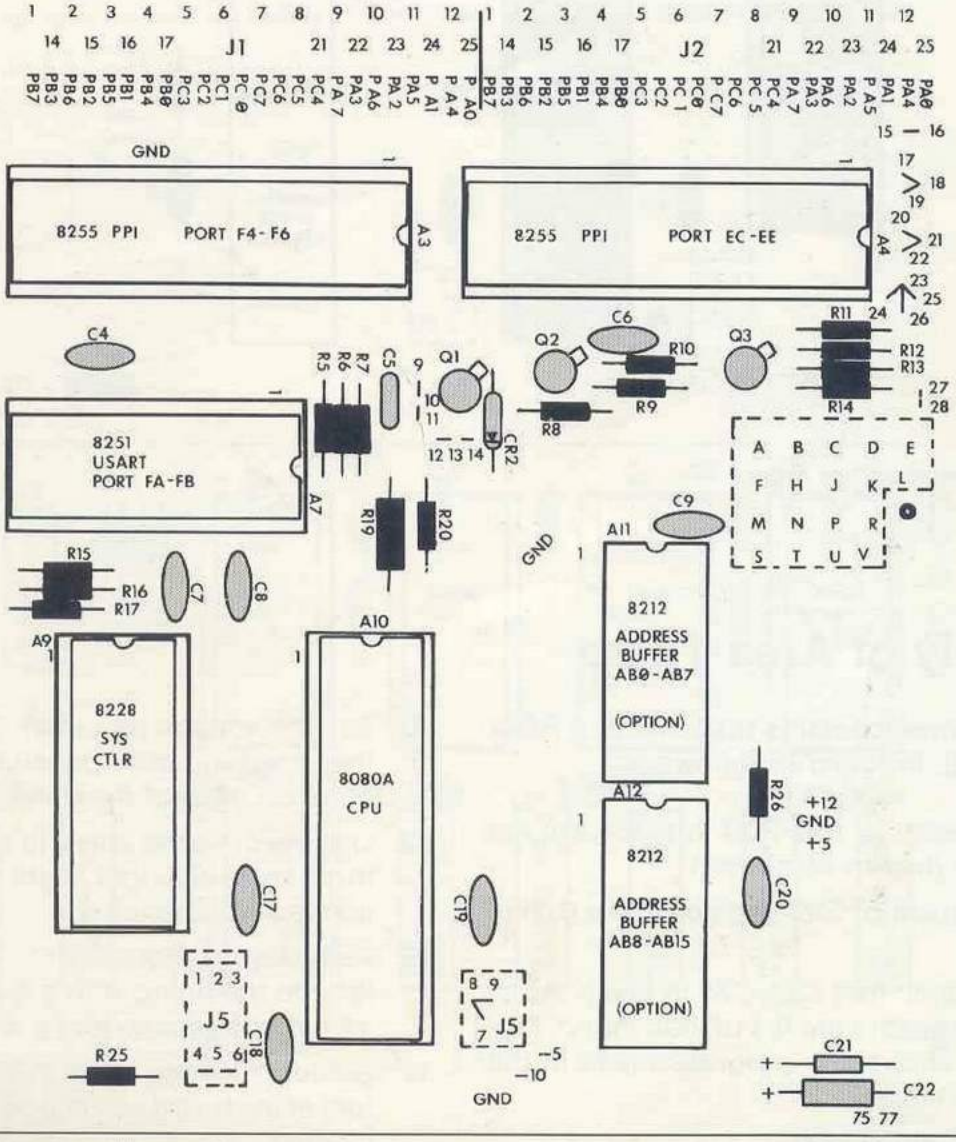


Figure 2-8. Construction Figure #6.

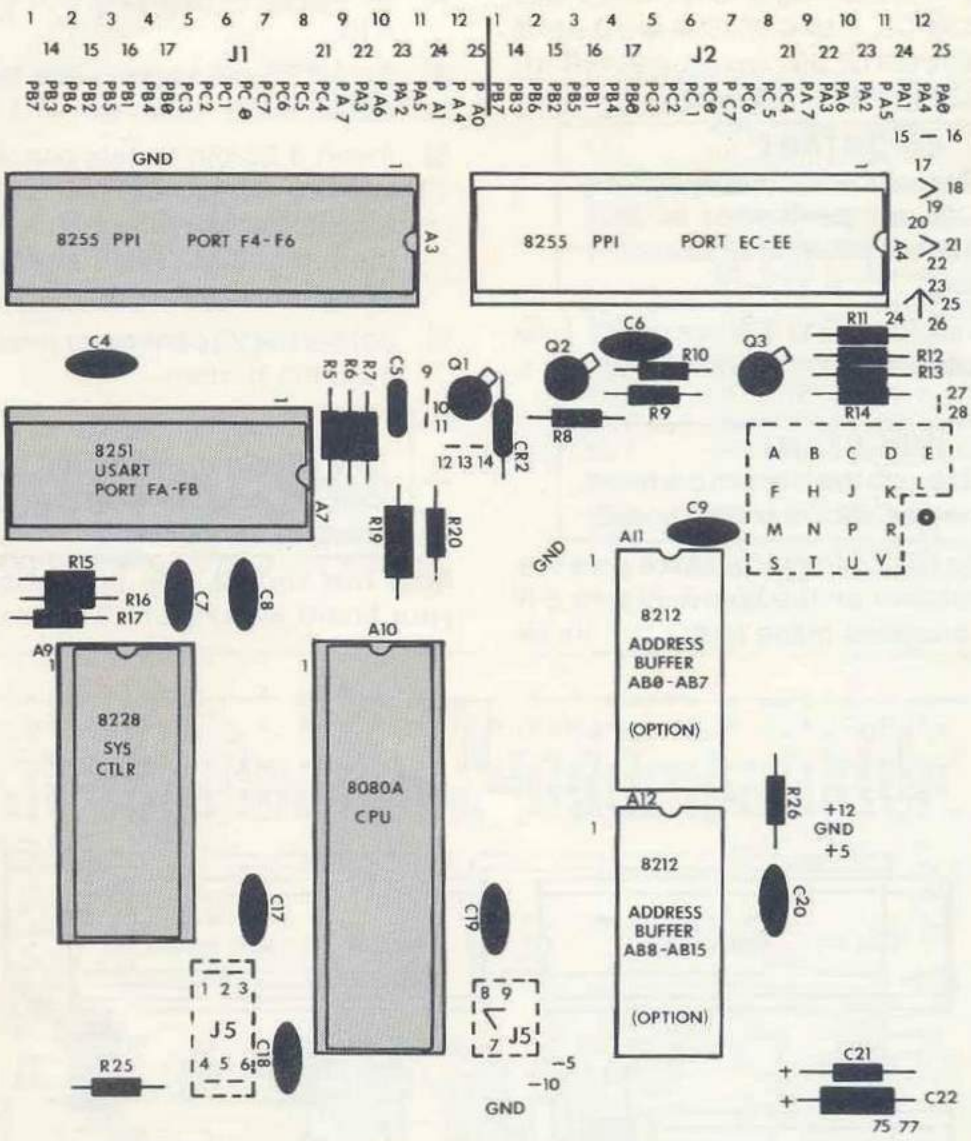


Figure 2-9. Construction Figure #7.

Assembly of Area Three

Board **area three** contains the RAM and ROM memory logic. Proceed as follows:

- ☐ Solder resistors R27-R30 in place. All are 1K, 1/4W (brown-black-red).
- ☐ Solder capacitor C23 in place. It is a 0.01 uf ceramic.
- ☐ Solder capacitors C24-C35 in place. All of these capacitors are 0.1 uf, 50V mono. Figure 2-10 shows the progress made in the last three steps.
- ☐ Solder 24-pin socket into location A14.
- ☐ Set the voltage regulator (79M05AUC) on the board and use a pencil to mark the bend point on each of the three leads.
- ☐ Use needle-nose pliers to bend each of the three leads at a right angle at the points you marked with pencil.
- ☐ Referring to Figure 2-11, fasten the regulator to the board with a 4-40 x 3/8 screw, 2 #4 nylon washers, and a 4-40 nut.
- ☐ Solder the three leads in place on the bottom of the board and clip off any excess lead lengths.

- ☐ Insert a 25-pin female connector into location J3. At each of two locations, place a 2-56x3/8 screw through the connector from the top and secure it at the bottom with a #2 nylon washer and a 2-56 nut.
- ☐ Solder the 25 connector pins onto the board from the bottom.

> **NOTE**
> **The 25-pin male connector provided can be used to interface your hardware to connector J3.**

You have now completed area three. Compare your board with Figure 2-12 (next page).

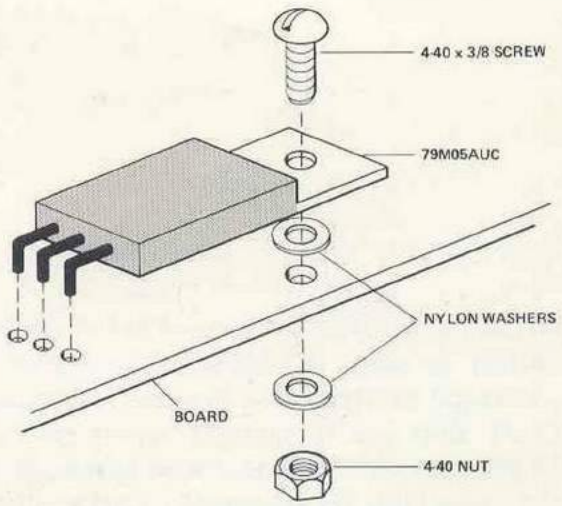


Figure 2-11. Voltage Regulator (VR1) Installation.

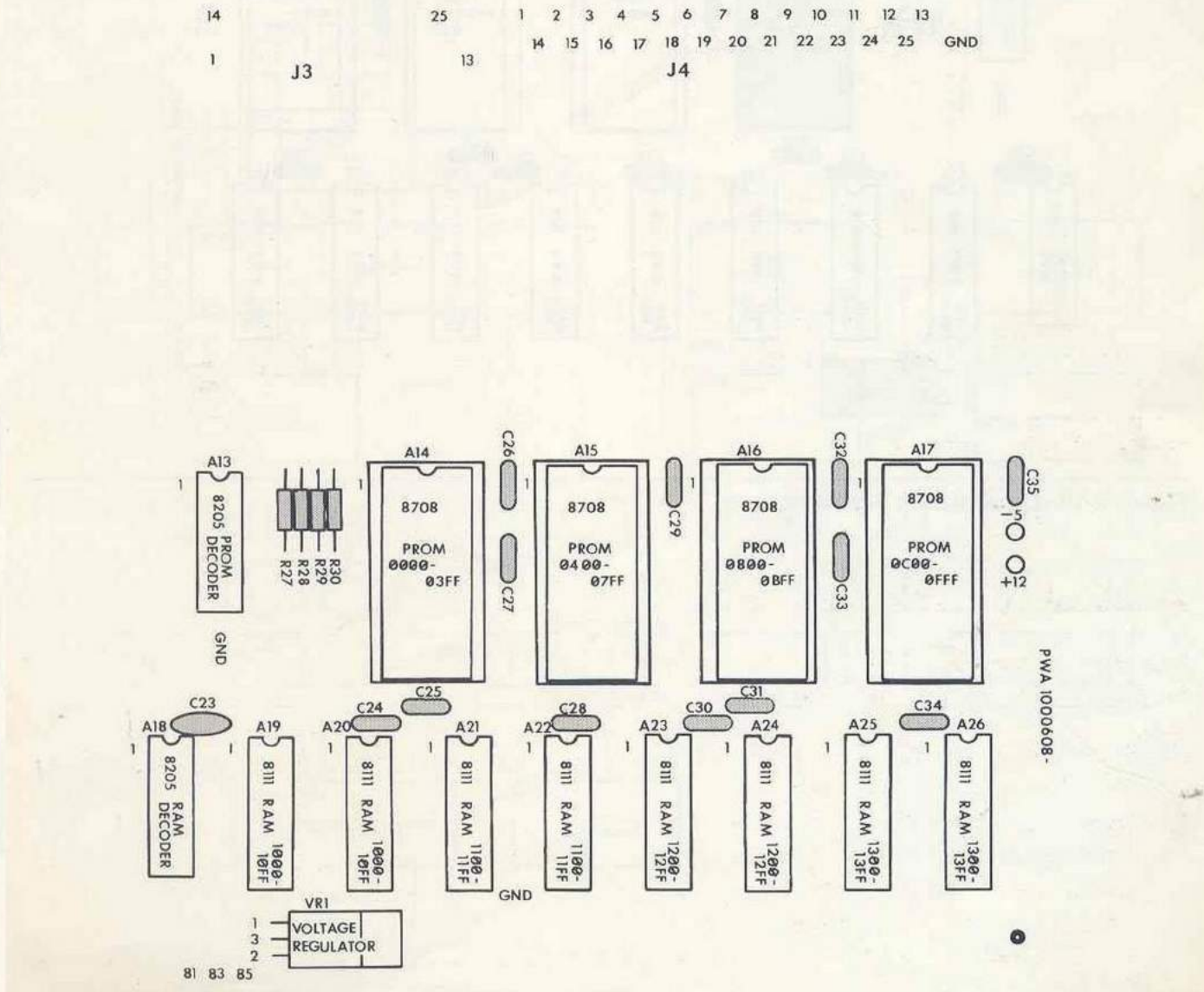


Figure 2-10. Construction Figure #8.

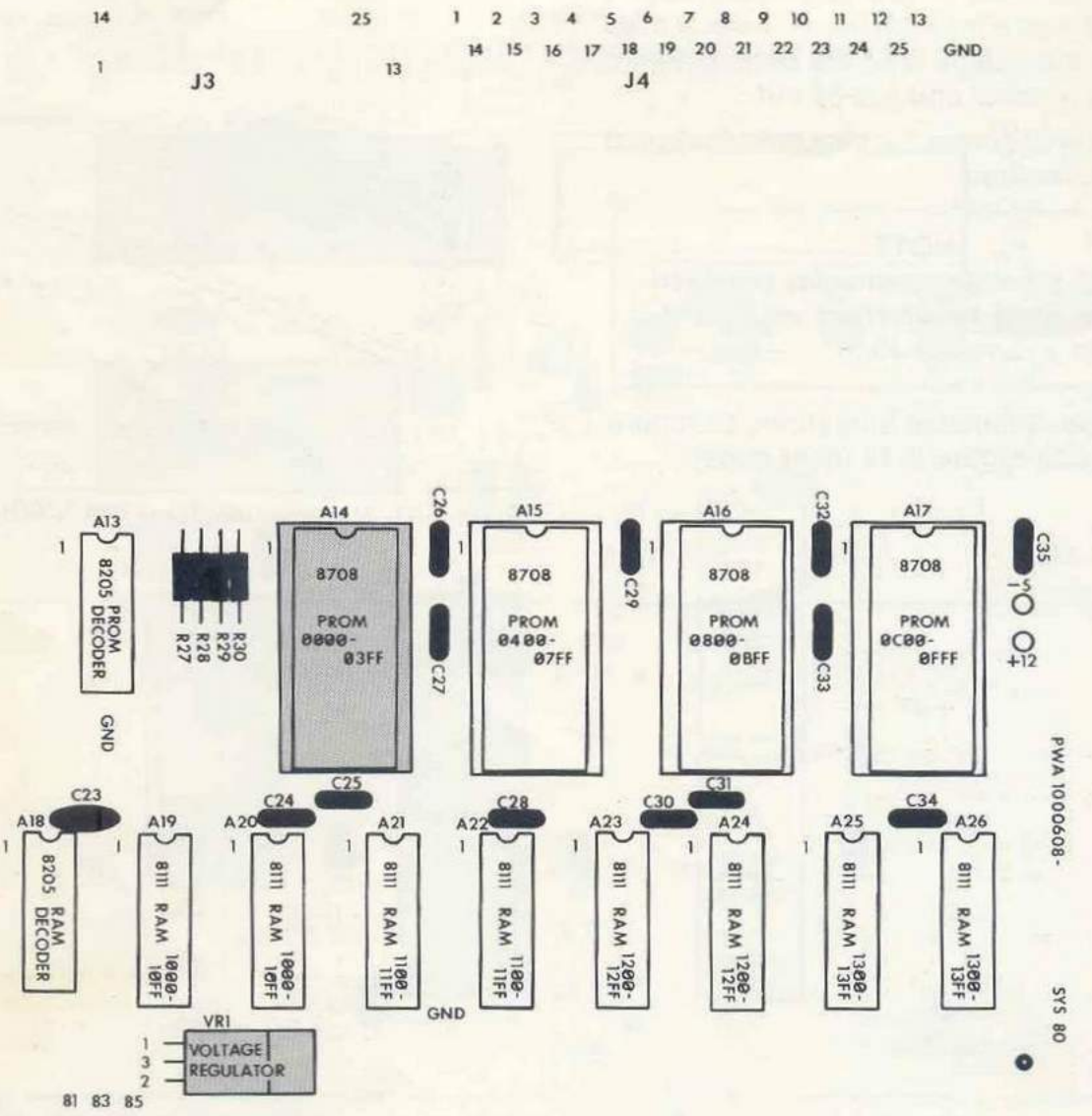


Figure 2-12. Construction Figure #9.

CHAPTER 3
THEORY OF OPERATION

GENERAL

Now that you have assembled the structure of the SDK-80 it is time to discuss the internal composition of the design. We will do this by presenting the functional organization of the SDK-80 logic and, in the process, bring in the decisions that you, as the user, must make before completing the kit.

Figure 3-1 is a functional block diagram of the SDK-80. It has been purposely drawn as simple as possible in order to give a basis for discussion. You will note that this figure shows only the major signals in the unit. For this reason, some occasional reference to the SDK-80 schematics (Appendix B) will be in order.

The text to follow describes each of the elements in the block diagram.

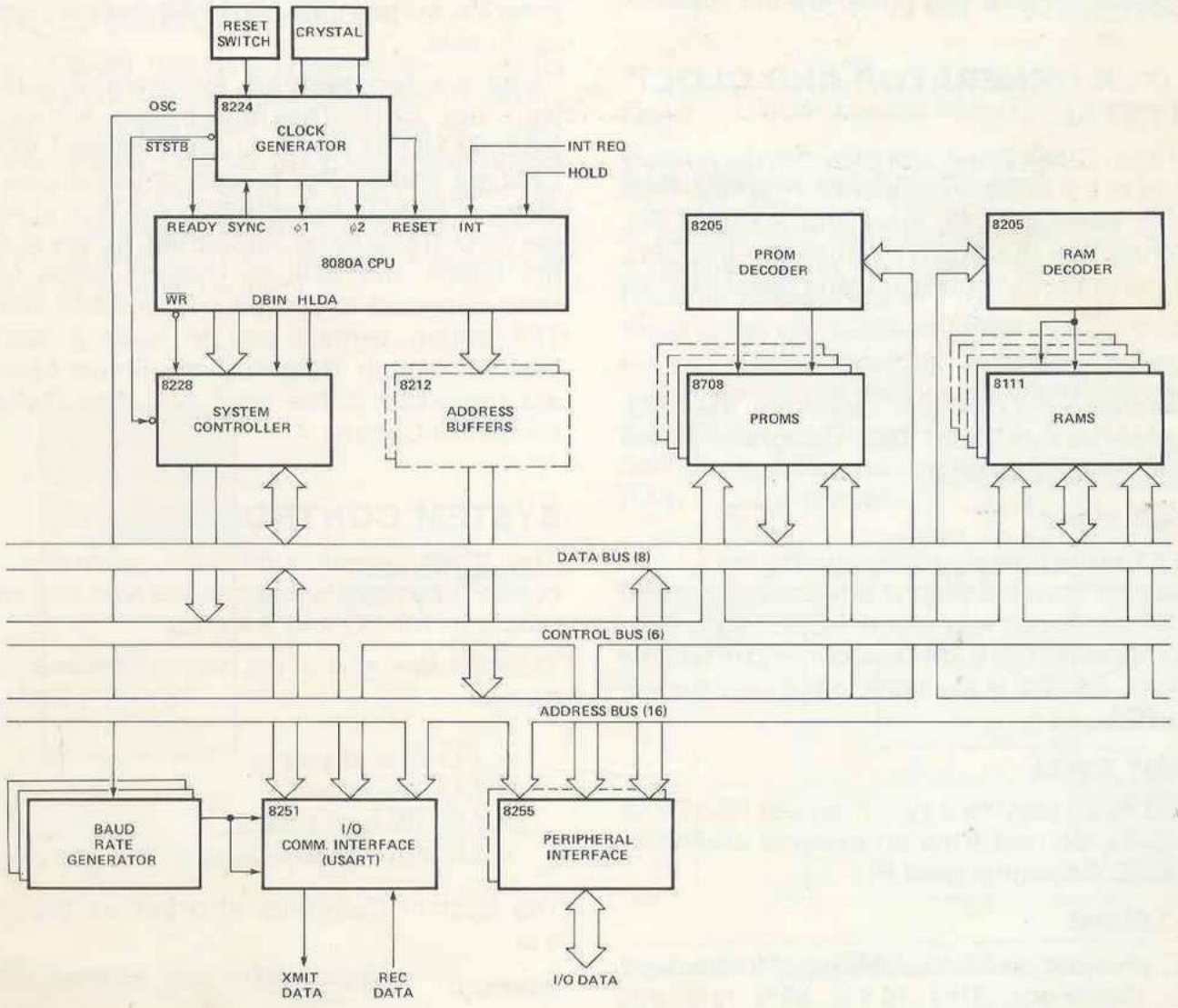


Figure 3-1. SDK-80 Functional Block Diagram.

