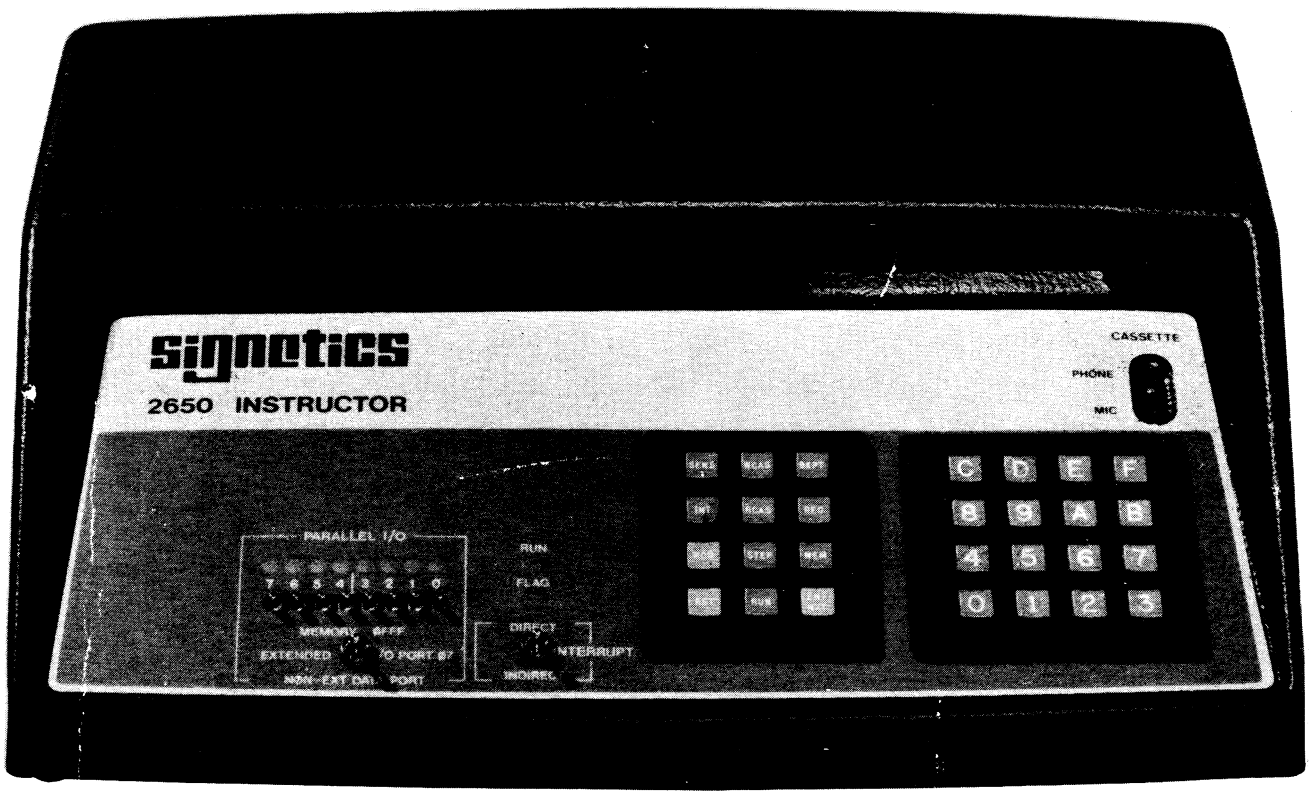
$5.00

SIGNETICS INSTRUCTOR 50 USER GUIDE

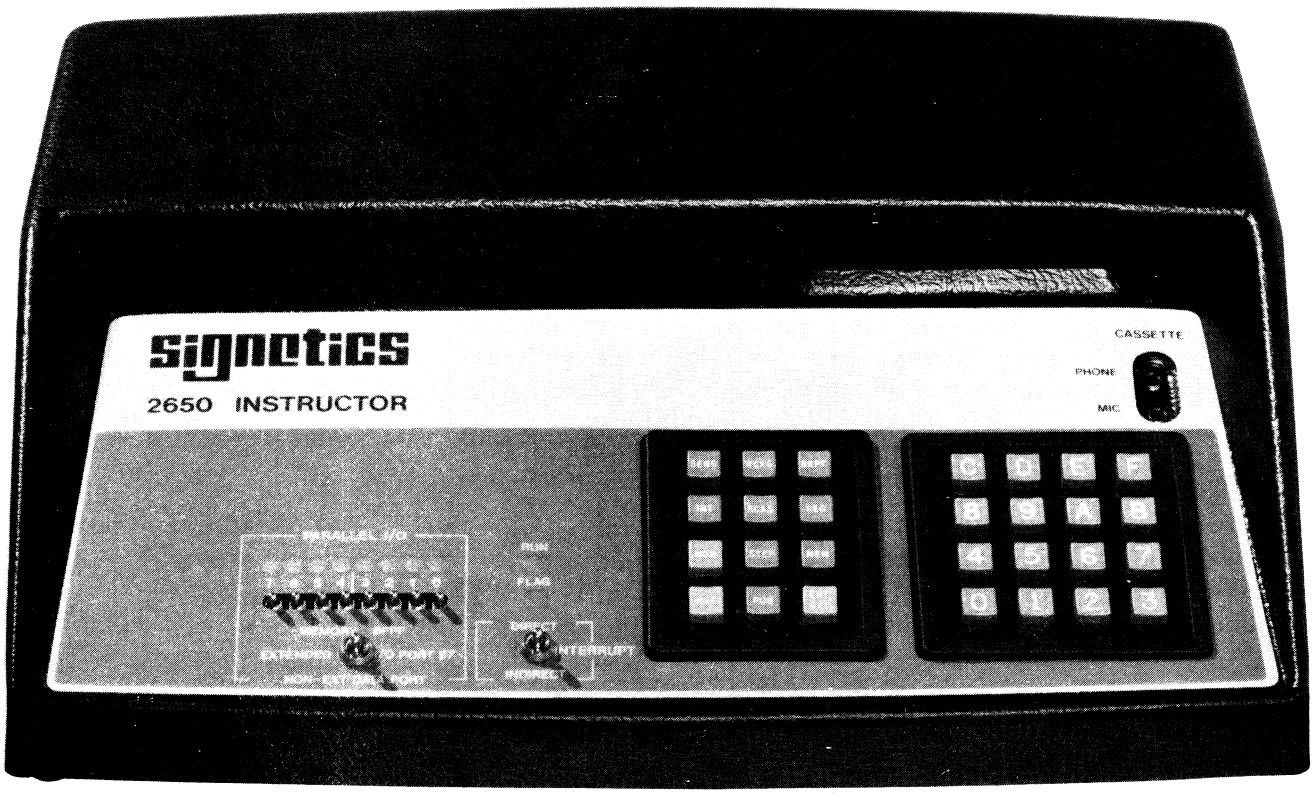


signetics
a subsidiary of U.S. Philips Corporation

Signetics Corporation
811 East Arques Avenue

$5.00

SIGNETICS INSTRUCTOR 50 USER GUIDE



signetics

a subsidiary of **U.S. Philips Corporation**

Signetics Corporation
811 East Arques Avenue
Sunnyvale, California 94086
Telephone 408/739-7700

PREFACE

This manual provides tutorial and reference information on the Signetics INSTRUCTOR 50--a complete, fully assembled and low cost microcomputer system. The INSTRUCTOR 50's computing power is enhanced by the Signetics 2650 Microprocessor which is described in detail in the 2650 manual accompanying this document.

INSTRUCTOR 50 is designed to assist you in learning programming and in writing, debugging, and testing the programs you develop. There is enough information here to get you started, whether or not you have ever written a program before. The only prerequisite is a familiarity with the 2650 microprocessor. Readers who are not familiar with the 2650's hardware structure and instruction set should read the 2650 Microprocessor Manual prior to using the INSTRUCTOR 50.

CONTENTS

CONTENTS (cont.)

CONTENTS (cont.)

LIST OF ILLUSTRATIONS

1. INTRODUCTION

Welcome aboard the INSTRUCTOR 50--a unique and powerful training tool designed to introduce you to the world of microcomputers in the shortest possible time.

INSTRUCTOR 50 is for computer hobbyists, students, engineers or anyone who wants to learn how to use a microcomputer the easy way, without having to face the drudgery of a long and tedious training program.

INSTRUCTOR 50 is a stand-alone microcomputer based on the Signetics 2650 microprocessor. It includes everything that you need to write, run, and debug machine-language programs. A 12-key Function Control Keyboard and a 16-key Hexadecimal Keyboard are used to enter data and perform various system functions associated with the INSTRUCTOR 50. The INSTRUCTOR 50 User System Executive (USE) monitor program guides you in the use of the system by displaying prompting messages and responses on an eight-digit LED display. All facilities required for program development are built into INSTRUCTOR 50 -- you don't need anything else to start.

Before getting into the details of what makes the INSTRUCTOR 50 tick, let's first take a short shakedown cruise and write a few simple programs.

Power On and Initial Display

To apply power to the INSTRUCTOR 50, connect the power cord into the rear panel receptacle, and insert the power pack into any standard 115 VAC domestic wall socket. The INSTRUCTOR 50 does not have a power ON/OFF switch. The initial display is the message HELLO, indicating that the INSTRUCTOR 50 is in the monitor mode and ready for use. If the HELLO message does not appear, depress the MON key to initialize the INSTRUCTOR 50. Unplug the power pack to turn the INSTRUCTOR 50 off.

Operating Modes

The INSTRUCTOR 50 has two basic modes of operation, the MONITOR mode and the EXECUTION mode. The MONITOR mode is entered automatically on power up or by depressing the MON key on the function control keyboard. The monitor responds by displaying HELLO. While in the MONITOR mode, you may:

- Enter and alter a program.
- Read in a previously saved program from audio cassette tape.
- Display and alter the contents of the microcomputer's general-purpose working registers and/or Program Status Word (PSW).
- Examine and alter the contents of memory locations.

- Examine and alter the contents of the Program Counter.
- Specify and examine a program breakpoint.
- Step through a program one instruction at a time.
- Save a program on cassette tape.

The EXECUTION mode is entered by depressing the RUN key, the STEP key, or the RESET (RST) key on the function control keyboard. Depressing the RUN key terminates the MONITOR mode and causes program execution to begin at the address specified in the Program Counter. Depressing the STEP key causes the INSTRUCTOR 50 to execute a single instruction and return to the MONITOR mode. When the RST key is depressed, current INSTRUCTOR 50 activity is terminated, and the processor begins program execution at address H'0000'.

Keying in and Entering Values

The INSTRUCTOR 50 uses the hexadecimal number system with a base of 16 for entering values. The term "hexadecimal", or hex for short, refers to a shorthand method of expressing a group of four consecutive binary bits by a single digit. Valid digits range from 0 through F, where F represents the highest decimal value (15). See Table 1.1.

Since the INSTRUCTOR 50 uses 8-bit bytes, two hexadecimal digits can be used to specify a byte. The smallest hexadecimal number is 00_{16} (00000000_2) and the largest is FF_{16} (11111111_2). The INSTRUCTOR 50 still reads only binary numbers; hexadecimal is the user's shorthand, not the microcomputer's.

Decimal	Hexadecimal	Binary
0	0	0000
1	1	0001
2	2	0010
3	3	0011
4	4	0100
5	5	0101
6	6	0110
7	7	0111
8	8	1000
9	9	1001
10	A	1010
11	B	1011
12	C	1100
13	D	1101
14	E	1110
15	F	1111

Table 1.1: Relationship among decimal, hexadecimal, and binary systems.