SYSTEM BUSSES

The SDK-80 logic is built around three system busses: the data bus, the address bus and the control bus. All of the MCS-80™ components communicate via these three busses.

The system busses can be selectively enabled/disabled from the user system if the board is jumpered for that capability. Bus enable jumpering is described in the System Bus Enable section of Chapter 4.

Each bus is more fully described in the 8080 Microcomputer Systems User's Manual.

RESET SWITCH

The Reset Switch gives you the capability of forcing a reset to the SDK-80 logic at any time. When the switch is pressed, the Clock Generator will send a RESET signal throughout the system. The Reset Switch should be pressed each time you power-up the system.

CLOCK GENERATOR AND CLOCK CRYSTAL

The 8224 Clock Generator provides the primary timing to the system. It generates the high-level clocks necessary to drive the 8080A CPU, synchronizes the READY signal into the CPU, and transmits the power-up (and Reset Switch) reset signal.

Ø1 and Ø2 Clocks

Ø1 and Ø2 are 2.048 MHz clocks for the CPU. They are derived from OSC using an internal divide-by-nine function.

RESET Signal

RESET is the primary reset signal to the system logic. It is asserted both at power-up and when the Reset Switch is pressed. RESET clears the CPU, disables the RAM Decoder, and resets the USART. RESET is available to the user system at pad V.

READY Signal

READY can provide a synchronized READY to the CPU, derived from an external asynchronous RDYIN signal (pad P).

OSC Signal

OSC provides an 18.432 MHz input to the Baud Rate Generator. This 18.432 MHz rate was chosen for two reasons. First, it permits the 8080A CPU to run at very close to its maximum speed. Second, it is a convenient rate to use in designing a simple, but highly stable, Baud Rate Generator.

$\overline{\text{STSTB}}$ (Status Strobe) Signal

At the beginning of each machine cycle, the CPU issues status information on its data bus. $\overline{\text{STSTB}}$ causes the 8228 System Controller to store this information into its status latch. $\overline{\text{STSTB}}$ is available to the user system as $\overline{\text{STATUS STROBE}}$ at pad J.

8080A CPU

The 8080A CPU is thoroughly described in the Intel® 8080 Microcomputer Systems User's Manual and need not be repeated here.

The CPU clocks, Ø1 and Ø2, will be supplied (at 2.048 MHz) by the Clock Generator.

The data bus will interface directly to the System Controller and the address bus will enter the system through the Address Buffers, if applicable.

There are two separate jumper-wire options with the CPU. The first option allows an external HOLD signal to be presented to the CPU via pad R. The second option allows an external READY signal to force a Wait state in the CPU. It should be pointed out, however, that the 8080A and SDK-80 memory chips have been designed to operate without Wait states. The option permits you to force a Wait if desired, though. Both of these jumper options are described in the Hold And Wait Options section of Chapter 4.

SYSTEM CONTROLLER

The 8228 System Controller generates the control bus signals that provide read and write functions for I/O and memory.

They are available to the user system as shown below:

- $\overline{\text{I/O W}}$ is at pad E
- $\overline{\text{I/O R}}$ is at pad L
- $\overline{\text{MEMW}}$ is at pad U
- $\overline{\text{MEMR}}$ is at pad T

The System Controller also buffers the data bus.

Interrupt

A single-level interrupt structure is provided such that whenever pad H ($\overline{\text{INT REQ}}$) is

grounded, the System Controller causes a Restart instruction (RST 7) to be inserted into the CPU. This feature provides a single interrupt vector without using additional components, such as an interrupt instruction port.

Multiple level interrupts will require additional chips to be installed into the wire-wrap area.

ADDRESS BUFFERS (OPTIONAL)

The 8212 Address Buffers are not included in the System Design Kit, but must be added if more than a nominal amount of memory (more than 1024 bytes of RAM and more than 4K bytes of ROM) is used. The Address Buffers are tri-state TTL buffers that provide 15mA drive.

The address bus level can be forced to the high-impedance state by inputting a high level on pad S (SYSTEM BUS ENABLE), if the board is jumpered for this capability.

SDK-80 MEMORY

The SDK-80 has two types of memory. Its ROM Memory can accommodate from 1K to 4K bytes, where the lower 1K bytes are dedicated to the system monitor. Its RAM Memory can accomodate from 256 to 1K bytes, in which all but the uppermost 30 bytes (addresses 13E2-13FF) are useable by your system. Figure 3-2 is a map of SDK-80 memory.

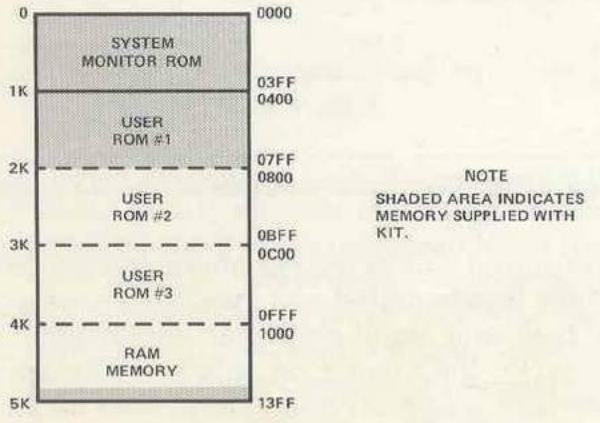


Figure 3-2. SDK-80 Memory Map.

ROM Decoder and ROM Memory

The SDK-80 can accomodate up to four 1024x8 8708/8308 Read Only Memory (ROM) chips.

The ROM that installs into board location A14 has been pre-programmed with the SDK-80 system monitor.

The 8708/8308 that installs into board locations A15, A16, and A17 can be used to hold a program that you have developed and checked out in RAM.

The 8205 ROM Decoder selects the ROM chip being addressed. Figure 3-3 shows the ROM address format.

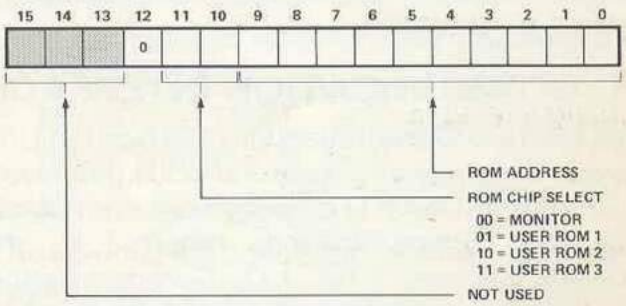


Figure 3-3. ROM Address Format.

RAM Decoder and RAM Memory

In the standard configuration, the SDK-80 can accommodate up to eight 256x4 Static MOS Random Access Memory (RAM) chips. Two of these chips are supplied in the System Design Kit, so users requiring only 256 bytes of memory need not install additional RAM chips.

The 8205 RAM Decoder selects the RAM chip pair being addressed. Figure 3-4 shows the RAM address format.

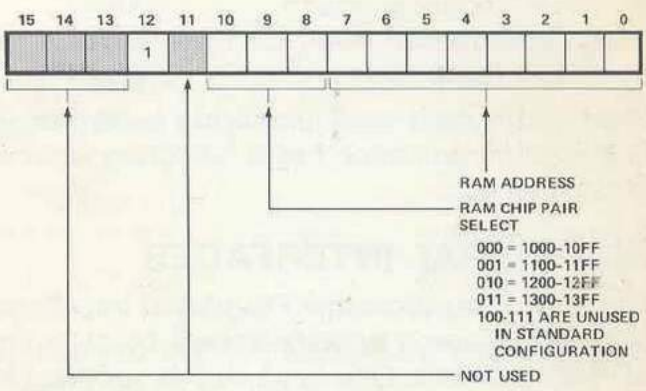


Figure 3-4. RAM Address Format.

RAM access is disabled whenever the RESET signal from the Clock Generator is asserted.

BAUD RATE GENERATOR

The Baud Rate Generator circuit supplies the transmitter and receiver clocks to the I/O Communication Interface. This circuit is made up of three IC chips: one 93S16 and two 74161s.

The Baud Rate Generator takes the 18.432 MHz OSC signal from the Clock Generator and, by internal division, generates a series of signals which represent baud rates between 75 and 4800. The baud rate that will be presented to the I/O Communication Interface is determined by jumper-wiring or a rotary switch. This selection will be discussed in the Baud Rate Selection section of Chapter 4.

I/O COMMUNICATION INTERFACE

The 8251 I/O Communication Interface is a Universal Synchronous/Asynchronous Receiver/Transmitter (USART) chip that accommodates any data communications required by the SDK-80 system. The I/O Communication Interface can accept parallel data from the data bus and send it serially to an external device. It can also accept serial data from an external device and put it onto the data bus in parallel form when eight bits have been collected. Figure 3-5 shows the address format for communications.

The baud rate at which the I/O Communication Interface will transmit and receive data is governed by the Baud Rate Generator.

The I/O Communication Interface circuit on the board also includes some jumpers that select the communication input/output level. Any of three levels may be selected.

- RS-232 level, which is typically used for CRT applications
- Current-loop level, for TTY applications
- TTL level.

The input/output level jumpering is discussed in the Communication Level Selection section of Chapter 4.

PERIPHERAL INTERFACES

The 8255 Programmable Peripheral Interfaces provide the user's primary access point to the SDK-80 data bus. One 8255 chip is supplied in the System Design Kit.

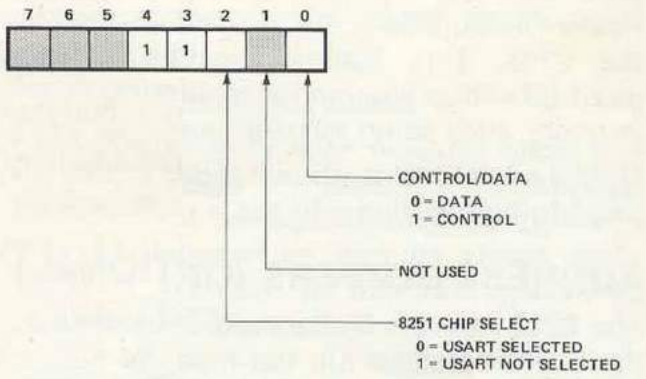


Figure 3-5. I/O Communication Interface Address Format.

Each Peripheral Interface chip provides three 8-bit parallel I/O ports, each of which is independently addressable. Figure 3-6 shows the address format for I/O port selection.

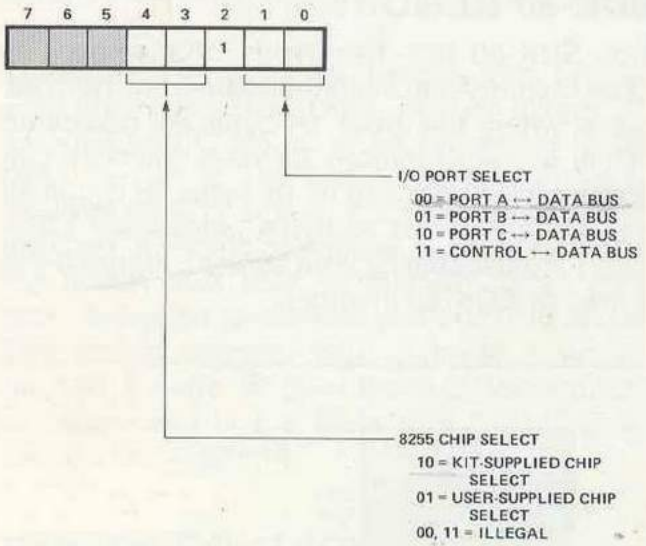


Figure 3-6. Peripheral Interface Address Format.

The output pins of the Peripheral Interfaces are totally uncommitted and may be jumper-wired to best suit your particular application. For example, they might be wired directly to the interface plugs or, alternately, they might be wired to standard TTL buffers in the wire-wrap area before coming back to the plugs. This wiring is further discussed in the Output Wiring section of Chapter 4.

CHAPTER 4
FINAL ASSEMBLY AND CHECKOUT

GENERAL

At this point in the manual you should have completed the preliminary assembly and read the theory of operation. You can now finish the board assembly and begin a checkout sequence.

JUMPER-WIRING THE BOARD

The SDK-80 is designed to be used in virtually any evaluation application and can be jumpered to suit your particular requirements. These questions will help you decide what jumpers are needed:

1. Will you ever want the CPU to enter a Hold or Wait state?
2. Will you ever want to disable the system busses?
3. What type of input device will you use to communicate with the SDK-80 (e.g., CRT, Teletype)?
4. What is its baud rate?
5. Will you be using 8212 Address Buffer chips?
6. What kind of information will be transferred to/from the SDK-80?

If you have a fairly good idea of the answers to all of these questions, you are ready to start jumper-wiring the board. The scrap leads that have been cut from previously-installed resistors are a good source of jumper wire. However, use 22-gauge insulated wire in situations where any jumpers may make contact with each other.

Hold and Wait Options

The SDK-80 is designed to run without Hold or Wait states. However, a jumper-wire option is available to give either capability.

- ☐ To disable the Hold state, wire J5-2 to J5-3.
- ☐ To enable the Hold state, wire J5-1 to J5-2.
- ☐ If READY is to force an 8080 Wait state, wire J5-8 to J5-9. If not, wire J5-8 to J5-7.

System Bus Enable

One jumper is available to make it possible to selectively disable the SDK-80 system bus.

- ☐ If the bus will be selectively disabled, wire J5-5 to J5-6.
- ☐ If the bus should remain enabled at all times, wire J5-4 to J5-5.

Baud Rate Selection

The communications baud rate can be selected in two ways, depending on the application. If only one baud rate will be employed, the rate can be selected by installing a single jumper wire. If two or more baud rates will be employed in the application, however, the Spectrol rotary switch installed in Chapter 1 will be used for this purpose.

- ☐ To select a fixed baud rate, jumper pad 29 to one of the pads 31-37 per Table 4-1.

Table 4-1. Baud Rate Selection Table.

Baud Rate	Wire Pad 29 To
4800	31
2400	32
1200	33
600	34
300	35
150	36
75 or 110	37

- ☐ For 110 baud, the standard Teletype rate, wire pad 4 to pad 5.

Communication Level Selection

Any of three communication levels can be selected: CRT, Teletype, or TTL. All serial data is passed through connector J3.

Table 4-2. Communication Level Jumper Table.

CRT Configuration Jumpers	TTY Configuration Jumpers	TTL Configuration Jumpers
23 to 24	23 to 26	23 to 25
17 to 18	18 to 19	17 to 18
9 to 10	10 to 11	12 to 13
13 to 14	13 to 14	2 to 3
2 to 3	1 to 2	20 to 21
6 to 8	7 to 8	
27 to 28	15 to 16	
21 to 22	21 to 22	

- ☐ Jumper wire pads 1 through 28 per Table 4-2.
- ☐ If your system does not contain a modem, jumper pad A to pad B.

Address Bus Jumpers

If you do not use 8212 Address Buffer chips on your SDK-80, the address bus must be jumpered across locations A11 and A12. In this situation, connect the following jumpers AT BOTH LOCATION A11 AND LOCATION A12. All jumpers should be installed from the circuit side of the board i.e., **NOT** the silk-screen side.

- ☐ Jumper pad 3 to pad 4.
- ☐ Jumper pad 5 to pad 6.
- ☐ Jumper pad 7 to pad 8.
- ☐ Jumper pad 9 to pad 10.
- ☐ Jumper pad 15 to pad 16.
- ☐ Jumper pad 17 to pad 18.
- ☐ Jumper pad 19 to pad 20.
- ☐ Jumper pad 21 to pad 22.

Output Wiring

Connector J3 is dedicated as a communications interface (see Table 4-3) and is, in fact, the only committed interface in the SDK-80. All other interfacing is at the discretion of the user.

For example, the 8255 Peripheral Interface might be jumpered directly to connector J1 or, alternately, might be jumpered to TTL buffers in the wire-wrap area before being passed to J1. Conversely, you might wish to add a switch array to the 8255 area in order to send data to the CPU.

Your System Design Kit includes male connectors that mate with the female connectors installed at J1 and J3.

A group of control signals are available at the alphabetic-labeled pads in area two of the board. Table 4-4 identifies these pads.

Table 4-3. Pin Assignments for Communications Interface (J3).

J3 Pin	CRT Configuration	TTY Configuration	TTL Configuration
1			
2	CRT REC. DATA		TTL REC. DATA
3	CRT XMIT DATA		TTL XMIT DATA
4			
5	+12 VDC		
6			
7	SIGNAL GND		SIGNAL GND
8	+12 VDC		
9			
10			
11			
12		TTY REC RETURN	
13		TTY XMIT	
14			
15			
16			
17			
18			
19			
20	+12 VDC		
21			
22			
23			
24		TTY REC	
25		TTY XMIT RETURN	

Table 4-4. SDK-80 Control Bus Pads.

Pad	Mnemonic	Description
A	$\overline{\text{CTS}}$	Clear To Send
B	$\overline{\text{RTS}}$	Request to Send
C	ø2 (TTL)	2.048 MHz Clock
D	$\overline{\text{DSR}}$	Data Set Ready
E	$\overline{\text{I/O W}}$	I/O Write
F	$\overline{\text{DTR}}$	Data Terminal Ready
H	$\overline{\text{INT REQ}}$	Interrupt Request
J	$\overline{\text{STATUS STROBE}}$	Status is on Data Bus
K	OSC	18.432 MHz Oscillator
L	$\overline{\text{I/O R}}$	I/O Read
M	HLDA	Hold Acknowledge
N	$\overline{\text{INTA}}$	Interrupt Acknowledge
P	READY	Ready
R	HOLD	Hold
S	$\overline{\text{SYSTEM BUS ENABLE}}$	Enables Data Bus and Address Bus
T	$\overline{\text{MEMR}}$	Memory Read
U	$\overline{\text{MEMW}}$	Memory Write
V	RESET	Reset

INSTALLING INTEGRATED CIRCUITS

You have now reached the point where you will start installing IC's in the board, but a few words are in order before you begin.

Special Precautions For Handling MOS IC's

The Kit's MOS IC's (8080, 8111, 8251, 8255, and 8708) are particularly susceptible to static electricity. They can be easily damaged if proper care is not taken in handling them. For this reason, the following steps should be adhered to as closely as possible:

1. All equipment (soldering iron, tools, solder, etc.) should be at the same potential as the PW board, the assembler, the work surface and the IC itself along with its container. This can be accomplished by continuous physical contact with the work surface, the components, and everything else involved in the operation.
2. When handling the IC, develop the habit of first touching the conductive container in which it is stored before touching the IC itself.
3. Always touch the SDK-80's PW board before touching the IC to the board. Try to maintain this contact as much as possible while installing the IC.
4. Handle the IC by the edges. Avoid touching the pins as much as possible.
5. In general, never touch anything to the IC that you have not touched first while touching both it and the IC itself.

Aligning the IC Pins

The connector pins of Integrated Circuit chips are very fragile and can be easily pushed out of line. In fact, sometimes IC's will arrive with one or more pins out of line. Trying to install a misaligned IC is a hapless task and, worse, might cause permanent damage to the chip.

Aligning the pins of an IC is an easy job. Simply lay the IC on its side on your work surface, hold the chip by its body and exert enough pressure so that all pins are perpendicular to the body.

Chip Orientation

The IC's must be correctly oriented on the board or they will not operate properly. One end of the chip will carry some sort of identifying mark, typically a notch or a dot or a + sign. The chip must be installed so that this identifier corresponds to the silkscreened "1" on the board.

Installing IC Chips

After orienting the IC, follow these steps to install it in the board:

1. Start the pins on one side of the IC into their respective holes on the silk-screened side of the PW board. DO NOT PUSH THE PINS IN ALL THE WAY. If you have difficulty getting the pins into the holes, use the tip of a small screwdriver to guide them.
2. Start the pins on the other side of the IC into their holes in the same manner. When all of the pins have been started, set the IC in place by gently rocking it back and forth until it rests as close as possible to the board or socket.
3. If the IC is not installed in a socket, turn the board over and solder each pin to the foil pattern on the back side of the board. Be sure to solder each pin and be careful not to leave any solder bridges.

Removing IC Chips

If required, an IC chip can be removed from a socket by gently rocking it back and forth to start its release. When a gap exists between the chip and socket, pry it gently at alternate ends until the pins start to come loose. A popsicle stick or small screwdriver works well here. Then hold the chip by the ends and pull it free. Try to keep the chip fairly parallel to the socket throughout this operation.

Clock Generator

Besides the 8080, the most critical chip in the SDK-80 circuit is the 8224 Clock Generator.

☐ Insert the 8224 Clock Chip into the socket at location A8.

Power, Clock and Reset Verification

With this single chip installed, we can check the power and clock inputs and the operation of the Reset Switch. The procedure is as follows:

☐ Connect your power supply to terminal lugs E1-E6 on the SDK-80 board.

NOTE

The SDK-80 edge connector is power-compatible with Intel's MDS (Microcomputer Development System). If you have an MDS, the SDK-80 can derive its power through installation in the MDS chassis.

- ☐ Turn power on.
- ☐ Using a voltmeter, verify +5 VDC at the pad provided.
- ☐ Verify +12 VDC at the "+12" pad.
- ☐ Verify your supply's negative voltage at the "-10" pad.
- ☐ Verify -5 VDC at the "-5" pad, near location A17.
- ☐ Press the Reset Switch a few times and check for +4 VDC at A8, pin 1(RESET).

NOTE

Develop the habit of pressing the Reset Switch each time you power-up the system.

- ☐ If you have an oscilloscope, verify that A8 pins 10 and 11 each show 2.048 MHz clocks (Ø2 and Ø1, respectively).
- ☐ Using an oscilloscope, verify that A8 pin 12 shows an 18.432 MHz clock (OSC).
- ☐ Turn the power off.

Remainder of SDK-80 ICs

After having verified that the SDK-80 logic is correctly receiving power, the system clocks and the RESET signal, you can finish installing the chip complement. Some of the IC's will plug into sockets, others will have to be soldered onto the board.

The procedure is as follows:

- ☐ Solder the 93S16 chip into location A1.
- ☐ Solder a 74161 chip into locations A2 and A5.
- ☐ Solder the 7406 chip into location A6.
- ☐ If applicable, solder 8212 chips into locations A11 and A12.
- ☐ Solder 8205 chips into locations A13 and A18.
- ☐ Solder 8111 chips into locations **A25** and **A26.**
- ☐ Insert the 8228 chip into the socket at location A9.
- ☐ Insert the 8080A chip into the socket at location A10.
- ☐ Insert the 8251 chip into the socket at location A7.
- ☐ Insert the 8255 chip into the socket at location A3.
- ☐ Insert the pre-programmed monitor ROM chip into the socket at location A14.

Table 4-5. Power Requirements.

Symbol	Voltage	Minimum System	Maximum System	Unit
V_{CC}	+5V ±5%	1.3	2.1	Amps
V_{DD}	+12V ±5%	.35	.45	Amps
V_{BB}	-10V ±5% -12V ±5%	.20	.30	Amps

START-UP PROCEDURE

You have now completed the SDK-80 assembly and are ready to start up the system. The start-up procedure is as follows:

- ☐ Plug your system communication monitor (CRT, Teletype, etc.) into the SDK-80 connector J3.
- ☐ Turn power on at both the SDK-80 power supply and your communication monitor.
- ☐ Press the Reset Switch.

At this point, your monitor will display the following message:

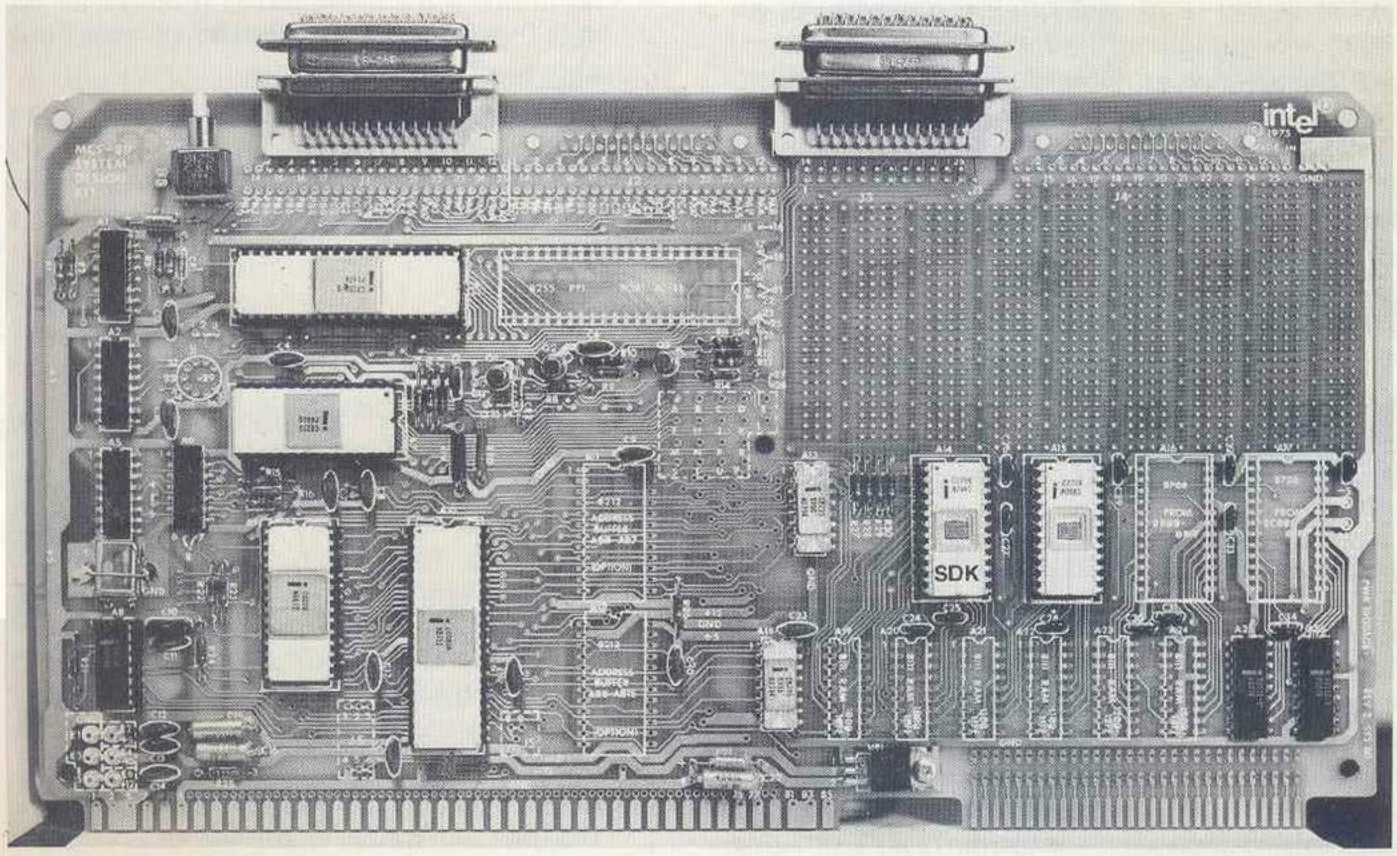
MCS-80™ KIT

Congratulations! You are now ready to start using the system.

TROUBLESHOOTING HINTS

If the SDK-80 system does not work properly, turn the power off and investigate these areas:

1. Verify that all resistors have been properly installed and are correctly color-coded. Appendix C summarizes the component values.
2. Verify that all capacitors have been properly installed and that all electrolytic capacitors are installed with proper polarity.
3. Verify that both diodes (CR1 and CR2) have been installed with proper polarity.
4. Verify that the metal tabs of all three transistors are properly positioned.
5. Verify that all IC's are installed with their "1"-end identifiers correctly oriented.
6. Verify that all jumpers have been properly installed.
7. If the above hints do not fix the problem contact the distributor where the SDK-80 was purchased.



Assembled Board (Without Jumpers).

CHAPTER 5 SDK-80 MONITOR

INTRODUCTION

The SDK-80 Monitor is an Intel® 8080 program provided in a pre-programmed ROM. The Monitor accepts and acts upon user commands to operate the SDK-80. It also provides input and output facilities in the form of I/O drivers for user console devices. The Monitor provides the following facilities:

- Displaying selected areas of memory.
- Initiating execution of user programs.
- Setting "BREAK POINTS" in user programs.
- Modifying contents of memory and processor registers.
- Inputting hexadecimal data from the console device to memory.

The Monitor communicates with the user through an interactive console device, normally a Teletype or CRT Terminal. The dialogue between the operator and Monitor consists of user-originated commands in the Monitor's command language, and Monitor responses, either in the form of a printed message or an action being performed. After the cold start procedure (described under the heading, "Cold Start Procedures" in Section III), the Monitor begins the dialogue by typing the sign-on message on the console and requesting a command by presenting a prompt character, "." (period).

MONITOR OPERATIONS

The SDK-80 Monitor is a command controlled operations supervisor for the 8080 Microcomputer System Design Kit. Control commands are discussed in Section II, "Command Structure".

I. FUNCTIONAL SPECIFICATION

A. General Characteristics and Scope of Product

The monitor is a program written in Intel® 8080 macro assembly language. The monitor resides in 1K (K = 1024 bytes) of programmed ROM and is located in the address space of the 8080 microcomputer between 0 and 1K. The non-volatile nature of the program's storage medium means that the monitor is available for use immediately after power-on or reset.

B. Description of All Major Functions Performed

1. CONSOLE COMMANDS

   The monitor communicates with the operator via an interactive console, normally a teletypewriter. The dialogue between the operator and the monitor consists of commands in the monitor's command language and the monitor's responses. After the cold start procedure, the monitor begins the dialogue by typing a sign-on message on the console and then requests a command by presenting a prompt character, ".". Commands are in the form of a single alphabetic character specifying the command, followed by a list of numeric or alphabetic parameters. Numeric parameters are entered as hexadecimal numbers. The monitor recognizes the characters 0 through 9 and A through F as legal hexadecimal digits. The valid range of numbers is from 1 to 4 hex digits. Longer numbers may be entered, but such numbers will be evaluated modulo 2^{16} so that they will fall into the range specified above.

   The only command requiring an alphabetic parameter is the "X" command. The nature of such parameters will be discussed in the section explaining the command.

2. USE OF THE MONITOR FOR PROGRAMMING AND CHECKOUT

   The monitor allows the user to enter, check out, and execute small demonstration programs. The monitor contains facilities for memory modification, 8080 CPU register display and modification, program loading from the console device, program initiation, and the recognition of an "RST 7" instruction as an unconditional branch to RAM address 13FDH. By inserting RST 7 instructions in a program under test, or by using the hardware generated RST 7 instruction (if available), the user can cause execution of a program to transfer to a dedicated location, for whatever purposes he desires.

   When the user wishes to re-enter the

monitor, he should use an RST 1 instruction, either generated by hardware or coded into his program. When entered in this manner, the monitor will automatically save the state of the 8080: specifically, it will save all registers (A, B, C, D, E, H, L), the CPU flags (F), the user's Program Counter (PC), and the user's Stack Pointer (SP). These may be examined with the X command. When the operator enters a G command, these values will be restored.

3. I/O SYSTEM

   The I/O system provides two routines, console character in and console character out, which the user may call upon to read and write, respectively, characters from and to the console device.

C. Applicable Standards

Throughout this specification, the numbering convention for bits in a word is that bit 0 is the least significant, or rightmost bit.

The internal code set used by the monitor is 7 bit (no parity) ASCII.

II. INTERFACE SPECIFICATIONS

A. Command Structure

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 that may be encountered while operating the monitor are described in Section IV.C.

The monitor requires each command to be terminated by a carriage return. With the exception of the "S" 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).

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 a null parameter. Null parameters are illegal in all commands except the "X" command (see below).

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

1. DISPLAY MEMORY COMMAND, D

   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
```

2. PROGRAM EXECUTE COMMAND, G

   G[<entry point>]

   Control of the CPU is transferred from the monitor to the user program by means of the program execute command, G. The <entry point> should be an address in RAM which contains an instruction in the user's program. If no entry point is specified, the monitor uses, as an address, the value on top of the stack when the monitor was entered.

   Example

```
G1400
Control is passed to location 1400H.
```

3. INSERT INSTRUCTIONS INTO RAM, I

   I <address>

   Single instructions, or an entire user program, are entered into RAM with the I command. After sensing the carriage return terminating the command line, the monitor waits for the user to enter a string of hexadecimal digits (0 to 9, A to 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

memory locations. Two hexadecimal digits are loaded into each byte of memory.

Separators between digits (spaces, commas, carriage returns) are ignored; illegal characters, however, will terminate the command with an error message (see section IV.C.1). The character ESC or ALTMODE (which is echoed to the console as "$") terminates the digit string. If an odd number of hex digits have been entered, a 0 will be appended to the string.

Example

```
I1410
1122334455667788 99$
```

This command puts the following pattern into RAM:

1410 11 22 33 44 55 66 77 88 99

```
I1440
123456789$
```

This command puts the following pattern into RAM:

1440 12 34 56 78 90

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

4. MOVE MEMORY COMMAND, M

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

   The M command moves the contents of memory <low address> and <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 1410 contains 1AH, the command

   M1410, 141F, 1411

   will result in locations 1410 to 1420 containing "1A1A1A...".

   The monitor will continue to move data until the source field is exhausted, or until it reaches address 0FFFFH. If the monitor reaches address 0FFFFH without exhausting the source field, it will move data into this location, then stop.

   Example

   ```
   M1410, 150F, 1510
   ```

   256 bytes of memory are moved from 1410-150F to 1510-160F by this command.

5. SUBSTITUTE MEMORY COMMAND, S

   S <address>

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

   i. Type an S, followed by the hexadecimal address of the first memory location you wish to examine, followed by a space or comma.

   ii. The contents of the location is displayed, followed by a dash (-).

   iii. To modify the contents of the location displayed, type in the new data, followed by a space, comma, or carriage return. If you do not wish to modify the location, type only the space, comma, or carriage return.

   iv. If a space or comma was typed in step (iii), the next memory location will be displayed as in step (ii). If a carriage return was typed, the S command will be terminated.

   Example

   ```
   S1450 AA- BB-CC 01-13 23-24
   ```

   Location 1450, which contains AA is unchanged, but location 1451 (which used to contain BB) now contains CC, 1452 (which used to contain 01) now contains 13, and 1453 (which used to contain 23) now contains 24.

6. EXAMINE AND MODIFY CPU REGISTERS COMMAND, X

   X [<register identifier>]

   Display and modification of the CPU registers is accomplished via 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 — 8080 CPU register A
B — 8080 CPU register B
C — 8080 CPU register C
D — 8080 CPU register D
E — 8080 CPU register E
F — 8080 CPU flags byte, displayed in the form as it is stored by the "PUSH PSW" (hex code F5) instruction
H — 8080 CPU register H
L — 8080 CPU register L
M — 8080 CPU registers H and L combined
P — 8080 Program Counter
S — 8080 Stack Pointer

The command operates as follows:

i. Type an X, followed by a register identifier or a carriage return.

ii. 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 and S), followed by a dash (-).

iii. 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.

iv. If a space or comma was typed in step (iii), the next register in sequence (alphabetical order) will be displayed as in step ii (unless S was just displayed in which case the command is terminated). If a carriage return was entered in step iii, the X command is terminated.

v. If a carriage return was typed in step (i) above, an annotated list of all registers and their contents are displayed.

Example

```
XA AA- BB- CC- DD- EE- FF- 12- 34- 1234- 0000
XA AA- 23- CC- 01- EE- FF- 12- 34- 1234- 1010
X
A-AA B-23 C-CC D-01 E-EE F-FF H-12 L-34 M-1234 P-01CF S-03CD
```

B. Console Device Drivers

The monitor interfaces to the console device via a universal synchronous/asynchronous receiver/transmitter (USART). The monitor drivers interface with the USART according to the USART specifications. At the time of the assembly of the kit, the USART may be configured for a particular type of console interface. The actual console device must conform to this interface.

C. Using the I/O System

The user may access the two monitor I/O system routines from his program by calling the routine desired. The following paragraphs describe the routines available and their respective functions.

CI — Console Input

This routine returns a 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 3FDH.

Example

```
CI      EQU  3FDH
        . . .
        CALL CI
        STA  DATA
        . . .
```

CO — Console Output

This routine transmits a 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 3FAH.

Example

```
CO      EQU  3FAH
        . . .
        MVI  C, "."
        CALL CO
```

III. OPERATING SPECIFICATIONS

A. Product Activation Instructions

1. COLD START PROCEDURE

   After a power-on or reset, the monitor will begin execution at location 0 in ROM. The monitor will perform an initialization sequence, and then display a sign-on message on the console. When the monitor is ready for a command, it will prompt with a period, ".".

2. USE OF RAM STORAGE IN THE MONITOR

   The monitor dynamically assigns its RAM stack near the top of the first 1K bytes of RAM (address space from 4K to 5K). The top 3 bytes in this block of RAM are reserved for a transfer address, supplied

by the user, which is used as a destination location for RST 7 instructions (or the optional hardwired instruction). Several additional bytes are used, below the stack, for temporary storage. Except for RAM addresses 5K-1 to 5K-256, all other RAM is available for the user.

3. BREAK POINT FACILITY

The monitor treats the RST 1 instruction (CF hex) as a special sequence initiator. Upon execution of an RST 1 instruction the monitor will automatically save the complete CPU status and output the sign on message "MCS-80™ KIT" on to the console device. The user can at that time display the contents of the CPU status by initiating an "X" command. After examining the machine status and making changes if necessary the user can resume execution of his program by simply inputting "G" and Carriage Return on the console device. By using the RST 1 break point facilities of the monitor the user can step through large portions of his program by inserting RST 1 instructions at key locations. This technique can significantly reduce the amount of time it takes to debug software.

4. INTERRUPT PROCESSING

The SDK-80 hardware is designed so that an external device can interrupt the CPU and execute an automatic RST 7 instruction. The monitor, upon execution of a RST 7 instruction, automatically executes an unconditional JUMP to RAM location (13FDH). This facility allows the user to initiate his program upon command of a peripheral device, such as a switch closure, without the activation of the monitor's program control command "G". At any time during the execution of the interrupt-invoked program, the user may re-enter the monitor by executing a RST 1 instruction. The sign-on message "MCS-80 KIT" will be displayed on the console device and all monitor commands are available to the user. To resume the user program, simply input "G" and Carriage Return on the console device.

B. Error Conditions

1. 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 an "*" to indicate the error.

Example

```
D1400, 145G*
```

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

```
Y*
```

Y is not a valid command.

2. 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.

Addresses are evaluated modulo 2^{16}. 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 a part of memory in a command 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.

APPENDIX A. MONITOR LISTING

```
ISIS-II 8080/8085 MACRO ASSEMBLER, X108         SDK80   PAGE    1


  LOC  OBJ         LINE        SOURCE STATEMENT

                      1 ;*************************************************************************
                      2 ;
                      3 ;                 PROGRAM: 8080A BOARD MONITOR
                      4 ;
                      5 ;                 COPYRIGHT (C) 1975
                      6 ;                 INTEL CORPORATION
                      7 ;                 3065 BOWERS AVENUE
                      8 ;                 SANTA CLARA, CALIFORNIA  95051
                      9 ;
                     10 ;*************************************************************************
                     11 ;
                     12 ; ABSTRACT
                     13 ; ========
                     14 ;
                     15 ; THIS PROGRAM RUNS ON THE 8080A BOARD AND IS DESIGNED TO PROVIDE
                     16 ; THE USER WITH A MINIMAL MONITOR.  BY USING THIS PROGRAM,
                     17 ; THE USER CAN EXAMINE AND CHANGE MEMORY OR CPU REGISTERS, LOAD
                     18 ; A PROGRAM (IN ABSOLUTE HEX) INTO RAM, AND EXECUTE INSTRUCTIONS
                     19 ; ALREADY IN MEMORY.  THE MONITOR ALSO PROVIDES THE USER WITH
                     20 ; ROUTINES FOR PERFORMING CONSOLE I/O.
                     21 ;
                     22 ;
                     23 ; PROGRAM ORGANIZATION
                     24 ; ======= ============
                     25 ;
                     26 ; THE LISTING IS ORGANIZED IN THE FOLLOWING WAY. FIRST THE COMMAND
                     27 ; RECOGNIZER, WHICH IS THE HIGHEST LEVEL ROUTINE IN THE PROGRAM.
                     28 ; NEXT, ARE THE ROUTINES TO IMPLEMENT THE VARIOUS COMMANDS, FINALLY
                     29 ; THE UTILITY ROUTINES WHICH ACTUALLY DO THE DIRTY WORK. WITHIN
                     30 ; EACH SECTION, THE ROUTINES ARE ORGANIZED IN ALPHABETICAL
                     31 ; ORDER, BY ENTRY POINT OF THE ROUTINE.
                     32 ;
                     33 ; THIS PROGRAM EXPECTS TO RUN IN THE FIRST 1K OF ADDRESS SPACE.
                     34 ; IF, FOR SOME REASON, THE PROGRAM IS RE-ORG'ED, CARE SHOULD
                     35 ; BE TAKEN TO MAKE SURE THAT THE TRANSFER INSTRUCTIONS FOR RST 1
                     36 ; AND RST 7 ARE ADJUSTED APPROPRIATELY.
                     37 ;
                     38 ; THE PROGRAM ALSO EXPECTS THAT RAM LOCATIONS 5K-1 TO 5K-256,
                     39 ; INCLUSIVE, ARE RESERVED FOR THE PROGRAM'S OWN USE.  THESE
                     40 ; LOCATIONS MAY BE ALTERED, HOWEVER, BY CHANGING THE EQU'ED
                     41 ; SYMBOL "DATA" AS DESIRED.
                     42 ;
                     43 ;
                     44 ; LIST OF FUNCTIONS
                     45 ; ==== == =========
                     46 ;
                     47 ;       GETCM
                     48 ;       -----
                     49 ;
                     50 ;       DCMD
                     51 ;       GCMD
```