To understand hex notation, take a decimal number like 107_{10}. In binary notation, this becomes 1101011_2. Breaking this number into two 4-bit nibbles (half-bytes), you get 0110_2 and 1011_2. The first and most-significant nibble is equal to 6_{10}, while the second and least-significant nibble is equal to 11_{10}. Thus, in hexadecimal notation, 107_{10} becomes $6B_{16}$ or H'6B'. To convert from decimal to hexadecimal, or vice versa, you must first convert the number into binary and then into hexadecimal as previously illustrated.

Address and data parameters are entered into the INSTRUCTOR 50 via the hexadecimal keyboard using the hex notation described earlier. When entering an address, you may enter as many as four hex digits starting with the most significant digit of the address. Leading zeroes need not be entered; if less than four digits are entered, the leading digits are automatically zeroed. Data values consist of one or two hex digits, with the most-significant digit entered first. If only one digit is entered, the most-significant digit is automatically zeroed.

Correcting Entry Errors

The numbers keyed in appear in the address/data display field and can be edited prior to depression of a function key by simply keying in the correct characters. The display shifts to the left each time a new character is entered, and characters shifted out of the field are disregarded. Only the last digits entered are retained, so that an error in entry can be corrected by entering the correct data.*

For example, if you were entering an address and you depressed 121 instead of the correct value of 120, the display would read:

```
.Ad. =    121
```

To recover from this error, simply key in the correct value by depressing the following hex keys:

[0] [1] [2] [0]

The correct value would then be displayed as indicated below:

```
.Ad. =   0120
```

The Prompt Light

A dot or period in the left-most position of the display (e.g., .Ad. =) is a prompt signal. It indicates that the INSTRUCTOR 50 is ready to accept a data or address value.

---

*Data values entered during operation in the FAST PATCH command mode cannot be corrected in this manner. See description of the FAST PATCH command in Section 4 .

Entering and Executing a Simple Program

To demonstrate the use of the INSTRUCTOR 50, let's write a simple program, enter it, and execute it. Prior to writing the program, we must decide what task or operation we want the program to perform.

Let's say we want to "show the operation of an 8-bit binary counter on the INSTRUCTOR 50's output port indicator LEDs". The flowchart for performing this task is shown in Figure 1.1.

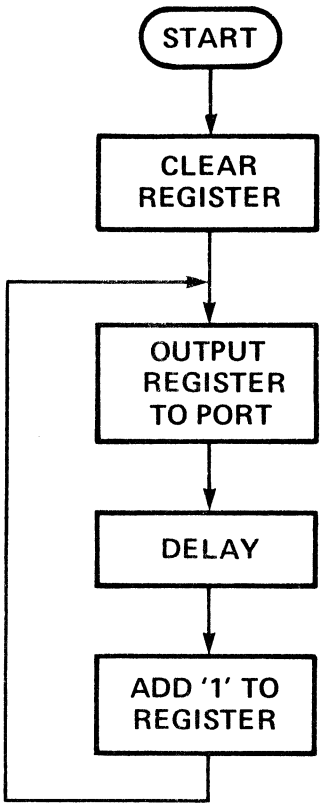


Figure 1.1: Flowchart for Binary Counter Program

The DELAY block shown in the flowchart provides a time interval between new values of the binary count in order to observe the counting action on the port indicators. This can be implemented in several ways, depending on the delay required.* We will use a double-loop technique, with the outer loop counting the number of excursions through the inner loop.