```
 LOC  OBJ         LINE        SOURCE STATEMENT

                    52 ;       ICMD
                    53 ;       MCMD
                    54 ;       SCMD
                    55 ;       XCMD
                    56 ;       ----
                    57 ;
                    58 ;       BREAK
                    59 ;       CI
                    60 ;       CNVBN
                    61 ;       CO
                    62 ;       CROUT
                    63 ;       ECHO
                    64 ;       ERROR
                    65 ;       FRET
                    66 ;       GETCH
                    67 ;       GETHX
                    68 ;       GETNM
                    69 ;       HILO
                    70 ;       NMOUT
                    71 ;       PRVAL
                    72 ;       REGDS
                    73 ;       RGADR
                    74 ;       RSTTF
                    75 ;       SRET
                    76 ;       STHF0
                    77 ;       STHLF
                    78 ;       VALDG
                    79 ;       VALDL
                    80 ;       -----
                    81 ;
 0000               82          ORG     0H
                    83 ;
                    84 ;*************************************************************************
                    85 ;
                    86 ;                         MONITOR EQUATES
                    87 ;
                    88 ;*************************************************************************
                    89 ;
                    90 ;
 001B               91 BRCHR    EQU     1BH     ; CODE FOR BREAK CHARACTER (ESCAPE)
 13FD               92 BRLOC    EQU     13FDH   ; LOCATION OF USER BRANCH INSTRUCTION IN RAM
 03FA               93 BRTAB    EQU     3FAH    ; LOCATION FOR START OF BRANCH TABLE IN ROM
 0027               94 CMD      EQU     027H    ; COMMAND INSTRUCTION FOR USART INITIALIZATION
 00FB               95 CNCTL    EQU     0FBH    ; CONSOLE (USART) CONTROL PORT
 00FA               96 CNIN     EQU     0FAH    ; CONSOLE INPUT PORT
 00FA               97 CNOUT    EQU     0FAH    ; CONSOLE OUTPUT PORT
 00FB               98 CONST    EQU     0FBH    ; CONSOLE STATUS INPUT PORT
 000D               99 CR       EQU     0DH     ; CODE FOR CARRIAGE RETURN
 1300              100 DATA     EQU     5*1024-256      ; END OF MONITOR RAM USAGE
 001B              101 ESC      EQU     1BH     ; CODE FOR ESCAPE CHARACTER
 000F              102 HCHAR    EQU     0FH     ; MASK TO SELECT LOWER HEX CHAR FROM BYTE
 00FF              103 INVRT    EQU     0FFH    ; MASK TO INVERT HALF BYTE FLAG
```

```
 LOC  OBJ          LINE        SOURCE STATEMENT

 000A               104 LF      EQU     0AH     ; CODE FOR LINE FEED
 0000               105 LOWER   EQU     0       ; DENOTES LOWER HALF OF BYTE IN ICMD
                    106 ;LSGNON EQU     ---     ; LENGTH OF SIGNON MESSAGE - DEFINED LATER
 00CF               107 MODE    EQU     0CFH    ; MODE SET FOR USART INITIALIZATION
                    108 ;MSTAK  EQU     ---     ; START OF MONITOR STACK - DEFINED LATER
                    109 ;NCMDS  EQU     ---     ; NUMBER OF VALID COMMANDS
 000F               110 NEWLN   EQU     0FH     ; MASK FOR CHECKING MEMORY ADDR DISPLAY
 007F               111 PRTY0   EQU     07FH    ; MASK TO CLEAR PARITY BIT FROM CONSOLE CHAR
 13ED               112 REGS    EQU     DATA+255-18     ; START OF REGISTER SAVE AREA
 0002               113 RBR     EQU     2       ; MASK TO TEST RECEIVER STATUS
 0038               114 RSTU    EQU     38H     ; TRANSFER LOCATION FOR RST7 INSTRUCTION
                    115 ;RTABS  EQU     ---     ; SIZE OF ENTRY IN RTAB TABLE
 001B               116 TERM    EQU     1BH     ; CODE FOR ICMD TERMINATING CHARACTER (ESCAPE)
 0001               117 TRDY    EQU     1       ; MASK TO TEST TRANSMITTER STATUS
 00FF               118 UPPER   EQU     0FFH    ; DENOTES UPPER HALF OF BYTE IN ICMD
                    119 ;
                    120 ;*****************************************************************
                    121 ;
                    122 ;                       MONITOR MACROS
                    123 ;
                    124 ;*****************************************************************
                    125 ;
                    126 ;
                    127 TRUE    MACRO   WHERE   ; BRANCH IF FUNCTION RETURNS TRUE (SUCCESS)
 -                  128         JC      WHERE
                    129         ENDM
                    130 ;
                    131 FALSE   MACRO   WHERE   ; BRANCH IF FUNCTION RETURNS FALSE (FAILURE)
 -                  132         JNC     WHERE
                    133         ENDM
                    134 ;
                    135 ;
                    136 ;*****************************************************************
                    137 ;
                    138 ;                       USART INITIALIZATION CODE
                    139 ;
                    140 ;*****************************************************************
                    141 ;
                    142 ;
                    143 ;   THE USART IS ASSUMED TO COME UP IN THE RESET POSITION (THIS
                    144 ;   FUNCTION IS TAKEN CARE OF BY THE HARDWARE).  THE USART WILL
                    145 ;   BE INITIALIZED IN THE SAME WAY FOR EITHER A TTY OR CRT
                    146 ;   INTERFACE. THE FOLLOWING PARAMETERS ARE USED:
                    147 ;
                    148 ;       MODE INSTRUCTION
                    149 ;       ==== ===========
                    150 ;
                    151 ;       2 STOP BITS
                    152 ;       PARITY DISABLED
                    153 ;       8 BIT CHARACTERS
                    154 ;       BAUD RATE FACTOR OF 64
                    155 ;
```

```
 LOC  OBJ         LINE        SOURCE STATEMENT

                   156 ;      COMMAND INSTRUCTION
                   157 ;      ======= ===========
                   158 ;
                   159 ;      NO HUNT MODE
                   160 ;      NOT(RTS) FORCED TO 0
                   161 ;      RECEIVE ENABLED
                   162 ;      TRANSMIT ENABLED
                   163 ;
 0000 3ECF         164         MVI     A,MODE
 0002 D3FB         165         OUT     CNCTL   ; OUTPUT MODE SET TO USART
 0004 3E27         166         MVI     A,CMD   ;
 0006 D3FB         167         OUT     CNCTL   ; OUTPUT COMMAND WORD TO USART
                   168 ;
                   169 ;*****************************************************************
                   170 ;
                   171 ;                       RESTART ENTRY POINT
                   172 ;
                   173 ;*****************************************************************
                   174 ;
                   175 ;
                   176 GO:
 0008 22F313       177         SHLD    LSAVE   ; SAVE HL REGISTERS
 000B E1           178         POP     H       ; GET TOP OF STACK ENTRY
 000C 22F513       179         SHLD    PSAVE   ; ASSUME THIS IS LAST P COUNTER
 000F 210000       180         LXI     H,0     ; CLEAR HL
 0012 39           181         DAD     SP      ; GET STACK POINTER VALUE
 0013 22F713       182         SHLD    SSAVE   ; SAVE USER`S STACK POINTER
 0016 21F313       183         LXI     H,ASAVE+1       ; NEW VALUE FOR STACK POINTER
 0019 F9           184         SPHL            ; SET MONITOR STACK POINTER FOR REG SAVE
 001A F5           185         PUSH    PSW     ; SAVE A AND FLAGS
 001B C5           186         PUSH    B       ; SAVE B AND C
 001C D5           187         PUSH    D       ; SAVE D AND E
                   188 ;
                   189 ;*************************************************************
                   190 ;
                   191 ;                       PRINT SIGNON MESSAGE
                   192 ;
                   193 ;*************************************************************
                   194 ;
                   195 ;
                   196 SOMSG:
 001D 219D03       197         LXI     H,SGNON ; GET ADDRESS OF SIGNON MESSAGE
 0020 060E         198         MVI     B,LSGNON        ; COUNTER FOR CHARACTERS IN MESSAGE
                   199 MSGL:
 0022 4E           200         MOV     C,M     ; FETCH NEXT CHAR TO C REG
 0023 CDE301       201         CALL    CO      ; SEND IT TO THE CONSOLE
 0026 23           202         INX     H       ; POINT TO NEXT CHARACTER
 0027 05           203         DCR     B       ; DECREMENT BYTE COUNTER
 0028 C22200       204         JNZ     MSGL    ; RETURN FOR NEXT CHARACTER
                   205 ;
                   206 ;
                   207 ;*****************************************************************
```

```
 LOC  OBJ         LINE        SOURCE STATEMENT

                   208 ;
                   209 ;                       COMMAND RECOGNIZING ROUTINE
                   210 ;
                   211 ;*************************************************************
                   212 ;
                   213 ; FUNCTION: GETCM
                   214 ; INPUTS: NONE
                   215 ; OUTPUTS: NONE
                   216 ; CALLS: GETCH,ECHO,ERROR
                   217 ; DESTROYS: A,B,C,H,L,F/F'S
                   218 ; DESCRIPTION: GETCM RECEIVES AN INPUT CHARACTER FROM THE USER
                   219 ;              AND ATTEMPTS TO LOCATE THIS CHARACTER IN ITS COMMAND
                   220 ;              CHARACTER TABLE.  IF SUCCESSFUL, THE ROUTINE
                   221 ;              CORRESPONDING TO THIS CHARACTER IS SELECTED FROM
                   222 ;              A TABLE OF COMMAND ROUTINE ADDRESSES, AND CONTROL
                   223 ;              IS TRANSFERRED TO THIS ROUTINE.  IF THE CHARACTER
                   224 ;              DOES NOT MATCH ANY ENTRIES, CONTROL IS PASSED TO
                   225 ;              THE ERROR HANDLER.
                   226 ;
                   227 GETCM:
  002B 21ED13      228         LXI     H,MSTAK ; ALWAYS WANT TO RESET STACK PTR TO MONITOR
  002E F9          229         SPHL            ; /STARTING VALUE SO ROUTINES NEEDN'T CLEAN UP
  002F 0E2E        230         MVI     C,'.'   ; PROMPT CHARACTER TO C
  0031 CDF401      231         CALL    ECHO    ; SEND PROMPT CHARACTER TO USER TERMINAL
  0034 C33B00      232         JMP     GTC03   ; WANT TO LEAVE ROOM FOR RST BRANCH
                   233 ;
  0038             234         ORG     RSTU    ; ORG TO RST TRANSFER LOCATION
  0038 C3FD13      235         JMP     USRBR   ; JUMP TO USER BRANCH LOCATION
                   236 ;
                   237 GTC03:
  003B CD1B02      238         CALL    GETCH   ; GET COMMAND CHARACTER TO A
  003E CDF401      239         CALL    ECHO    ; ECHO CHARACTER TO USER
  0041 79          240         MOV     A,C     ; PUT COMMAND CHARACTER INTO ACCUMULATOR
  0042 010600      241         LXI     B,NCMDS ; C CONTAINS LOOP AND INDEX COUNT
  0045 21B903      242         LXI     H,CTAB  ; HL POINTS INTO COMMAND TABLE
                   243 GTC05:
  0048 BE          244         CMP     M       ; COMPARE TABLE ENTRY AND CHARACTER
  0049 CA5400      245         JZ      GTC10   ; BRANCH IF EQUAL - COMMAND RECOGNIZED
  004C 23          246         INX     H       ; ELSE, INCREMENT TABLE POINTER
  004D 0D          247         DCR     C       ; DECREMENT LOOP COUNT
  004E C24800      248         JNZ     GTC05   ; BRANCH IF NOT AT TABLE END
  0051 C30D02      249         JMP     ERROR   ; ELSE, COMMAND CHARACTER IS ILLEGAL
                   250 GTC10:
  0054 21AB03      251         LXI     H,CADR  ; IF GOOD COMMAND, LOAD ADDRESS OF TABLE
                   252                         ; /OF COMMAND ROUTINE ADDRESSES
  0057 09          253         DAD     B       ; ADD WHAT IS LEFT OF LOOP COUNT
  0058 09          254         DAD     B       ; ADD AGAIN - EACH ENTRY IN CADR IS 2 BYTES LONG
  0059 7E          255         MOV     A,M     ; GET LSP OF ADDRESS OF TABLE ENTRY TO A
  005A 23          256         INX     H       ; POINT TO NEXT BYTE IN TABLE
  005B 66          257         MOV     H,M     ; GET MSP OF ADDRESS OF TABLE ENTRY TO H
  005C 6F          258         MOV     L,A     ; PUT LSP OF ADDRESS OF TABLE ENTRY INTO L
  005D E9          259         PCHL            ; NEXT INSTRUCTION COMES FROM COMMAND ROUTINE
```

```
 LOC  OBJ         LINE        SOURCE STATEMENT

                   260 ;
                   261 ;
                   262 ;*****************************************************************
                   263 ;
                   264 ;                       COMMAND IMPLEMENTING ROUTINES
                   265 ;
                   266 ;*****************************************************************
                   267 ;
                   268 ;
                   269 ; FUNCTION: DCMD
                   270 ; INPUTS: NONE
                   271 ; OUTPUTS: NONE
                   272 ; CALLS: ECHO,NMOUT,HILO,GETCM,CROUT,GETNM
                   273 ; DESTROYS: A,B,C,D,E,H,L,F/F'S
                   274 ; DESCRIPTION: DCMD IMPLEMENTS THE DISPLAY MEMORY (D) COMMAND
                   275 ;
                   276 DCMD:
 005E 0E02         277         MVI     C,2     ; GET TWO NUMBERS FROM INPUT STREAM
 0060 CD5702       278         CALL    GETNM
 0063 D1           279         POP     D       ; ENDING ADDRESS TO DE
 0064 E1           280         POP     H       ; STARTING ADDRESS TO HL
                   281 DCM05:
 0065 CDEE01       282         CALL    CROUT   ; ECHO CARRIAGE RETURN/LINE FEED
 0068 7C           283         MOV     A,H     ; DISPLAY ADDRESS OF FIRST LOCATION IN LINE
 0069 CDC302       284         CALL    NMOUT   ;
 006C 7D           285         MOV     A,L     ; ADDRESS IS 2 BYTES LONG
 006D CDC302       286         CALL    NMOUT   ;
                   287 DCM10:
 0070 0E20         288         MVI     C,' '
 0072 CDF401       289         CALL    ECHO    ; USE BLANK AS SEPARATOR
 0075 7E           290         MOV     A,M     ; GET CONTENTS OF NEXT MEMORY LOCATION
 0076 CDC302       291         CALL    NMOUT   ; DISPLAY CONTENTS
 0079 CDBD01       292         CALL    BREAK   ; SEE IF USER WANTS OUT
                   293         TRUE    DCM12   ; IF SO, BRANCH
 007C DA8500       294+        JC      DCM12
 007F CD9C02       295         CALL    HILO    ; SEE IF ADDRESS OF DISPLAYED LOCATION IS
                   296                         ; /GREATER THAN OR EQUAL TO ENDING ADDRESS
                   297         FALSE   DCM15   ; IF NOT, MORE TO DISPLAY
 0082 D28B00       298+        JNC     DCM15
                   299 DCM12:
 0085 CDEE01       300         CALL    CROUT   ; CARRIAGE RETURN/LINE FEED TO END LINE
 0088 C32B00       301         JMP     GETCM   ; ALL DONE
                   302 DCM15:
 008B 23           303         INX     H       ; IF MORE TO GO, POINT TO NEXT LOC TO DISPLAY
 008C 7D           304         MOV     A,L     ; GET LOW ORDER BITS OF NEW ADDRESS
 008D E60F         305         ANI     NEWLN   ; SEE IF LAST HEX DIGIT OF ADDRESS DENOTES
                   306                         ; /START OF NEW LINE
 008F C27000       307         JNZ     DCM10   ; NO - NOT AT END OF LINE
 0092 C36500       308         JMP     DCM05   ; YES - START NEW LINE WITH ADDRESS
                   309 ;
                   310 ;
                   311 ;*****************************************************************
```

```
 LOC  OBJ         LINE        SOURCE STATEMENT

                   312 ;
                   313 ;
                   314 ; FUNCTION: GCMD
                   315 ; INPUTS: NONE
                   316 ; OUTPUTS: NONE
                   317 ; CALLS: ERROR,GETHX,RSTTF
                   318 ; DESTROYS: A,B,C,D,E,H,L,F/F'S
                   319 ; DESCRIPTION: GCMD IMPLEMENTS THE BEGIN EXECUTION (G) COMMAND.
                   320 ;
                   321 GCMD:
 0095 CD2202       322         CALL    GETHX   ; GET ADDRESS (IF PRESENT) FROM INPUT STREAM
                   323         FALSE   GCM05   ; BRANCH IF NO NUMBER PRESENT
 0098 D2AA00       324+        JNC     GCM05
 009B 7A           325         MOV     A,D     ; ELSE, GET TERMINATOR
 009C FE0D         326         CPI     CR      ; SEE IF CARRIAGE RETURN
 009E C20D02       327         JNZ     ERROR   ; ERROR IF NOT PROPERLY TERMINATED
 00A1 21F513       328         LXI     H,PSAVE ; WANT NUMBER TO REPLACE SAVE PGM COUNTER
 00A4 71           329         MOV     M,C
 00A5 23           330         INX     H
 00A6 70           331         MOV     M,B
 00A7 C3B000       332         JMP     GCM10
                   333 GCM05:
 00AA 7A           334         MOV     A,D     ; IF NO STARTING ADDRESS, MAKE SURE THAT
 00AB FE0D         335         CPI     CR      ; /CARRIAGE RETURN TERMINATED COMMAND
 00AD C20D02       336         JNZ     ERROR   ; ERROR IF NOT
                   337 GCM10:
 00B0 C32E03       338         JMP     RSTTF   ; RESTORE REGISTERS AND BEGIN EXECUTION
                   339
                   340 ;
                   341 ;
                   342 ;*****************************************************************
                   343 ;
                   344 ;
                   345 ; FUNCTION: ICMD
                   346 ; INPUTS: NONE
                   347 ; OUTPUTS: NONE
                   348 ; CALLS: ERROR,ECHO,GETCH,VALDL,VALDG,CNVBN,STHLF,GETNM,CROUT
                   349 ; DESTROYS: A,B,C,D,E,H,L,F/F'S
                   350 ; DESCRIPTION: ICMD IMPLEMENTS THE INSERT CODE INTO MEMORY (I) COMMAND.
                   351 ;
                   352 ICMD:
 00B3 0E01         353         MVI     C,1
 00B5 CD5702       354         CALL    GETNM   ; GET SINGLE NUMBER FROM INPUT STREAM
 00B8 3EFF         355         MVI     A,UPPER
 00BA 32F913       356         STA     TEMP    ; TEMP WILL HOLD THE UPPER/LOWER HALF BYTE FLAG
 00BD D1           357         POP     D       ; ADDRESS OF START TO DE
                   358 ICM05:
 00BE CD1B02       359         CALL    GETCH   ; GET A CHARACTER FROM INPUT STREAM
 00C1 4F           360         MOV     C,A     ;
 00C2 CDF401       361         CALL    ECHO    ; ECHO IT
 00C5 79           362         MOV     A,C     ; PUT CHARACTER BACK INTO A
 00C6 FE1B         363         CPI     TERM    ; SEE IF CHARACTER IS A TERMINATING CHARACTER
```

```
 LOC  OBJ         LINE        SOURCE STATEMENT

 00C8 CAF400       364         JZ      ICM25   ; IF SO, ALL DONE ENTERING CHARACTERS
 00CB CD8A03       365         CALL    VALDL   ; ELSE, SEE IF VALID DELIMITER
                   366         TRUE    ICM05   ; IF SO SIMPLY IGNORE THIS CHARACTER
 00CE DABE00       367+        JC      ICM05
 00D1 CD6F03       368         CALL    VALDG   ; ELSE, CHECK TO SEE IF VALID HEX DIGIT
                   369         FALSE   ICM20   ; IF NOT, BRANCH TO HANDLE ERROR CONDITION
 00D4 D2EE00       370+        JNC     ICM20
 00D7 CDDA01       371         CALL    CNVBN   ; CONVERT DIGIT TO BINARY
 00DA 4F           372         MOV     C,A     ; MOVE RESULT TO C
 00DB CD5003       373         CALL    STHLF   ; STORE IN APPROPRIATE HALF WORD
 00DE 3AF913       374         LDA     TEMP    ; GET HALF BYTE FLAG
 00E1 B7           375         ORA     A       ; SET F/F'S
 00E2 C2E600       376         JNZ     ICM10   ; BRANCH IF FLAG SET FOR UPPER
 00E5 13           377         INX     D       ; IF LOWER, INC ADDRESS OF BYTE TO STORE IN
                   378 ICM10:
 00E6 EEFF         379         XRI     INVRT   ; TOGGLE STATE OF FLAG
 00E8 32F913       380         STA     TEMP    ; PUT NEW VALUE OF FLAG BACK
 00EB C3BE00       381         JMP     ICM05   ; PROCESS NEXT DIGIT
                   382 ICM20:
 00EE CD4503       383         CALL    STHF0   ; ILLEGAL CHARACTER
 00F1 C30D02       384         JMP     ERROR   ; MAKE SURE ENTIRE BYTE FILLED THEN ERROR
                   385 ICM25:
 00F4 CD4503       386         CALL    STHF0   ; HERE FOR ESCAPE CHARACTER - INPUT IS DONE
 00F7 CDEE01       387         CALL    CROUT   ; ADD CARRIAGE RETURN
 00FA C32B00       388         JMP     GETCM   ;
                   389 ;
                   390 ;
                   391 ;*****************************************************************
                   392 ;
                   393 ;
                   394 ; FUNCTION: MCMD
                   395 ; INPUTS: NONE
                   396 ; OUTPUTS: NONE
                   397 ; CALLS: GETCM,HILO,GETNM
                   398 ; DESTROYS: A,B,C,D,E,H,L,F/F'S
                   399 ; DESCRIPTION: MCMD IMPLEMENTS THE MOVE DATA IN MEMORY (M) COMMAND.
                   400 ;
                   401 MCMD:
 00FD 0E03         402         MVI     C,3     ;
 00FF CD5702       403         CALL    GETNM   ; GET 3 NUMBERS FROM INPUT STREAM
 0102 C1           404         POP     B       ; DESTINATION ADDRESS TO BC
 0103 E1           405         POP     H       ; ENDING ADDRESS TO HL
 0104 D1           406         POP     D       ; STARTING ADDRESS TO DE
                   407 MCM05:
 0105 E5           408         PUSH    H       ; SAVE ENDING ADDRESS
 0106 62           409         MOV     H,D
 0107 6B           410         MOV     L,E     ; SOURCE ADDRESS TO HL
 0108 7E           411         MOV     A,M     ; GET SOURCE BYTE
 0109 60           412         MOV     H,B
 010A 69           413         MOV     L,C     ; DESTINATION ADDRESS TO HL
 010B 77           414         MOV     M,A     ; MOVE BYTE TO DESTINATION
 010C 03           415         INX     B       ; INCREMENT DESTINATION ADDRESS
```

```
 LOC  OBJ         LINE        SOURCE STATEMENT

 010D 78           416         MOV     A,B
 010E B1           417         ORA     C       ; TEST FOR DESTINATION ADDRESS OVERFLOW
 010F CA2B00       418         JZ      GETCM   ; IF SO, CAN TERMINATE COMMAND
 0112 13           419         INX     D       ; INCREMENT SOURCE ADDRESS
 0113 E1           420         POP     H       ; ELSE, GET BACK ENDING ADDRESS
 0114 CD9C02       421         CALL    HILO    ; SEE IF ENDING ADDR>=SOURCE ADDR
                   422         FALSE   GETCM   ; IF NOT, COMMAND IS DONE
 0117 D22B00       423+        JNC     GETCM
 011A C30501       424         JMP     MCM05   ; MOVE ANOTHER BYTE
                   425 ;
                   426 ;
                   427 ;*****************************************************************
                   428 ;
                   429 ;
                   430 ; FUNCTION: SCMD
                   431 ; INPUTS: NONE
                   432 ; OUTPUTS: NONE
                   433 ; CALLS: GETHX,GETCM,NMOUT,ECHO
                   434 ; DESTROYS: A,B,C,D,E,H,L,F/F'S
                   435 ; DESCRIPTION: SCMD IMPLEMENTS THE SUBSTITUTE INTO MEMORY (S) COMMAND.
                   436 ;
                   437 SCMD:
 011D CD2202       438         CALL    GETHX   ; GET A NUMBER, IF PRESENT, FROM INPUT
 0120 C5           439         PUSH    B
 0121 E1           440         POP     H       ; GET NUMBER TO HL - DENOTES MEMORY LOCATION
                   441 SCM05:
 0122 7A           442         MOV     A,D     ; GET TERMINATOR
 0123 FE20         443         CPI     ' '     ; SEE IF SPACE
 0125 CA2D01       444         JZ      SCM10   ; YES - CONTINUE PROCESSING
 0128 FE2C         445         CPI     ','     ; ELSE, SEE IF COMMA
 012A C22B00       446         JNZ     GETCM   ; NO - TERMINATE COMMAND
                   447 SCM10:
 012D 7E           448         MOV     A,M     ; GET CONTENTS OF SPECIFIED LOCATION TO A
 012E CDC302       449         CALL    NMOUT   ; DISPLAY CONTENTS ON CONSOLE
 0131 0E2D         450         MVI     C,'-'
 0133 CDF401       451         CALL    ECHO    ; USE DASH FOR SEPARATOR
 0136 CD2202       452         CALL    GETHX   ; GET NEW VALUE FOR MEMORY LOCATION, IF ANY
                   453         FALSE   SCM15   ; IF NO VALUE PRESENT, BRANCH
 0139 D23D01       454+        JNC     SCM15
 013C 71           455         MOV     M,C     ; ELSE, STORE LOWER 8 BITS OF NUMBER ENTERED
                   456 SCM15:
 013D 23           457         INX     H       ; INCREMENT ADDRESS OF MEMORY LOCATION TO VIEW
 013E C32201       458         JMP     SCM05
                   459 ;
                   460 ;
                   461 ;*****************************************************************
                   462 ;
                   463 ;
                   464 ; FUNCTION: XCMD
                   465 ; INPUTS: NONE
                   466 ; OUTPUTS: NONE
                   467 ; CALLS: GETCH,ECHO,REGDS,GETCM,ERROR,RGADR,NMOUT,CROUT,GETHX
```

```
 LOC  OBJ         LINE        SOURCE STATEMENT

                   468 ; DESTROYS: A,B,C,D,E,H,L,F/F'S
                   469 ; DESCRIPTION: XCMD IMPLEMENTS THE REGISTER EXAMINE AND CHANGE (X)
                   470 ;              COMMAND.
                   471 ;
                   472 XCMD:
  0141 CD1B02      473          CALL    GETCH   ; GET REGISTER IDENTIFIER
  0144 4F          474          MOV     C,A     ;
  0145 CDF401      475          CALL    ECHO    ; ECHO IT
  0148 79          476          MOV     A,C
  0149 FE0D        477          CPI     CR
  014B C25401      478          JNZ     XCM05   ; BRANCH IF NOT CARRIAGE RETURN
  014E CDE602      479          CALL    REGDS   ; ELSE, DISPLAY REGISTER CONTENTS
  0151 C32B00      480          JMP     GETCM   ; THEN TERMINATE COMMAND
                   481 XCM05:
  0154 4F          482          MOV     C,A     ; GET REGISTER IDENTIFIER TO C
  0155 CD1703      483          CALL    RGADR   ; CONVERT IDENTIFIER INTO RTAB TABLE ADDR
  0158 C5          484          PUSH    B
  0159 E1          485          POP     H       ; PUT POINTER TO REGISTER ENTRY INTO HL
  015A 0E20        486          MVI     C,' '
  015C CDF401      487          CALL    ECHO    ; ECHO SPACE TO USER
  015F 79          488          MOV     A,C
  0160 32F913      489          STA     TEMP    ; PUT SPACE INTO TEMP AS DELIMITER
                   490 XCM10:
  0163 3AF913      491          LDA     TEMP    ; GET TERMINATOR
  0166 FE20        492          CPI     ' '     ; SEE IF A BLANK
  0168 CA7001      493          JZ      XCM15   ; YES - GO CHECK POINTER INTO TABLE
  016B FE2C        494          CPI     ','     ; NO - SEE IF COMMA
  016D C22B00      495          JNZ     GETCM   ; NO - MUST BE CARRIAGE RETURN TO END COMMAND
                   496 XCM15:
  0170 7E          497          MOV     A,M
  0171 B7          498          ORA     A       ; SET F/F'S
  0172 C27B01      499          JNZ     XCM18   ; BRANCH IF NOT AT END OF TABLE
  0175 CDEE01      500          CALL    CROUT   ; ELSE, OUTPUT CARRIAGE RETURN LINE FEED
  0178 C32B00      501          JMP     GETCM   ; AND EXIT
                   502 XCM18:
  017B E5          503          PUSH    H       ; PUT POINTER ON STACK
  017C 5E          504          MOV     E,M
  017D 1613        505          MVI     D,DATA SHR 8    ; FETCH ADDRESS OF SAVE LOCATION FROM
TABLE
  017F 23          506          INX     H               ;
  0180 46          507          MOV     B,M     ; FETCH LENGTH FLAG FROM TABLE
  0181 D5          508          PUSH    D       ; SAVE ADDRESS OF SAVE LOCATION
  0182 D5          509          PUSH    D
  0183 E1          510          POP     H       ; MOVE ADDRESS TO HL
  0184 C5          511          PUSH    B       ; SAVE LENGTH FLAG
  0185 7E          512          MOV     A,M     ; GET 8 BITS OF REGISTER FROM SAVE LOCATION
  0186 CDC302      513          CALL    NMOUT   ; DISPLAY IT
  0189 F1          514          POP     PSW     ; GET BACK LENGTH FLAG
  018A F5          515          PUSH    PSW     ; SAVE IT AGAIN
  018B B7          516          ORA     A       ; SET F/F'S
  018C CA9401      517          JZ      XCM20   ; IF 8 BIT REGISTER, NOTHING MORE TO DISPLAY
  018F 2B          518          DCX     H       ; ELSE, FOR 16 BIT REGISTER, GET LOWER 8 BITS
  0190 7E          519          MOV     A,M
```

```
 LOC  OBJ         LINE        SOURCE STATEMENT

 0191 CDC302       520         CALL    NMOUT   ; DISPLAY THEM
                   521 XCM20:
 0194 0E2D         522         MVI     C,'-'
 0196 CDF401       523         CALL    ECHO    ; USE DASH AS SEPARATOR
 0199 CD2202       524         CALL    GETHX   ; SEE IF THERE IS A VALUE TO PUT INTO REGISTER
                   525         FALSE   XCM30   ; NO - GO CHECK FOR NEXT REGISTER
 019C D2B401       526+        JNC     XCM30
 019F 7A           527         MOV     A,D     ;
 01A0 32F913       528         STA     TEMP    ; ELSE, SAVE THE TERMINATOR FOR NOW
 01A3 F1           529         POP     PSW     ; GET BACK LENGTH FLAG
 01A4 E1           530         POP     H       ; PUT ADDRESS OF SAVE LOCATION INTO HL
 01A5 B7           531         ORA     A       ; SET F/F'S
 01A6 CAAB01       532         JZ      XCM25   ; IF 8 BIT REGISTER, BRANCH
 01A9 70           533         MOV     M,B     ; SAVE UPPER 8 BITS
 01AA 2B           534         DCX     H       ; POINT TO SAVE LOCATION FOR LOWER 8 BITS
                   535 XCM25:
 01AB 71           536         MOV     M,C     ; STORE ALL OF 8 BIT OR LOWER 1/2 OF 16 BIT REG
                   537 XCM27:
 01AC 110300       538         LXI     D,RTABS ; SIZE OF ENTRY IN RTAB TABLE
 01AF E1           539         POP     H       ; POINTER INTO REGISTER TABLE RTAB
 01B0 19           540         DAD     D       ; ADD ENTRY SIZE TO POINTER
 01B1 C36301       541         JMP     XCM10   ; DO NEXT REGISTER
                   542 XCM30:
 01B4 7A           543         MOV     A,D     ; GET TERMINATOR
 01B5 32F913       544         STA     TEMP    ; SAVE IN MEMORY
 01B8 D1           545         POP     D       ; CLEAR STACK OF LENGTH FLAG AND ADDRESS
 01B9 D1           546         POP     D       ; /OF SAVE LOCATION
 01BA C3AC01       547         JMP     XCM27   ; GO INCREMENT REGISTER TABLE POINTER
                   548 ;
                   549 ;
                   550 ;*****************************************************************
                   551 ;
                   552 ;                       UTILITY ROUTINES
                   553 ;
                   554 ;*****************************************************************
                   555 ;
                   556 ;
                   557 ; FUNCTION: BREAK
                   558 ; INPUTS: NONE
                   559 ; OUTPUTS: CARRY - 1 IF ESCAPE CHARACTER INPUT
                   560 ;                - 0 IF ANY OTHER CHARACTER OR NO CHARACTER PENDING
                   561 ; CALLS: NOTHING
                   562 ; DESTROYS: A,F/F'S
                   563 ; DESCRIPTION: BREAK IS USED TO SENSE AN ESCAPE CHARACTER FROM
                   564 ;              THE USER.  IF NO CHARACTER IS PENDING, OR IF THE
                   565 ;              PENDING CHARACTER IS NOT THE ESCAPE, THEN A FAILURE
                   566 ;              RETURN (CARRY=0) IS TAKEN.  IN THIS CASE, THE
                   567 ;              PENDING CHARACTER (IF ANY) IS LOST.  IF THE PENDING
                   568 ;              CHARACTER IS AN ESCAPE CHARACTER, BREAK TAKES A SUCCESS
                   569 ;              RETURN (CARRY=1).
                   570 ;
                   571 BREAK:
```

```
 LOC  OBJ         LINE        SOURCE STATEMENT

 01BD DBFB         572         IN      CONST   ; GET CONSOLE STATUS
 01BF E602         573         ANI     RBR     ; SEE IF CHARACTER PENDING
 01C1 CA1802       574         JZ      FRET    ; NO - TAKE FAILURE RETURN
 01C4 DBFA         575         IN      CNIN    ; YES - PICK UP CHARACTER
 01C6 E67F         576         ANI     PRTY0   ; STRIP OFF PARITY BIT
 01C8 FE1B         577         CPI     BRCHR   ; SEE IF BREAK CHARACTER
 01CA CA4303       578         JZ      SRET    ; YES - SUCCESS RETURN
 01CD C31802       579         JMP     FRET    ; NO - FAILURE RETURN - CHARACTER LOST
                   580 ;
                   581 ;
                   582 ;*****************************************************************
                   583 ;
                   584 ;
                   585 ; FUNCTION: CI
                   586 ; INPUTS: NONE
                   587 ; OUTPUTS: A - CHARACTER FROM CONSOLE
                   588 ; CALLS: NOTHING
                   589 ; DESTROYS: A,F/F'S
                   590 ; DESCRIPTION: CI WAITS UNTIL A CHARACTER HAS BEEN ENTERED AT THE
                   591 ;              CONSOLE AND THEN RETURNS THE CHARACTER, VIA THE A
                   592 ;              REGISTER, TO THE CALLING ROUTINE.  THIS ROUTINE
                   593 ;              IS CALLED BY THE USER VIA A JUMP TABLE IN RAM.
                   594 ;
                   595 CI:
 01D0 DBFB         596         IN      CONST   ; GET STATUS OF CONSOLE
 01D2 E602         597         ANI     RBR     ; CHECK FOR RECEIVER BUFFER READY
 01D4 CAD001       598         JZ      CI      ; NOT YET - WAIT
 01D7 DBFA         599         IN      CNIN    ; READY SO GET CHARACTER
 01D9 C9           600         RET
                   601 ;
                   602 ;
                   603 ;*****************************************************************
                   604 ;
                   605 ;
                   606 ; FUNCTION: CNVBN
                   607 ; INPUTS: C - ASCII CHARACTER '0'-'9' OR 'A'-'F'
                   608 ; OUTPUTS: A - 0 TO F HEX
                   609 ; CALLS: NOTHING
                   610 ; DESTROYS: A,F/F'S
                   611 ; DESCRIPTION: CNVBN CONVERTS THE ASCII REPRESENTATION OF A HEX
                   612 ;              CHARACTER INTO ITS CORRESPONDING BINARY VALUE.  CNVBN
                   613 ;              DOES NOT CHECK THE VALIDITY OF ITS INPUT.
                   614 ;
                   615 CNVBN:
 01DA 79           616         MOV     A,C
 01DB D630         617         SUI     '0'     ; SUBTRACT CODE FOR '0' FROM ARGUMENT
 01DD FE0A         618         CPI     10      ; WANT TO TEST FOR RESULT OF 0 TO 9
 01DF F8           619         RM              ; IF SO, THEN ALL DONE
 01E0 D607         620         SUI     7       ; ELSE, RESULT BETWEEN 17 AND 23 DECIMAL
 01E2 C9           621         RET             ; SO RETURN AFTER SUBTRACTING BIAS OF 7
                   622 ;
                   623 ;
```

```
 LOC  OBJ         LINE        SOURCE STATEMENT

                   624 ;*****************************************************************
                   625 ;
                   626 ;
                   627 ; FUNCTION: CO
                   628 ; INPUTS: C - CHARACTER TO OUTPUT TO CONSOLE
                   629 ; OUTPUTS: C - CHARACTER OUTPUT TO CONSOLE
                   630 ; CALLS: NOTHING
                   631 ; DESTROYS: A,F/F'S
                   632 ; DESCRIPTION: CO WAITS UNTIL THE CONSOLE IS READY TO ACCEPT A CHARACTER
                   633 ;              AND THEN SENDS THE INPUT ARGUMENT TO THE CONSOLE.
                   634 ;
                   635 CO:
 01E3 DBFB         636         IN      CONST   ; GET STATUS OF CONSOLE
 01E5 E601         637         ANI     TRDY    ; SEE IF TRANSMITTER READY
 01E7 CAE301       638         JZ      CO      ; NO - WAIT
 01EA 79           639         MOV     A,C     ; ELSE, MOVE CHARACTER TO A REGISTER FOR OUTPUT
 01EB D3FA         640         OUT     CNOUT   ; SEND TO CONSOLE
 01ED C9           641         RET
                   642 ;
                   643 ;
                   644 ;*****************************************************************
                   645 ;
                   646 ;
                   647 ; FUNCTION CROUT
                   648 ; INPUTS: NONE
                   649 ; OUTPUTS: NONE
                   650 ; CALLS: ECHO
                   651 ; DESTROYS: A,B,C,F/F'S
                   652 ; DESCRIPTION: CROUT SENDS A CARRIAGE RETURN (AND HENCE A LINE
                   653 ;              FEED) TO THE CONSOLE.
                   654 ;
                   655 CROUT:
 01EE 0E0D         656         MVI     C,CR
 01F0 CDF401       657         CALL    ECHO    ; OUTPUT CARRIAGE RETURN TO USER TERMINAL
 01F3 C9           658         RET
                   659 ;
                   660 ;
                   661 ;*****************************************************************
                   662 ;
                   663 ;
                   664 ; FUNCTION: ECHO
                   665 ; INPUTS: C - CHARACTER TO ECHO TO TERMINAL
                   666 ; OUTPUTS: C - CHARACTER ECHOED TO TERMINAL
                   667 ; CALLS: CO
                   668 ; DESTROYS: A,B,F/F'S
                   669 ; DESCRIPTION: ECHO TAKES A SINGLE CHARACTER AS INPUT AND, VIA
                   670 ;              THE MONITOR, SENDS THAT CHARACTER TO THE USER
                   671 ;              TERMINAL.  A CARRIAGE RETURN IS ECHOED AS A CARRIAGE
                   672 ;              RETURN LINE FEED, AND AN ESCAPE CHARACTER IS ECHOED AS $.
                   673 ;
                   674 ECHO:
 01F4 41           675         MOV     B,C     ; SAVE ARGUMENT
```

```
 LOC  OBJ         LINE        SOURCE STATEMENT

 01F5 3E1B         676         MVI     A,ESC
 01F7 B8           677         CMP     B       ; SEE IF ECHOING AN ESCAPE CHARACTER
 01F8 C2FD01       678         JNZ     ECH05   ; NO - BRANCH
 01FB 0E24         679         MVI     C,'$'   ; YES - ECHO AS $
                   680 ECH05:
 01FD CDE301       681         CALL    CO      ; DO OUTPUT THROUGH MONITOR
 0200 3E0D         682         MVI     A,CR
 0202 B8           683         CMP     B       ; SEE IF CHARACTER ECHOED WAS A CARRIAGE RETURN
 0203 C20B02       684         JNZ     ECH10   ; NO - NO NEED TO TAKE SPECIAL ACTION
 0206 0E0A         685         MVI     C,LF    ; YES - WANT TO ECHO LINE FEED, TOO
 0208 CDE301       686         CALL    CO
                   687 ECH10:
 020B 48           688         MOV     C,B     ; RESTORE ARGUMENT
 020C C9           689         RET
                   690 ;
                   691 ;
                   692 ;*****************************************************************
                   693 ;
                   694 ;
                   695 ; FUNCTION: ERROR
                   696 ; INPUTS: NONE
                   697 ; OUTPUTS: NONE
                   698 ; CALLS: ECHO,CROUT,GETCM
                   699 ; DESTROYS: A,B,C,F/F'S
                   700 ; DESCRIPTION: ERROR PRINTS THE ERROR CHARACTER (CURRENTLY A NUMBER SIGN)
                   701 ;              ON THE CONSOLE, FOLLOWED BY A CARRIAGE RETURN-LINE FEED,
                   702 ;              AND THEN RETURNS CONTROL TO THE COMMAND RECOGNIZER.
                   703 ;
                   704 ERROR:
 020D 0E2A         705         MVI     C,'*'
 020F CDF401       706         CALL    ECHO    ; SEND # TO CONSOLE
                   707 EXIT:
 0212 CDEE01       708         CALL    CROUT   ; SKIP TO BEGINNING OF NEXT LINE
 0215 C32B00       709         JMP     GETCM   ; TRY AGAIN FOR ANOTHER COMMAND
                   710 ;
                   711 ;
                   712 ;*****************************************************************
                   713 ;
                   714 ;
                   715 ; FUNCTION: FRET
                   716 ; INPUTS: NONE
                   717 ; OUTPUTS: CARRY - ALWAYS 0
                   718 ; CALLS: NOTHING
                   719 ; DESTROYS: CARRY
                   720 ; DESCRIPTION: FRET IS JUMPED TO BY ANY ROUTINE THAT WISHES TO
                   721 ;              INDICATE FAILURE ON RETURN.  FRET SETS THE CARRY
                   722 ;              FALSE, DENOTING FAILURE, AND THEN RETURNS TO THE
                   723 ;              CALLER OF THE ROUTINE INVOKING FRET.
                   724 ;
                   725 FRET:
 0218 37           726         STC             ; FIRST SET CARRY TRUE
 0219 3F           727         CMC             ; THEN COMPLEMENT IT TO MAKE IT FALSE
```