The next step is to select registers for the binary counter and the delay loop counters, and to select an output port for the display operation. Let's arbitrarily make the following assignments:

*See Signetics 2650 Applications Note AS52 - General Delay Routines.

```
Register 0 = Binary counter
Register 1 = Outer loop counter
Register 2 = Inner loop counter
Port D     = Output display port
```

We are now ready to write the program:

ADDRESS	HEX VALUE	LABEL	INSTRUCTION	COMMENTS
00	75,11	START	CPSL C + RS	Operations without Carry, Reg. bank 0
02	20		EORZ,R0	Clear R0
03	F0	OUT	WRTD,R0	Output R0 to D
04	05,20	LOOP1	LODI,R1 H'20'	Initialize outer loop
06	06,20	LOOP2	LODI,R2 H'00'	Initialize inner loop
08	FA,7E	SELF	BDRR,R2 SELF	Count inner loop
0A	F9,7A		BDRR,R1 LOOP2	Count outer loop
0C	84,01		ADD1,R0 H'01'	Add 1 to R0
0E	1F,00,03		BCTA,UN OUT	Go back to output

Let's begin entering the program using the INSTRUCTOR 50's FAST PATCH command, which is used for entering long hex data strings. The FAST PATCH mode is enabled by depressing the [REG] key followed by the [F] key:

KEY	DISPLAY		COMMENTS
[MON]	HELLO		Enter monitor mode
[REG] [F]	.Ad. =		Enter FAST PATCH
[0] [ENT/NXT]	.0000		Enter starting address
[7] [5]	.0000	75	Begin program entry.
[1] [1]	.0001	11	
[2] [0]	.0002	20	
.			
.			
.			
[1] [F]	.000E	1F	
[0] [0]	.000F	00	
[0] [3]	.0010	03	
[ENT/NXT]	0010	03	Terminate FAST PATCH

We will now verify correct entry by using the DISPLAY & ALTER MEMORY command:

KEY	DISPLAY		COMMENTS
[MEM]	.Ad. =		Display and Alter memory
[0] [ENT/NXT]	.0000	75	Address entered, data displayed
[ENT/NXT]	.0001	11	
[ENT/NXT]	.0002	20	
[ENT/NXT]	.0003	F0	
.			
.			
.			
[ENT/NXT]	.0010	03	Verification complete

If an error is detected during verfication, it can be corrected by entering the correct value before depressing the [ENT/NXT] key. For example:

KEY	DISPLAY	COMMENTS
.		
.		
[ENT/NXT]	.0003 F8	Error. Data should be F0.
[F] [0]	.0003 F0	Correct data entered.
[ENT/NXT]	.0004 05	New data deposited.
.		
.		

We are now ready to exercise the program. Before proceeding, make certain that the Interrupt Select Switch which is accessible from the bottom side of the case is in the keyboard position. (towards the center). The Port Address Select Switch is placed in the NON-EXTENDED Port D position, and, since the program begins at address zero, the [RST] key is depressed to initiate execution. The program operation can be observed on the I/O port indicators.

We can use the INSTRUCTOR 50 facilities to change the program parameters or to observe the internal operation of the program. For example, to change the delay time, we can change the delay constant at address H'05' with the DISPLAY AND ALTER MEMORY command.

KEY	DISPLAY	COMMENTS
[MON]	HELLO	Return to monitor mode.
[MEM]	.Ad. =	Display and Alter memory.
[5] [ENT/NXT]	.0005 20	Address entered, data shown.
[4] [0]	.0005 40	New constant entered.
[ENT/NXT]	.0006 06	New constant deposited.
[RST]		Program re-started.

The counter now operates about half as fast as before.

We can observe the internal operation of the program by using a breakpoint, which will stop program execution at a selected instruction and return to the monitor mode. Let's watch the outer delay loop operate by placing a breakpoint at address H'0A'. To enable the breakpoint during program execution, the program must be started via the RUN command. Before running the program, the starting address (H'00') must be entered by using the DISPLAY AND ALTER PROGRAM COUNTER (PC) command:

KEY	DISPLAY	COMMENTS
[MON]	HELLO	Return to monitor.
[BKPT] [A] [ENT/NXT]	b.P = A	Breakpoint entered.
[REG] [C] [0]	.PC = 0	Enter starting address.
[RUN]	-000A F9	Start execution. Program stops at breakpoint and returns to monitor.
[REG] [1]	.r1 = 3F	R1 has decremented by 1.
[RUN]	-000A F9	Execute again.
[REG] [1]	.r1 = 3E	R1 has decremented again.
[RUN] [REG] [1]	.r1 = 3D	And again.
[BKPT] [BKPT]	b.P =	Breakpoint removed.
[RUN]		Program runs without stopping.