```
 LOC  OBJ         LINE        SOURCE STATEMENT

 021A C9           728         RET             ; RETURN APPROPRIATELY
                   729 ;
                   730 ;
                   731 ;*****************************************************************
                   732 ;
                   733 ;
                   734 ; FUNCTION: GETCH
                   735 ; INPUTS: NONE
                   736 ; OUTPUTS: C - NEXT CHARACTER IN INPUT STREAM
                   737 ; CALLS: CI
                   738 ; DESTROYS: A,C,F/F'S
                   739 ; DESCRIPTION: GETCH RETURNS THE NEXT CHARACTER IN THE INPUT STREAM
                   740 ;              TO THE CALLING PROGRAM.
                   741 ;
                   742 GETCH:
 021B CDD001       743         CALL    CI      ; GET CHARACTER FROM TERMINAL
 021E E67F         744         ANI     PRTY0   ; TURN OFF PARITY BIT IN CASE SET BY CONSOLE
 0220 4F           745         MOV     C,A     ; PUT VALUE IN C REGISTER FOR RETURN
 0221 C9           746         RET
                   747 ;
                   748 ;
                   749 ;*****************************************************************
                   750 ;
                   751 ;
                   752 ; FUNCTION: GETHX
                   753 ; INPUTS: NONE
                   754 ; OUTPUTS: BC - 16 BIT INTEGER
                   755 ;          D - CHARACTER WHICH TERMINATED THE INTEGER
                   756 ;          CARRY - 1 IF FIRST CHARACTER NOT DELIMITER
                   757 ;                - 0 IF FIRST CHARACTER IS DELIMITER
                   758 ; CALLS: GETCH,ECHO,VALDL,VALDG,CNVBN,ERROR
                   759 ; DESTROYS: A,B,C,D,E,F/F'S
                   760 ; DESCRIPTION: GETHX ACCEPTS A STRING OF HEX DIGITS FROM THE INPUT
                   761 ;              STREAM AND RETURNS THEIR VALUE AS A 16 BIT BINARY
                   762 ;              INTEGER.  IF MORE THAN 4 HEX DIGITS ARE ENTERED,
                   763 ;              ONLY THE LAST 4 ARE USED.  THE NUMBER TERMINATES WHEN
                   764 ;              A VALID DELIMITER IS ENCOUNTERED.  THE DELIMITER IS
                   765 ;              ALSO RETURNED AS AN OUTPUT OF THE FUNCTION.  ILLEGAL
                   766 ;              CHARACTERS (NOT HEX DIGITS OR DELIMITERS) CAUSE AN
                   767 ;              ERROR INDICATION.  IF THE FIRST (VALID) CHARACTER
                   768 ;              ENCOUNTERED IN THE INPUT STREAM IS NOT A DELIMITER,
                   769 ;              GETHX WILL RETURN WITH THE CARRY BIT SET TO 1;
                   770 ;              OTHERWISE, THE CARRY BIT IS SET TO 0 AND THE CONTENTS
                   771 ;              OF BC ARE UNDEFINED.
                   772 ;
                   773 GETHX:
 0222 E5           774         PUSH    H       ; SAVE HL
 0223 210000       775         LXI     H,0     ; INITIALIZE RESULT
 0226 1E00         776         MVI     E,0     ; INITIALIZE DIGIT FLAG TO FALSE
                   777 GHX05:
 0228 CD1B02       778         CALL    GETCH   ; GET A CHARACTER
 022B 4F           779         MOV     C,A     ;
```

```
 LOC  OBJ         LINE        SOURCE STATEMENT

 022C CDF401       780         CALL    ECHO    ; ECHO THE CHARACTER
 022F CD8A03       781         CALL    VALDL   ; SEE IF DELIMITER
                   782         FALSE   GHX10   ; NO - BRANCH
 0232 D24102       783+        JNC     GHX10
 0235 51           784         MOV     D,C     ; YES - ALL DONE, BUT WANT TO RETURN DELIMITER
 0236 E5           785         PUSH    H
 0237 C1           786         POP     B       ; MOVE RESULT TO BC
 0238 E1           787         POP     H       ; RESTORE HL
 0239 7B           788         MOV     A,E     ; GET FLAG
 023A B7           789         ORA     A       ; SET F/F'S
 023B C24303       790         JNZ     SRET    ; IF FLAG NON-0, A NUMBER HAS BEEN FOUND
 023E CA1802       791         JZ      FRET    ; ELSE, DELIMITER WAS FIRST CHARACTER
                   792 GHX10:
 0241 CD6F03       793         CALL    VALDG   ; IF NOT DELIMITER, SEE IF DIGIT
                   794         FALSE   ERROR   ; ERROR IF NOT A VALID DIGIT, EITHER
 0244 D20D02       795+        JNC     ERROR
 0247 CDDA01       796         CALL    CNVBN   ; CONVERT DIGIT TO ITS BINARY VALUE
 024A 1EFF         797         MVI     E,0FFH  ; SET DIGIT FLAG NON-0
 024C 29           798         DAD     H       ; *2
 024D 29           799         DAD     H       ; *4
 024E 29           800         DAD     H       ; *8
 024F 29           801         DAD     H       ; *16
 0250 0600         802         MVI     B,0     ; CLEAR UPPER 8 BITS OF BC PAIR
 0252 4F           803         MOV     C,A     ; BINARY VALUE OF CHARACTER INTO C
 0253 09           804         DAD     B       ; ADD THIS VALUE TO PARTIAL RESULT
 0254 C32802       805         JMP     GHX05   ; GET NEXT CHARACTER
                   806 ;
                   807 ;
                   808 ;*****************************************************************
                   809 ;
                   810 ;
                   811 ; FUNCTION: GETNM
                   812 ; INPUTS: C - COUNT OF NUMBERS TO FIND IN INPUT STREAM
                   813 ; OUTPUTS: TOP OF STACK - NUMBERS FOUND IN REVERSE ORDER (LAST ON TOP
                   814 ;                         OF STACK)
                   815 ; CALLS: GETHX,HILO,ERROR
                   816 ; DESTROYS: A,B,C,D,E,H,L,F/F'S
                   817 ; DESCRIPTION: GETNM FINDS A SPECIFIED COUNT OF NUMBERS, BETWEEN 1
                   818 ;              AND 3, INCLUSIVE,  IN THE INPUT
                   819 ;              STREAM AND RETURNS THEIR VALUES ON THE STACK.  IF 2
                   820 ;              OR MORE NUMBERS ARE REQUESTED, THEN THE FIRST MUST BE
                   821 ;              LESS THAN OR EQUAL TO THE SECOND, OR THE FIRST AND
                   822 ;              SECOND NUMBERS WILL BE SET EQUAL.  THE LAST NUMBER
                   823 ;              REQUESTED MUST BE TERMINATED BY A CARRIAGE RETURN
                   824 ;              OR AN ERROR INDICATION WILL RESULT.
                   825 ;
                   826 GETNM:
 0257 2E03         827         MVI     L,3     ; PUT MAXIMUM ARGUMENT COUNT INTO L
 0259 79           828         MOV     A,C     ; GET THE ACTUAL ARGUMENT COUNT
 025A E603         829         ANI     3       ; FORCE TO MAXIMUM OF 3
 025C C8           830         RZ              ; IF 0, DON'T BOTHER TO DO ANYTHIING
 025D 67           831         MOV     H,A     ; ELSE, PUT ACTUAL COUNT INTO H
```

```
 LOC  OBJ         LINE        SOURCE STATEMENT

                   832 GNM05:
 025E CD2202       833         CALL    GETHX   ; GET A NUMBER FROM INPUT STREAM
                   834         FALSE   ERROR   ; ERROR IF NOT THERE - TOO FEW NUMBERS
 0261 D20D02       835+        JNC     ERROR
 0264 C5           836         PUSH    B       ; ELSE, SAVE NUMBER ON STACK
 0265 2D           837         DCR     L       ; DECREMENT MAXIMUM ARGUMENT COUNT
 0266 25           838         DCR     H       ; DECREMENT ACTUAL ARGUMENT COUNT
 0267 CA7302       839         JZ      GNM10   ; BRANCH IF NO MORE NUMBERS WANTED
 026A 7A           840         MOV     A,D     ; ELSE, GET NUMBER TERMINATOR TO A
 026B FE0D         841         CPI     CR      ; SEE IF CARRIAGE RETURN
 026D CA0D02       842         JZ      ERROR   ; ERROR IF SO - TOO FEW NUMBERS
 0270 C35E02       843         JMP     GNM05   ; ELSE, PROCESS NEXT NUMBER
                   844 GNM10:
 0273 7A           845         MOV     A,D     ; WHEN COUNT 0, CHECK LAST TERMINATOR
 0274 FE0D         846         CPI     CR
 0276 C20D02       847         JNZ     ERROR   ; ERROR IF NOT CARRIAGE RETURN
 0279 01FFFF       848         LXI     B,0FFFFH        ; HL GETS LARGEST NUMBER
 027C 7D           849         MOV     A,L     ; GET WHAT'S LEFT OF MAXIMUM ARG COUNT
 027D B7           850         ORA     A       ; CHECK FOR 0
 027E CA8602       851         JZ      GNM20   ; IF YES, 3 NUMBERS WERE INPUT
                   852 GNM15:
 0281 C5           853         PUSH    B       ; IF NOT, FILL REMAINING ARGUMENTS WITH 0FFFFH
 0282 2D           854         DCR     L
 0283 C28102       855         JNZ     GNM15
                   856 GNM20:
 0286 C1           857         POP     B       ; GET THE 3 ARGUMENTS OUT
 0287 D1           858         POP     D
 0288 E1           859         POP     H
 0289 CD9C02       860         CALL    HILO    ; SEE IF FIRST >= SECOND
                   861         FALSE   GNM25   ; NO - BRANCH
 028C D29102       862+        JNC     GNM25
 028F 54           863         MOV     D,H
 0290 5D           864         MOV     E,L     ; YES - MAKE SECOND EQUAL TO THE FIRST
                   865 GNM25:
 0291 E3           866         XTHL            ; PUT FIRST ON STACK - GET RETURN ADDR
 0292 D5           867         PUSH    D       ; PUT SECOND ON STACK
 0293 C5           868         PUSH    B       ; PUT THIRD ON STACK
 0294 E5           869         PUSH    H       ; PUT RETURN ADDRESS ON STACK
                   870 GNM30:
 0295 3D           871         DCR     A       ; DECREMENT RESIDUAL COUNT
 0296 F8           872         RM              ; IF NEGATIVE, PROPER RESULTS ON STACK
 0297 E1           873         POP     H       ; ELSE, GET RETURN ADDR
 0298 E3           874         XTHL            ; REPLACE TOP RESULT WITH RETURN ADDR
 0299 C39502       875         JMP     GNM30   ; TRY AGAIN
                   876 ;
                   877 ;
                   878 ;*****************************************************************
                   879 ;
                   880 ;
                   881 ; FUNCTION: HILO
                   882 ; INPUTS: DE - 16 BIT INTEGER
                   883 ;         HL - 16 BIT INTEGER
```

```
 LOC  OBJ         LINE        SOURCE STATEMENT

                   884 ; OUTPUTS: CARRY - 0 IF HL<DE
                   885 ;                - 1 IF HL>=DE
                   886 ; CALLS: NOTHING
                   887 ; DESTROYS: A,F/F'S
                   888 ; DESCRIPTION: HILO COMPARES THE 2 16 BIT INTEGERS IN HL AND DE.  THE
                   889 ;              INTEGERS ARE TREATED AS UNSIGNED NUMBERS.  THE CARRY
                   890 ;              BIT IS SET ACCORDING TO THE RESULT OF THE COMPARISON.
                   891 ;
                   892 HILO:
 029C C5           893         PUSH    B       ; SAVE BC
 029D 47           894         MOV     B,A     ; SAVE A REGISTER
 029E 23           895         INX     H       ; INCREMENT HL BY 1
 029F 7C           896         MOV     A,H     ; WANT TO TEST FOR 0 RESULT AFTER
 02A0 B5           897         ORA     L       ; /INCREMENTING
 02A1 CABD02       898         JZ      HIL05   ; WE`RE AUTOMATICALLY DONE IF IT IS
 02A4 23           899         INX     H       ; INCREMENT HL BY 1
 02A5 7C           900         MOV     A,H     ; WANT TO TEST FOR 0 RESULT AFTER
 02A6 B5           901         ORA     L       ; /INCREMENTING
 02A7 CABD02       902         JZ      HIL05   ; IF SO, HL MUST HAVE CONTAINED 0FFFFH
 02AA E1           903         POP     H       ; IF NOT, RESTORE ORIGINAL HL
 02AB D5           904         PUSH    D       ; SAVE DE
 02AC 3EFF         905         MVI     A,0FFH  ; Want TO TAKE 2`S COMPLEMENT OF DE CONTENTS
 02AE AA           906         XRA     D       ;
 02AF 57           907         MOV     D,A     ;
 02B0 3EFF         908         MVI     A,0FFH  ;
 02B2 AB           909         XRA     E       ;
 02B3 5F           910         MOV     E,A     ;
 02B4 13           911         INX     D       ; 2`S COMPLEMENT ODE TO DE
 02B5 7D           912         MOV     A,L     ;
 02B6 83           913         ADD     E       ; ADD HL AND DE
 02B7 7C           914         MOV     A,H     ;
 02B8 8A           915         ADC     D       ; THIS OPERATION SETS CARRY PROPERLY
 02B9 D1           916         POP     D       ; RESTORE ORIGINAL DE CONTENTS
 02BA 78           917         MOV     A,B     ; RESTORE ORIGINAL CONTENTS OF A
 02BB C1           918         POP     B       ; RESTORE ORIGINAL CONTENTS OF BC
 02BC C9           919         RET             ; RETURN WITH CARRY SET AS REQUIRED
                   920 HIL05:
 02BD E1           921         POP     H       ; IF HL CONTAINS 0FFFFH, THEN CARRY CAN
 02BE 78           922         MOV     A,B     ; /Only BE WSET TO 1
 02BF C1           923         POP     B       ; RESTORE ORIGINAL CONTENTS OF REGISTERS
 02C0 C34303       924         JMP     SRET    ; SET CARRY AND RETURN
                   925 ;
                   926 ;
                   927 ;*****************************************************************
                   928 ;
                   929 ;
                   930 ; FUNCTION: NMOUT
                   931 ; INPUTS: A - 8 BIT INTEGER
                   932 ; OUTPUTS: NONE
                   933 ; CALLS: ECHO,PRVAL
                   934 ; DESTROYS: A,B,C,F/F'S
                   935 ; DESCRIPTION: NMOUT CONVERTS THE 8 BIT, UNSIGNED INTEGER IN THE
```

```
 LOC  OBJ         LINE        SOURCE STATEMENT

                   936 ;             A REGISTER INTO 2 ASCII CHARACTERS.  THE ASCII CHARACTERS
                   937 ;             ARE THE ONES REPRESENTING THE 8 BITS.  THESE TWO
                   938 ;             CHARACTERS ARE SENT TO THE CONSOLE AT THE CURRENT PRINT
                   939 ;             POSITION OF THE CONSOLE.
                   940 ;
                   941 NMOUT:
 02C3 E5           942         PUSH    H       ; SAVE HL - DESTROYED BY PRVAL
 02C4 F5           943         PUSH    PSW     ; SAVE ARGUMENT
 02C5 0F           944         RRC
 02C6 0F           945         RRC
 02C7 0F           946         RRC
 02C8 0F           947         RRC             ; GET UPPER 4 BITS TO LOW 4 BIT POSITIONS
 02C9 E60F         948         ANI     HCHAR   ; MASK OUT UPPER 4 BITS - WANT 1 HEX CHAR
 02CB 4F           949         MOV     C,A     ;
 02CC CDDE02       950         CALL    PRVAL   ; CONVERT LOWER 4 BITS TO ASCII
 02CF CDF401       951         CALL    ECHO    ; SEND TO TERMINAL
 02D2 F1           952         POP     PSW     ; GET BACK ARGUMENT
 02D3 E60F         953         ANI     HCHAR   ; MASK OUT UPPER 4 BITS - WANT 1 HEX CHAR
 02D5 4F           954         MOV     C,A     ;
 02D6 CDDE02       955         CALL    PRVAL   ;
 02D9 CDF401       956         CALL    ECHO    ;
 02DC E1           957         POP     H       ; RESTORE SAVED VALUE OF HL
 02DD C9           958         RET
                   959 ;
                   960 ;
                   961
;*****************************************************************************
                   962 ;
                   963 ;
                   964 ; FUNCTION;  PRVAL
                   965 ; INPUTS: A - INTEGER, RANGE 0 TO F
                   966 ; OUTPUTS: A - ASCII CHARACTER
                   967 ; CALLS: NOTHING
                   968 ; DESTROYS: B,C,H,L,F/F`S
                   969 ; DESCRIPTION:  PRVAL CONVERTS A NUMBER IN THE RANGE 0 TO F HEX TO
                   970 ;              THE CORRESPONDING ASCII CHARACTER, 0-9,A-F.  PRVAL
                   971 ;              DOES NOT CHECK THE VALIDITY OF ITS INPUT ARGUMENT.
                   972 ;
                   973 PRVAL:
 02DE 21BF03       974         LXI     H,DIGTB ; ADDRESS OF TABLE
 02E1 0600         975         MVI     B,0     ; CLEAR HIGH ORDER BITS OF BC
 02E3 09           976         DAD     B       ; ADD DIGIT VALUE TO HL ADDRESS
 02E4 4E           977         MOV     C,M     ; FETCH CHARACTER FROM MEMORY
 02E5 C9           978         RET             ;
                   979 ;
                   980
                   981 ;*****************************************************************
                   982 ;
                   983 ;
                   984 ; FUNCTION: REGDS
                   985 ; INPUTS: NONE
                   986 ; OUTPUTS: NONE
                   987 ; CALLS: ECHO,NMOUT,ERROR,CROUT
```

```
  LOC  OBJ         LINE        SOURCE STATEMENT

                    988 ; DESTROYS: A,B,C,D,E,H,L,F/F'S
                    989 ; DESCRIPTION: REGDS DISPLAYS THE CONTENTS OF THE REGISTER SAVE
                    990 ;              LOCATIONS, IN FORMATTED FORM, ON THE CONSOLE.  THE
                    991 ;              DISPLAY IS DRIVEN FROM A TABLE, RTAB, WHICH CONTAINS
                    992 ;              THE REGISTER'S PRINT SYMBOL, SAVE LOCATION ADDRESS,
                    993 ;              AND LENGTH (8 OR 16 BITS).
                    994 ;
                    995 REGDS:
  02E6 21CF03       996         LXI     H,RTAB  ; LOAD HL WITH ADDRESS OF START OF TABLE
                    997 REG05:
  02E9 4E           998         MOV     C,M     ; GET PRINT SYMBOL OF REGISTER
  02EA 79           999         MOV     A,C
  02EB B7          1000         ORA     A       ; TEST FOR 0 - END OF TABLE
  02EC C2F302      1001         JNZ     REG10   ; IF NOT END, BRANCH
  02EF CDEE01      1002         CALL    CROUT   ; ELSE, CARRIAGE RETURN/LINE FEED TO END
  02F2 C9          1003         RET             ; /DISPLAY
                   1004 REG10:
  02F3 CDF401      1005         CALL    ECHO    ; ECHO CHARACTER
  02F6 0E3D        1006         MVI     C,'='
  02F8 CDF401      1007         CALL    ECHO    ; OUTPUT EQUALS SIGN, I.E. A=
  02FB 23          1008         INX     H       ; POINT TO START OF SAVE LOCATION ADDRESS
  02FC 5E          1009         MOV     E,M     ; GET LSP OF SAVE LOCATION ADDRESS TO E
  02FD 1613        1010         MVI     D,REGS SHR 8    ; PUT MSP OF SAVE LOC ADDRESS INTO D
  02FF 23          1011         INX     H       ; POINT TO LENGTH FLAG
  0300 1A          1012         LDAX    D       ; GET CONTENTS OF SAVE ADDRESS
  0301 CDC302      1013         CALL    NMOUT   ; DISPLAY ON CONSOLE
  0304 7E          1014         MOV     A,M     ; GET LENGTH FLAG
  0305 B7          1015         ORA     A       ; SET SIGN F/F
  0306 CA0E03      1016         JZ      REG15   ; IF 0, REGISTER IS 8 BITS
  0309 1B          1017         DCX     D       ; ELSE, 16 BIT REGISTER SO MORE TO DISPLAY
  030A 1A          1018         LDAX    D       ; GET LOWER 8 BITS
  030B CDC302      1019         CALL    NMOUT   ; DISPLAY THEM
                   1020 REG15:
  030E 0E20        1021         MVI     C,' '
  0310 CDF401      1022         CALL    ECHO    ; OUTPUT BLANK CHARACTER
  0313 23          1023         INX     H       ; POINT TO START OF NEXT TABLE ENTRY
  0314 C3E902      1024         JMP     REG05   ; DO NEXT REGISTER
                   1025 ;
                   1026 ;
                   1027
;*****************************************************************************
                   1028 ;
                   1029 ;
                   1030 ; FUNCTION: RGADR
                   1031 ; INPUTS: C - CHARACTER DENOTING REGISTER
                   1032 ; OUTPUTS: BC - ADDRESS OF ENTRY IN RTAB CORRESPONDING TO REGISTER
                   1033 ; CALLS: ERROR
                   1034 ; DESTROYS: A,B,C,D,E,H,L,F/F'S
                   1035 ; DESCRIPTION: RGADR TAKES A SINGLE CHARACTER AS INPUT.  THIS CHARACTER
                   1036 ;              DENOTES A REGISTER.  RGADR SEARCHES THE TABLE RTAB
                   1037 ;              FOR A MATCH ON THE INPUT ARGUMENT.  IF ONE OCCURS,
                   1038 ;              RGADR RETURNS THE ADDRESS OF THE ADDRESS OF THE
                   1039 ;              SAVE LOCATION CORRESPONDING TO THE REGISTER.  THIS
```

```
 LOC  OBJ         LINE        SOURCE STATEMENT

                  1040 ;              ADDRESS POINTS INTO RTAB.  IF NO MATCH OCCURS, THEN
                  1041 ;              THE REGISTER IDENTIFIER IS ILLEGAL AND CONTROL IS
                  1042 ;              PASSED TO THE ERROR ROUTINE.
                  1043 ;
                  1044 RGADR:
 0317 21CF03      1045         LXI     H,RTAB  ; HL GETS ADDRESS OF TABLE START
 031A 110300      1046         LXI     D,RTABS ; DE GET SIZE OF A TABLE ENTRY
                  1047 RGA05:
 031D 7E          1048         MOV     A,M     ; GET REGISTER IDENTIFIER
 031E B7          1049         ORA     A       ; CHECK FOR TABLE END (IDENTIFIER IS 0)
 031F CA0D02      1050         JZ      ERROR   ; IF AT END OF TABLE, ARGUMENT IS ILLEGAL
 0322 B9          1051         CMP     C       ; ELSE, COMPARE TABLE ENTRY AND ARGUMENT
 0323 CA2A03      1052         JZ      RGA10   ; IF EQUAL, WE'VE FOUND WHAT WE'RE LOOKING FOR
 0326 19          1053         DAD     D       ; ELSE, INCREMENT TABLE POINTER TO NEXT ENTRY
 0327 C31D03      1054         JMP     RGA05   ; TRY AGAIN
                  1055 RGA10:
 032A 23          1056         INX     H       ; IF A MATCH, INCREMENT TABLE POINTER TO
 032B 44          1057         MOV     B,H     ; /SAVE LOCATION ADDRESS
 032C 4D          1058         MOV     C,L     ; RETURN THIS VALUE
 032D C9          1059         RET
                  1060 ;
                  1061 ;
                  1062 ;*****************************************************************
                  1063 ;
                  1064 ;
                  1065 ; FUNCTION: RSTTF
                  1066 ; INPUTS: NONE
                  1067 ; OUTPUTS: NONE
                  1068 ; CALLS: NOTHING
                  1069 ; DESTROYS: A,B,C,D,E,H,L,F/F'S
                  1070 ; DESCRIPTION: RSTTF RESTORES ALL CPU REGISTER, FLIP/FLOPS, STACK
                  1071 ;              POINTER AND PROGRAM COUNTER FROM THEIR RESPECTIVE
                  1072 ;              SAVE LOCATIONS IN MEMORY.  THE ROUTINE THEN TRANSFERS
                  1073 ;              CONTROL TO THE LOCATION SPECIFIED BY THE PROGRAM
                  1074 ;              COUNTER (I.E. THE RESTORED VALUE).  THE ROUTINE
                  1075 ;              EXITS WITH THE INTERRUPTS ENABLED.
                  1076 ;
                  1077 RSTTF:
 032E F3          1078         DI              ; DISABLE INTERRUPTS WHILE RESTORING THINGS
 032F 21ED13      1079         LXI     H,MSTAK ; SET MONITOR STACK POINTER TO START OF STACK
 0332 F9          1080         SPHL            ;
 0333 D1          1081         POP     D       ; START ALSO END OF REGISTER SAVE AREA
 0334 C1          1082         POP     B       ;
 0335 F1          1083         POP     PSW     ;
 0336 2AF713      1084         LHLD    SSAVE   ; RESTORE USER STACK POINTER
 0339 F9          1085         SPHL            ;
 033A 2AF513      1086         LHLD    PSAVE   ;
 033D E5          1087         PUSH    H       ; PUT USER RETURN ADDRESS ON USER STACK
 033E 2AF313      1088         LHLD    LSAVE   ; RESTORE HL REGISTERS
 0341 FB          1089         EI              ; ENABLE INTERRUPTS NOW
 0342 C9          1090         RET             ; JUMP TO RESTORED PC LOCATION
                  1091 ;
```

```
 LOC  OBJ         LINE        SOURCE STATEMENT

                  1092 ;
                  1093 ;*****************************************************************
                  1094 ;
                  1095 ;
                  1096 ; FUNCTION: SRET
                  1097 ; INPUTS: NONE
                  1098 ; OUTPUTS: CARRY = 1
                  1099 ; CALLS: NOTHING
                  1100 ; DESTROYS: CARRY
                  1101 ; DESCRIPTION: SRET IS JUMPED TO BY ROUTINES WISHING TO RETURN SUCCESS.
                  1102 ;              SRET SETS THE CARRY TRUE AND THEN RETURNS TO THE
                  1103 ;              CALLER OF THE ROUTINE INVOKING SRET.
                  1104 ;
                  1105 SRET:
 0343 37          1106         STC             ; SET CARRY TRUE
 0344 C9          1107         RET             ; RETURN APPROPRIATELY
                  1108 ;
                  1109 ;
                  1110 ;*****************************************************************
                  1111 ;
                  1112 ;
                  1113 ; FUNCTION: STHF0
                  1114 ; INPUTS: DE - 16 BIT ADDRESS OF BYTE TO BE STORED INTO
                  1115 ; OUTPUTS: NONE
                  1116 ; CALLS: NOTHING
                  1117 ; DESTROYS: A,B,C,H,L,F/F'S
                  1118 ; DESCRIPTION: STHF0 CHECKS THE HALF BYTE FLAG IN TEMP TO SEE IF
                  1119 ;              IT IS SET TO LOWER.  IF SO, STHF0 STORES A 0 TO
                  1120 ;              PAD OUT THE LOWER HALF OF THE ADDRESSED BYTE;
                  1121 ;              OTHERWISE, THE ROUTINE TAKES NO ACTION.
                  1122 ;
                  1123 STHF0:
 0345 3AF913      1124         LDA     TEMP    ; GET HALF BYTE FLAG
 0348 B7          1125         ORA     A       ; SET F/F'S
 0349 C0          1126         RNZ             ; IF SET TO UPPER, DON'T DO ANYTHING
 034A 0E00        1127         MVI     C,0     ; ELSE, WANT TO STORE THE VALUE 0
 034C CD5003      1128         CALL    STHLF   ; DO IT
 034F C9          1129         RET
                  1130 ;
                  1131 ;
                  1132 ;*****************************************************************
                  1133 ;
                  1134 ;
                  1135 ; FUNCTION: STHLF
                  1136 ; INPUTS: C - 4 BIT VALUE TO BE STORED IN HALF BYTE
                  1137 ;         DE - 16 BIT ADDRESS OF BYTE TO BE STORED INTO
                  1138 ; OUTPUTS: NONE
                  1139 ; CALLS: NOTHING
                  1140 ; DESTROYS: A,B,C,H,L,F/F'S
                  1141 ; DESCRIPTION: STHLF TAKES THE 4 BIT VALUE IN C AND STORES IT IN
                  1142 ;              HALF OF THE BYTE ADDRESSED BY REGISTERS DE.  THE
                  1143 ;              HALF BYTE USED (EITHER UPPER OR LOWER) IS DENOTED
```

```
 LOC  OBJ         LINE        SOURCE STATEMENT

                  1144 ;              BY THE VALUE OF THE FLAG IN TEMP.  STHLF ASSUMES
                  1145 ;              THAT THIS FLAG HAS BEEN PREVIOUSLY SET
                  1146 ;              (NOMINALLY BY ICMD).
                  1147 ;
                  1148 STHLF:
 0350 D5          1149         PUSH    D
 0351 E1          1150         POP     H       ; MOVE ADDRESS OF BYTE INTO HL
 0352 79          1151         MOV     A,C     ; GET VALUE
 0353 E60F        1152         ANI     0FH     ; FORCE TO 4 BIT LENGTH
 0355 4F          1153         MOV     C,A     ; PUT VALUE BACK
 0356 3AF913      1154         LDA     TEMP    ; GET HALF BYTE FLAG
 0359 B7          1155         ORA     A       ; CHECK FOR LOWER HALF
 035A C26303      1156         JNZ     STH05   ; BRANCH IF NOT
 035D 7E          1157         MOV     A,M     ; ELSE, GET BYTE
 035E E6F0        1158         ANI     0F0H    ; CLEAR LOWER 4 BITS
 0360 B1          1159         ORA     C       ; OR IN VALUE
 0361 77          1160         MOV     M,A     ; PUT BYTE BACK
 0362 C9          1161         RET
                  1162 STH05:
 0363 7E          1163         MOV     A,M     ; IF UPPER HALF, GET BYTE
 0364 E60F        1164         ANI     0FH     ; CLEAR UPPER 4 BITS
 0366 47          1165         MOV     B,A     ; SAVE BYTE IN B
 0367 79          1166         MOV     A,C     ; GET VALUE
 0368 0F          1167         RRC
 0369 0F          1168         RRC
 036A 0F          1169         RRC
 036B 0F          1170         RRC             ; ALIGN TO UPPER 4 BITS
 036C B0          1171         ORA     B       ; OR IN ORIGINAL LOWER 4 BITS
 036D 77          1172         MOV     M,A     ; PUT NEW CONFIGURATION BACK
 036E C9          1173         RET
                  1174 ;
                  1175 ;
                  1176 ;*****************************************************************
                  1177 ;
                  1178 ;
                  1179 ; FUNCTION: VALDG
                  1180 ; INPUTS: C - ASCII CHARACTER
                  1181 ; OUTPUTS: CARRY - 1 IF CHARACTER REPRESENTS VALID HEX DIGIT
                  1182 ;                - 0 OTHERWISE
                  1183 ; CALLS: NOTHING
                  1184 ; DESTROYS: A,F/F'S
                  1185 ; DESCRIPTION: VALDG RETURNS SUCCESS IF ITS INPUT ARGUMENT IS
                  1186 ;              AN ASCII CHARACTER REPRESENTING A VALID HEX DIGIT
                  1187 ;              (0-9,A-F), AND FAILURE OTHERWISE.
                  1188 ;
                  1189 VALDG:
 036F 79          1190         MOV     A,C
 0370 FE30        1191         CPI     '0'     ; TEST CHARACTER AGAINST '0'
 0372 FA1802      1192         JM      FRET    ; IF ASCII CODE LESS, CANNOT BE VALID DIGIT
 0375 FE39        1193         CPI     '9'     ; ELSE, SEE IF IN RANGE '0'-'9'
 0377 FA4303      1194         JM      SRET    ; CODE BETWEEN '0' AND '9'
 037A CA4303      1195         JZ      SRET    ; CODE EQUAL '9'
```

```
 LOC  OBJ         LINE        SOURCE STATEMENT

 037D FE41        1196         CPI     'A'     ; NOT A DIGIT - TRY FOR A LETTER
 037F FA1802      1197         JM      FRET    ; NO - CODE BETWEEN '9' AND 'A'
 0382 FE47        1198         CPI     'G'
 0384 F21802      1199         JP      FRET    ; NO - CODE GREATER THAN 'F'
 0387 C34303      1200         JMP     SRET    ; OKAY - CODE IS 'A' TO 'F', INCLUSIVE
                  1201 ;
                  1202 ;
                  1203 ;*****************************************************************
                  1204 ;
                  1205 ;
                  1206 ; FUNCTION: VALDL
                  1207 ; INPUTS: C - CHARACTER
                  1208 ; OUTPUTS: CARRY - 1 IF INPUT ARGUMENT VALID DELIMTER
                  1209 ;                - 0 OTHERWISE
                  1210 ; CALLS: NOTHING
                  1211 ; DESTROYS: A,F/F'S
                  1212 ; DESCRIPTION: VALDL RETURNS SUCCESS IF ITS INPUT ARGUMENT IS A VALID
                  1213 ;              DELIMITER CHARACTER (SPACE, COMMA, CARRIAGE RETURN) AND
                  1214 ;              FAILURE OTHERWISE.
                  1215 ;
                  1216 VALDL:
 038A 79          1217         MOV     A,C
 038B FE2C        1218         CPI     ','     ; CHECK FOR COMMA
 038D CA4303      1219         JZ      SRET
 0390 FE0D        1220         CPI     CR      ; CHECK FOR CARRIAGE RETURN
 0392 CA4303      1221         JZ      SRET
 0395 FE20        1222         CPI     ' '     ; CHECK FOR SPACE
 0397 CA4303      1223         JZ      SRET
 039A C31802      1224         JMP     FRET    ; ERROR IF NONE OF THE ABOVE
                  1225 ;
                  1226 ;
                  1227 ;*****************************************************************
                  1228 ;
                  1229 ;                       MONITOR TABLES
                  1230 ;
                  1231 ;*****************************************************************
                  1232 ;
                  1233 ;
                  1234 SGNON:                  ; SIGNON MESSAGE
 039D 0D          1235         DB      CR,LF,'MCS-80 KIT',CR,LF
 039E 0A
 039F 4D43532D
 03A3 3830204B
 03A7 4954
 03A9 0D
 03AA 0A
 000E             1236 LSGNON  EQU     $-SGNON ; LENGTH OF SIGNON MESSAGE
                  1237 ;
                  1238 CADR:                   ; TABLE OF ADDRESSES OF COMMAND ROUTINES
 03AB 0000        1239         DW      0       ; DUMMY
 03AD 4101        1240         DW      XCMD
 03AF 1D01        1241         DW      SCMD
```

```
 LOC  OBJ         LINE        SOURCE STATEMENT

 03B1 FD00        1242         DW      MCMD
 03B3 B300        1243         DW      ICMD
 03B5 9500        1244         DW      GCMD
 03B7 5E00        1245         DW      DCMD
                  1246 ;
                  1247 CTAB:                   ; TABLE OF VALID COMMAND CHARACTERS
 03B9 44          1248         DB      'D'
 03BA 47          1249         DB      'G'
 03BB 49          1250         DB      'I'
 03BC 4D          1251         DB      'M'
 03BD 53          1252         DB      'S'
 03BE 58          1253         DB      'X'
 0006             1254 NCMDS   EQU     $-CTAB  ; NUMBER OF VALID COMMANDS
                  1255 ;
                  1256 DIGTB:
 03BF 30          1257         DB      '0'
 03C0 31          1258         DB      '1'
 03C1 32          1259         DB      '2'
 03C2 33          1260         DB      '3'
 03C3 34          1261         DB      '4'
 03C4 35          1262         DB      '5'
 03C5 36          1263         DB      '6'
 03C6 37          1264         DB      '7'
 03C7 38          1265         DB      '8'
 03C8 39          1266         DB      '9'
 03C9 41          1267         DB      'A'
 03CA 42          1268         DB      'B'
 03CB 43          1269         DB      'C'
 03CC 44          1270         DB      'D'
 03CD 45          1271         DB      'E'
 03CE 46          1272         DB      'F'
                  1273 ;
                  1274 RTAB:                   ; TABLE OF REGISTER INFORMATION
 03CF 41          1275         DB      'A'     ; REGISTER IDENTIFIER
 03D0 F2          1276         DB      ASAVE AND 0FFH  ; ADDRESS OF REGISTER SAVE LOCATION
 03D1 00          1277         DB      0       ; LENGTH FLAG - 0=8 BITS, 1=16 BITS
 0003             1278 RTABS   EQU     $-RTAB  ; SIZE OF AN ENTRY IN THIS TABLE
 03D2 42          1279         DB      'B'
 03D3 F0          1280         DB      BSAVE AND 0FFH
 03D4 00          1281         DB      0
 03D5 43          1282         DB      'C'
 03D6 EF          1283         DB      CSAVE AND 0FFH
 03D7 00          1284         DB      0
 03D8 44          1285         DB      'D'
 03D9 EE          1286         DB      DSAVE AND 0FFH
 03DA 00          1287         DB      0
 03DB 45          1288         DB      'E'
 03DC ED          1289         DB      ESAVE AND 0FFH
 03DD 00          1290         DB      0
 03DE 46          1291         DB      'F'
 03DF F1          1292         DB      FSAVE AND 0FFH
 03E0 00          1293         DB      0
```

```
 LOC  OBJ         LINE        SOURCE STATEMENT

 03E1 48          1294         DB      'H'
 03E2 F4          1295         DB      HSAVE AND 0FFH
 03E3 00          1296         DB      0
 03E4 4C          1297         DB      'L'
 03E5 F3          1298         DB      LSAVE AND 0FFH
 03E6 00          1299         DB      0
 03E7 4D          1300         DB      'M'
 03E8 F4          1301         DB      HSAVE AND 0FFH
 03E9 01          1302         DB      1
 03EA 50          1303         DB      'P'
 03EB F6          1304         DB      PSAVE+1 AND 0FFH
 03EC 01          1305         DB      1
 03ED 53          1306         DB      'S'
 03EE F8          1307         DB      SSAVE+1 AND 0FFH
 03EF 01          1308         DB      1
 03F0 00          1309         DB      0       ; END OF TABLE MARKERS
 03F1 00          1310         DB      0
                  1311 ;
 03FA             1312         ORG     BRTAB
                  1313 ;
 03FA C3E301      1314         JMP     CO      ; BRANCH TABLE FOR USER ACCESIBLE ROUTINES
 03FD C3D001      1315         JMP     CI      ;
                  1316 ;
                  1317 ;
                  1318 ;*****************************************************************
                  1319 ;
                  1320 ;
 1300             1321         ORG     DATA
 13ED             1322         ORG     REGS    ; ORG TO REGISTER SAVE - STACK GOES IN HERE
                  1323 ;
 13ED             1324 MSTAK   EQU     $       ; START OF MONITOR STACK
 13ED 00          1325 ESAVE:  DB      0       ; E REGISTER SAVE LOCATION
 13EE 00          1326 DSAVE:  DB      0       ; D REGISTER SAVE LOCATION
 13EF 00          1327 CSAVE:  DB      0       ; C REGISTER SAVE LOCATION
 13F0 00          1328 BSAVE:  DB      0       ; B REGISTER SAVE LOCATION
 13F1 00          1329 FSAVE:  DB      0       ; FLAGS SAVE LOCATION
 13F2 00          1330 ASAVE:  DB      0       ; A REGISTER SAVE LOCATION
 13F3 00          1331 LSAVE:  DB      0       ; L REGISTER SAVE LOCATION
 13F4 00          1332 HSAVE:  DB      0       ; H REGISTER SAVE LOCATION
 13F5 0000        1333 PSAVE:  DW      0       ; PGM COUNTER SAVE LOCATION
 13F7 0000        1334 SSAVE:  DW      0       ; USER STACK POINTER SAVE LOCATION
 13F9 00          1335 TEMP:   DB      0       ; TEMPORARY MONITOR CELL
                  1336 ;
 13FD             1337         ORG     BRLOC   ; ORG TO USER BRANCH LOCATION
                  1338 ;
 13FD             1339 USRBR:  DS      3       ; BRANCH GOES IN HERE
                  1340 ;
                  1341 ;
                  1342         END