The simple program outlined above is designed to demonstrate some of the capabilities of the INSTRUCTOR 50 and to give you a feel for how the system works. More comprehensive programming examples are presented in subsequent sections of this manual.

2. SYSTEM OVERVIEW

Introduction

A simplified block diagram of the INSTRUCTOR 50 system is shown in Figure 2.1 Major system components include:

- 2650 8-bit, N-channel microprocessor
- 2656 System Memory Interface (SMI)
- Sixteen-key hexadecimal keyboard
- Twelve-key function selection keyboard
- Eight-digit, 7-segment display
- Audio tape cassette interface
- S100-compatible expansion bus
- User System Executive (USE) monitor
- Debugging aids
- On-board user Input/Output
- Forced jump logic
- 512 bytes of on-board user RAM
- Crystal-controlled system clock

2650 Microprocessor

The 2650 processor is a single-chip microprocessor made using an ion-implanted, N-channel silicon-gate process. It has a fixed command set of 75 instructions, operates on 8-bit parallel data and can address 32,768 bytes of memory. All bus outputs of the 2650 are three-state and can drive either one 7400-type load, or four 74LS loads.

The 2650 contains a total of seven general-purpose registers, each eight bits long. They may be used as source or destination for arithmetic operations, as index registers, and for Input/Output (I/O) data transfers.

The processor instructions are one, two, or three bytes long, depending on the instruction. Variable length instructions tend to conserve memory space since a one or two-byte instruction may often be used rather than a three-byte instruction. The first byte of each instruction always specifies the operation to be performed and the addressing mode to be used. Most instructions use six of the first eight bits for this purpose, with the remaining two bits forming the register field. Some instructions use the full eight bits as an operation code.

The 2650 has a versatile set of addressing modes used for locating operands for operations and an interrupt mechanism which is implemented as a single level, address vectoring type. Address vectoring means that an interrupting device can force the processor to execute code at a device-determined location in memory.

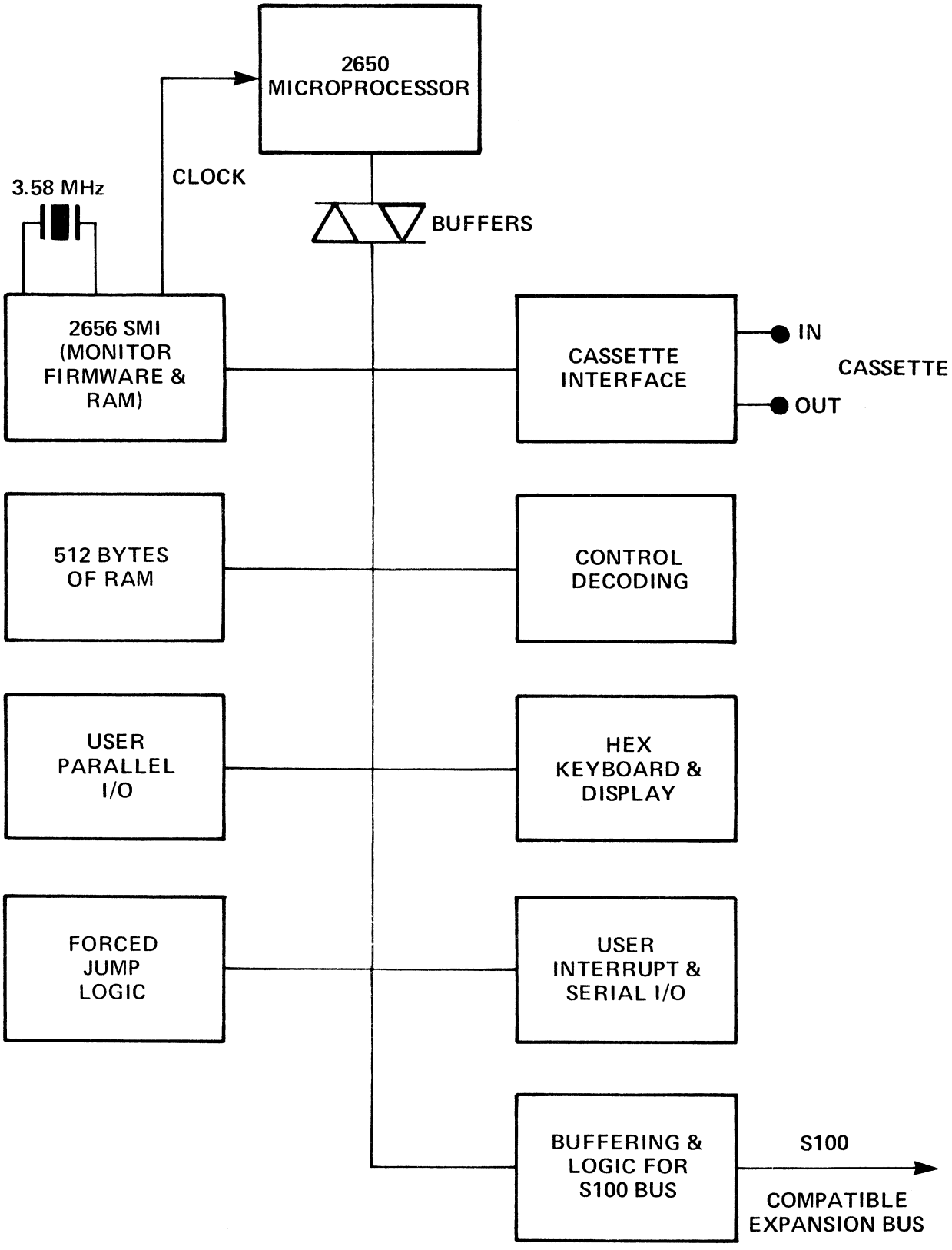


Figure 2.1: Instructor 50 Basic Block Diagram

Detailed hardware and software information on the 2650 microprocessor is provided in the accompanying Signetics 2650 Microprocessor Manual.

2656 System Memory Interface

The Signetics 2656 System Memory Interface (SMI) contains Read-Only Memory (ROM), Random-Access Memory (RAM), and a programmable I/O port. Two notable features are onboard decoders that make it possible to place the ROM and RAM anywhere in the memory space and an I/O port that can be set up as either a bidirectional port or as chip-select lines. The chip-select capability eliminates a great deal of the TTL that usually surrounds microprocessors. The 2K USE monitor, 128 bytes of scratch pad memory, I/O decode logic, and the system clock are housed in the 2656 SMI.

Keyboards

A 16-key hexadecimal keyboard and a 12-key function control keyboard enable you to communicate with the INSTRUCTOR 50. Both the hexadecimal keyboard and the function keyboard are under control of the USE monitor. The monitor performs a scanning process to determine what key has been depressed and what action is to be taken by the INSTRUCTOR 50 as a result of the depression. A functional description of the various controls and indicators is provided in Section 3.

Display Panel

The 8-digit, 7-segment display panel provides responses to input commands and guides you in the use of the INSTRUCTOR 50 by displaying prompting messages describing the data that must be entered.

Messages or responses are displayed using the seven-segment display font illustrated in Figure 2.2. Note that the characters 'b' and 'd' are always displayed with the right-hand decimal point attached in order to distinguish these characters from the number '6'.