PUBLIC SYMBOLS
```

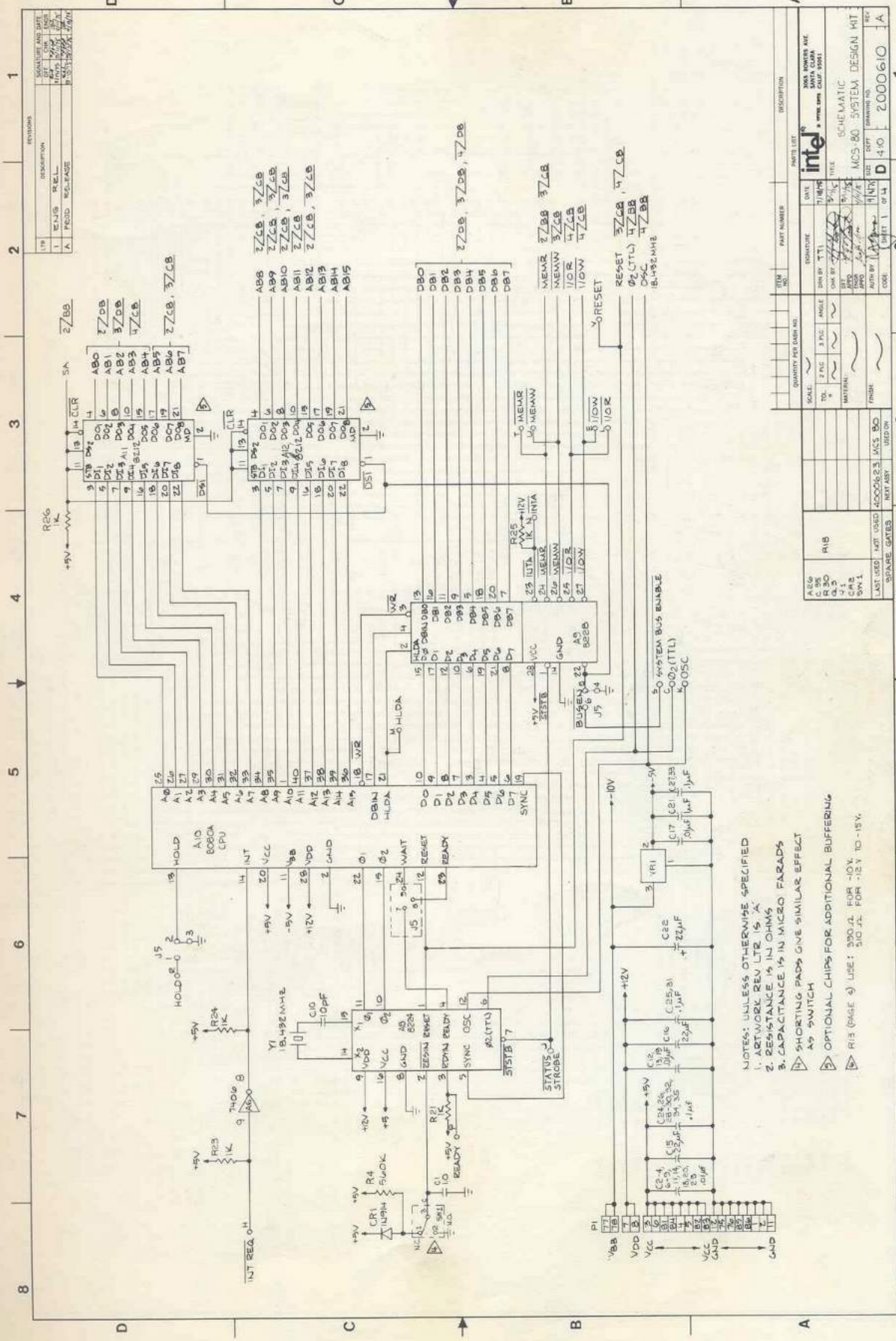
EXTERNAL SYMBOLS

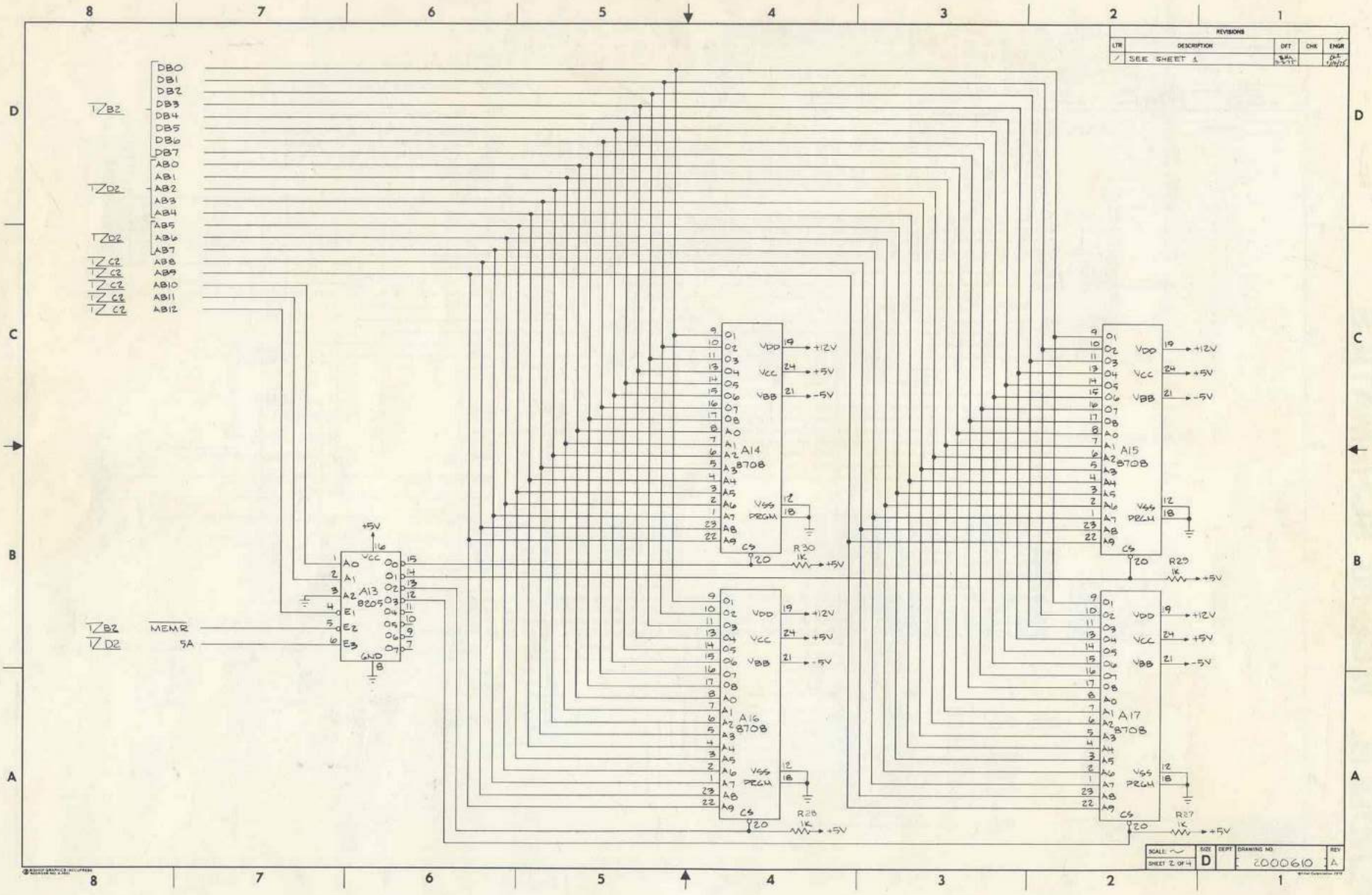
USER SYMBOLS

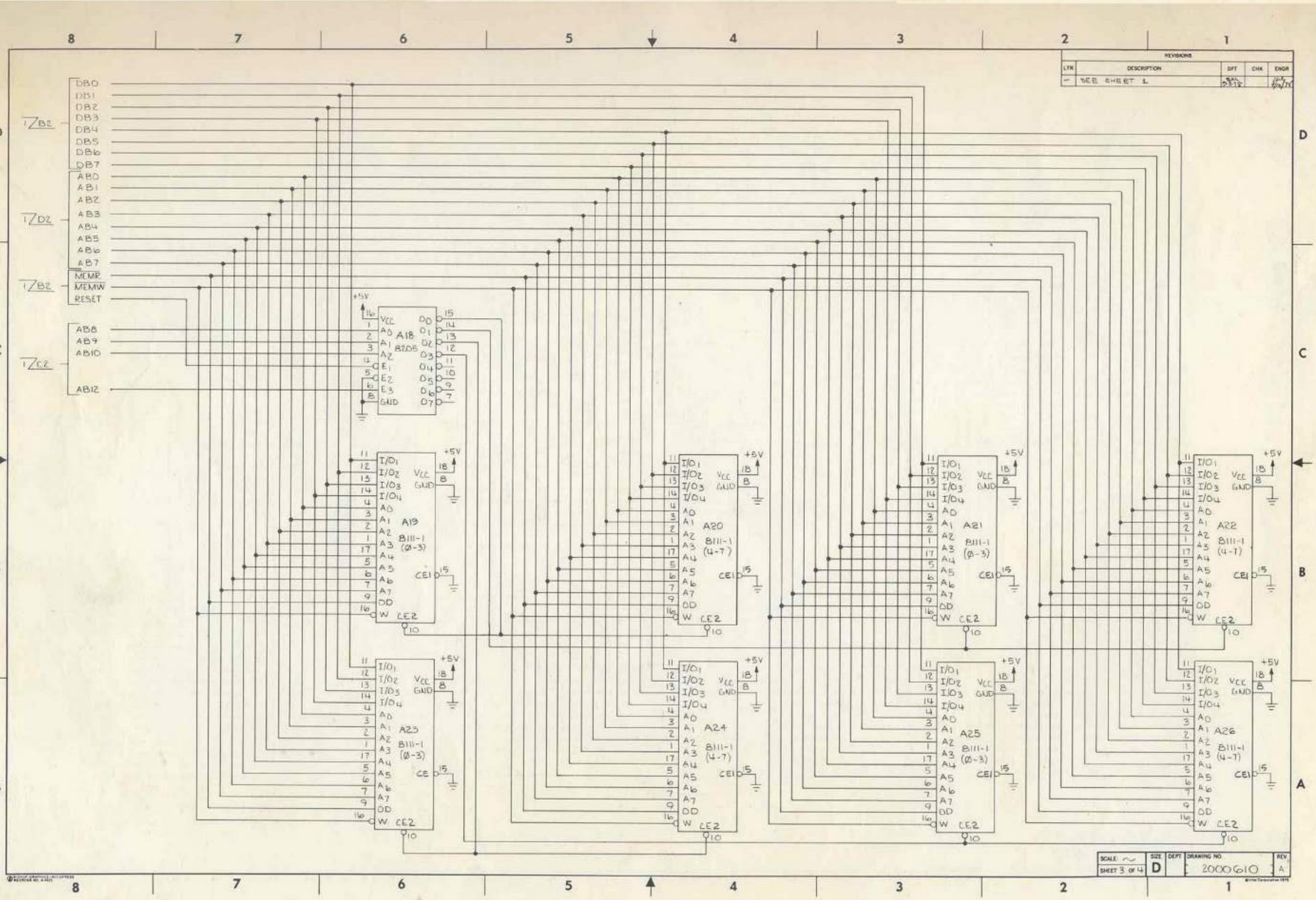
ASAVE	A 13F2	BRCHR	A 001B	BREAK	A 01BD	BRLOC	A 13FD	BRTAB	A 03FA	BSAVE	A 13F0	CADR	A 03AB
CI	A 01D0	CMD	A 0027	CNCTL	A 00FB	CNIN	A 00FA	CNOUT	A 00FA	CNVBN	A 01DA	CO	A 01E3
CONST	A 00FB	CR	A 000D	CROUT	A 01EE	CSAVE	A 13EF	CTAB	A 03B9	DATA	A 1300	DCM05	A 0065
DCM10	A 0070	DCM12	A 0085	DCM15	A 008B	DCMD	A 005E	DIGTB	A 03BF	DSAVE	A 13EE	ECH05	A 01FD
ECH10	A 020B	ECHO	A 01F4	ERROR	A 020D	ESAVE	A 13ED	ESC	A 001B	EXIT	A 0212	FALSE	+ 0001
FRET	A 0218	FSAVE	A 13F1	GCM05	A 00AA	GCM10	A 00B0	GCMD	A 0095	GETCH	A 021B	GETCM	A 002B
GETHX	A 0222	GETNM	A 0257	GHX05	A 0228	GHX10	A 0241	GNM05	A 025E	GNM10	A 0273	GNM15	A 0281
GNM20	A 0286	GNM25	A 0291	GNM30	A 0295	GO	A 0008	GTC03	A 003B	GTC05	A 0048	GTC10	A 0054
HCHAR	A 000F	HIL05	A 02BD	HILO	A 029C	HSAVE	A 13F4	ICM05	A 00BE	ICM10	A 00E6	ICM20	A 00EE
ICM25	A 00F4	ICMD	A 00B3	INVRT	A 00FF	LF	A 000A	LOWER	A 0000	LSAVE	A 13F3	LSGNON	A 000E
MCM05	A 0105	MCMD	A 00FD	MODE	A 00CF	MSGL	A 0022	MSTAK	A 13ED	NCMDS	A 0006	NEWLN	A 000F
NMOUT	A 02C3	PRTY0	A 007F	PRVAL	A 02DE	PSAVE	A 13F5	RBR	A 0002	REG05	A 02E9	REG10	A 02F3
REG15	A 030E	REGDS	A 02E6	REGS	A 13ED	RGA05	A 031D	RGA10	A 032A	RGADR	A 0317	RSTTF	A 032E
RSTU	A 0038	RTAB	A 03CF	RTABS	A 0003	SCM05	A 0122	SCM10	A 012D	SCM15	A 013D	SCMD	A 011D
SGNON	A 039D	SOMSG	A 001D	SRET	A 0343	SSAVE	A 13F7	STH05	A 0363	STHF0	A 0345	STHLF	A 0350
TEMP	A 13F9	TERM	A 001B	TRDY	A 0001	TRUE	+ 0000	UPPER	A 00FF	USRBR	A 13FD	VALDG	A 036F
VALDL	A 038A	XCM05	A 0154	XCM10	A 0163	XCM15	A 0170	XCM18	A 017B	XCM20	A 0194	XCM25	A 01AB
XCM27	A 01AC	XCM30	A 01B4	XCMD	A 0141								

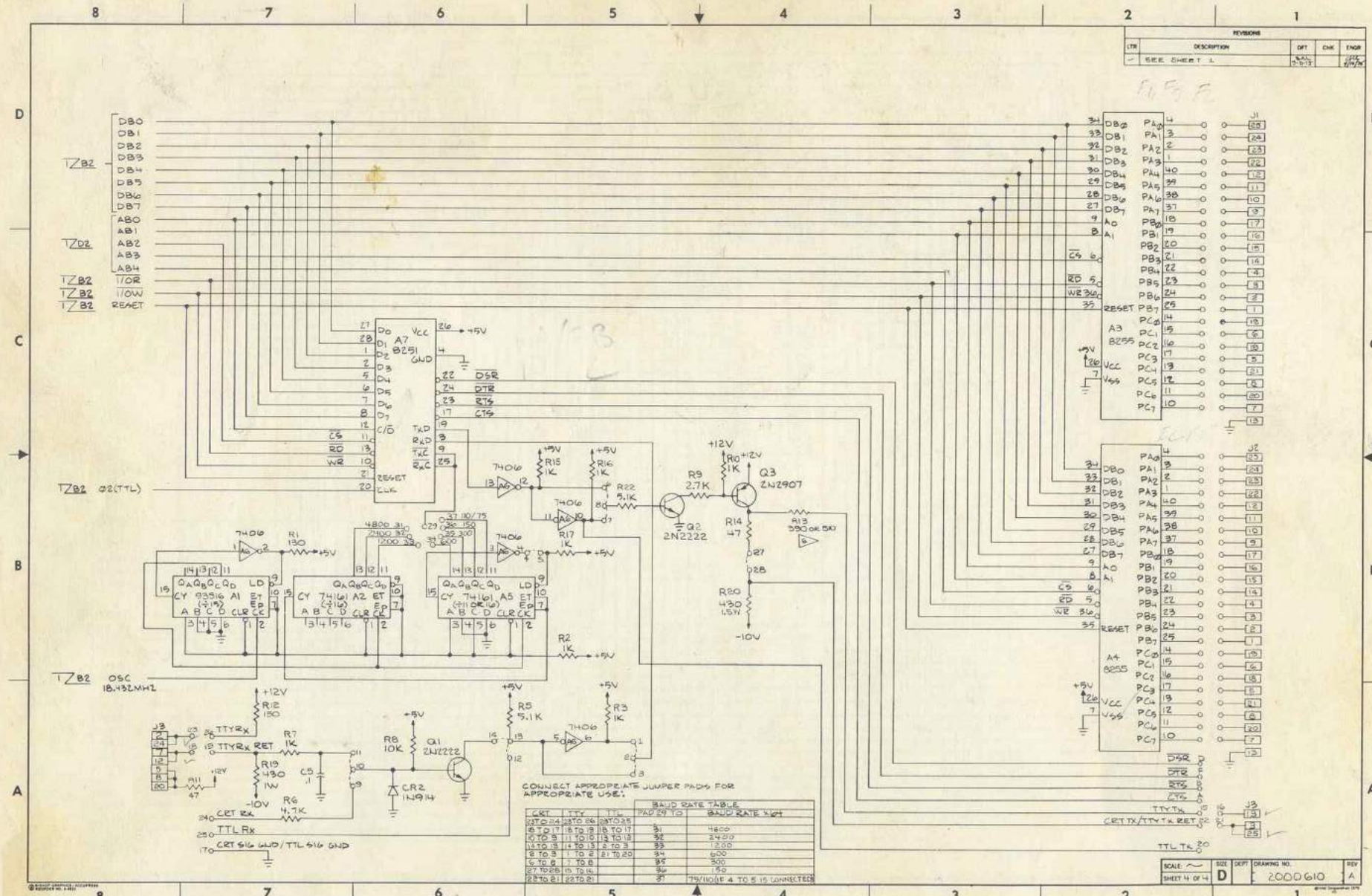
ASSEMBLY COMPLETE, NO ERRORS

APPENDIX B. MCS-80™ SCHEMATICS

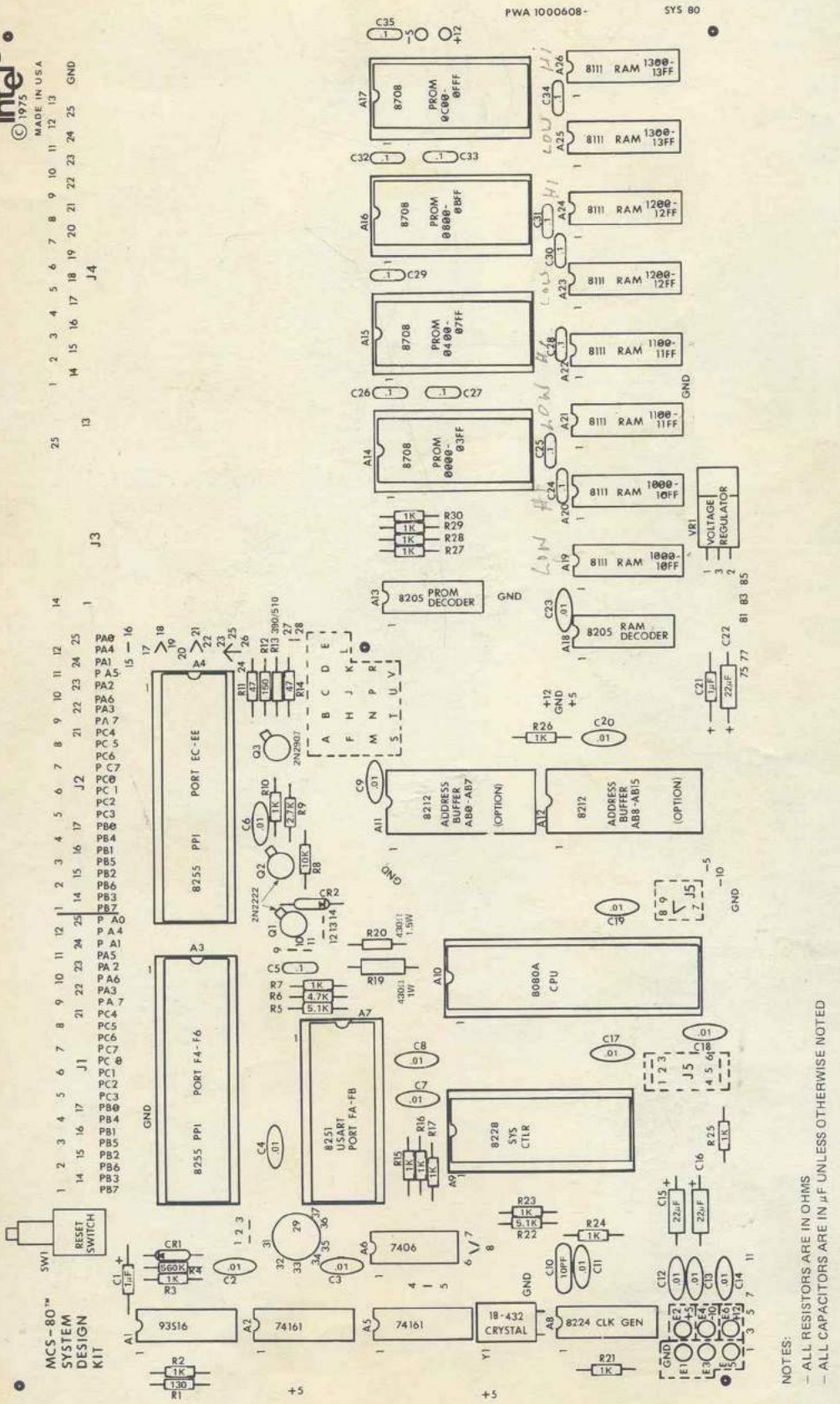



REVISIONS
LTR DESCRIPTION DFT CHK ENGR
SEE SHEET 1
DB0 DB1 DB2 DB3 DB4 DB5 DB6 DB7
AB0 AB1 AB2 AB3 AB4 AB5 AB6 AB7 AB8 AB9 AB10 AB11 AB12
MEMR
SA
A13 8205
A14 8708
A15 8708
A16 8708
A17 8708
VDD +12V
VCC +5V
VBB -5V
VSS
PGM
CS
R30 1K
R29 1K
R28 1K
R27 1K
SCALE:
SHEET 2 OF 4
SIZE D
DRAWING NO. 2000610
REV A


REVISIONS
LTR DESCRIPTION DFT CHK ENGR
– SEE SHEET 1.
DB0
DB1
DB2
DB3
DB4
DB5
DB6
DB7
ADB0
ADB1
ADB2
ADB3
ADB4
ADB5
ADB6
ADB7
MEMR
MEMW
RESET
ADB8
ADB9
ADB10
ADB12
1/B2
1/D2
1/B2
1/C2
A18
A19
A20
A21
A22
A23
A24
A25
A26
8208
B111-1 (0-3)
B111-1 (4-7)
+5V
SCALE
SHEET 3 OF 4
SIZE D
DEPT
DRAWING NO. 2000610
REV A


REVISIONS
LTR
DESCRIPTION
DFT
CHK
ENGR
SEE SHEET 1
A7 8251
A3 8255
A4 8255
Q1 2N2222
Q2 2N2222
Q3 2N2907
CR2 1N914
7406
93516 A1
74161 A2
74161 A5
OSC 18.432MHZ
TTYRx
TTYRx RET
CRT Rx
TTL Rx
CRT SIG GND/TTL SIG GND
TTYTx
CRT TX/TTY TX RET
TTL TX
CONNECT APPROPRIATE JUMPER PADS FOR APPROPRIATE USE:
BAUD RATE TABLE
CRT TTY TTL
PAD 29 TO
BAUD RATE X64
23 TO 24 23 TO 26 23 TO 25
18 TO 17 18 TO 19 18 TO 17
10 TO 9 11 TO 10 13 TO 12
14 TO 13 14 TO 13 2 TO 3
2 TO 3 1 TO 2 21 TO 20
6 TO 8 7 TO 8
27 TO 28 15 TO 16
22 TO 21 22 TO 21
31 4800
32 2400
33 1200
34 600
35 300
36 150
37 75/110 (IF 4 TO 5 IS CONNECTED)
SCALE
SHEET 4 OF 4
SIZE D
DEPT
DRAWING NO. 2000610
REV A

APPENDIX C. BOARD LAYOUT WITH COMPONENT VALUES



INTEL CORPORATION, 3065 Bowers Avenue, Santa Clara, California 95051 (408) 246-7501