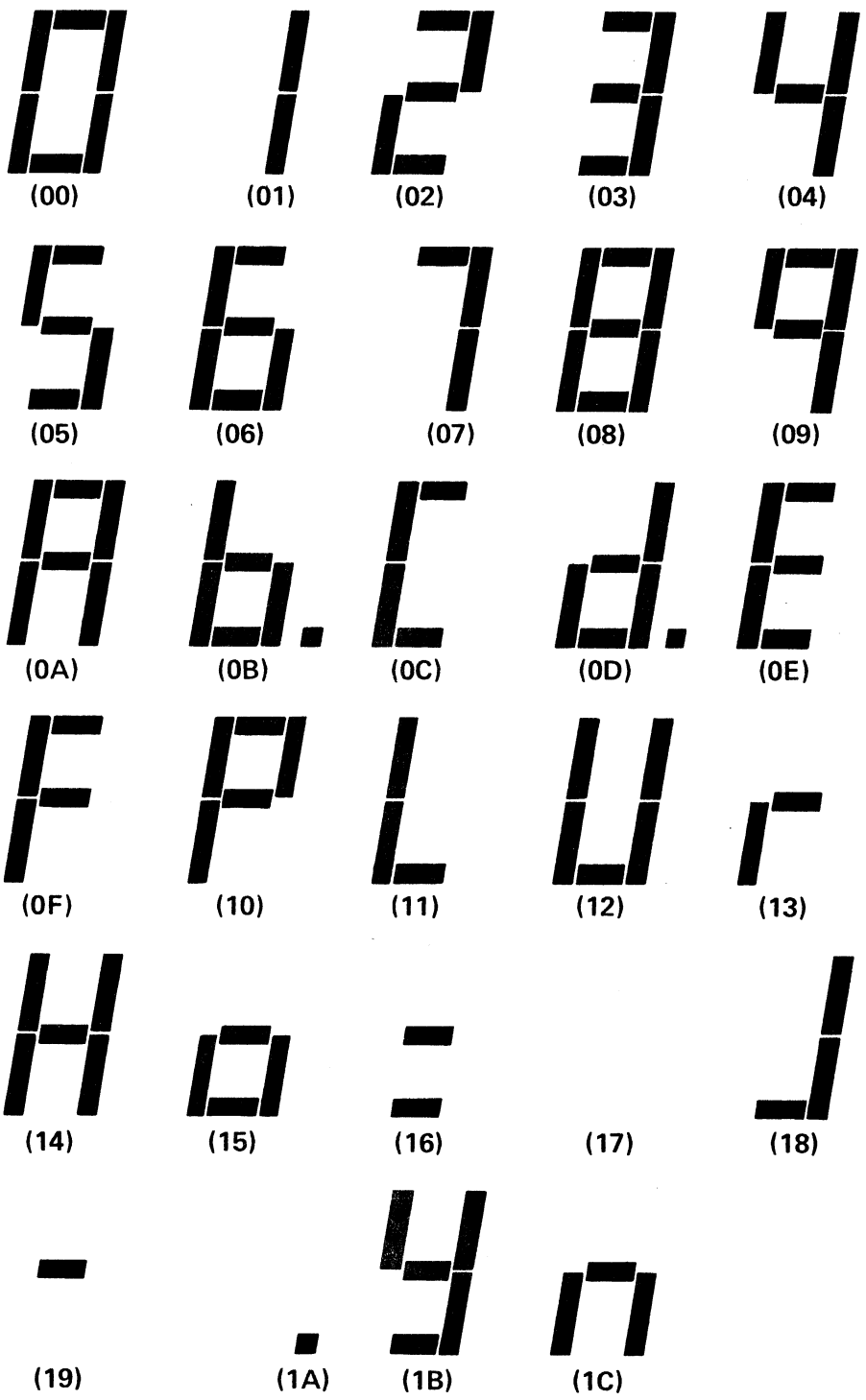
Figure 2.2 also shows the hexadecimal code required in the monitor's display buffer to display the character illustrated. To display a character with a right-hand decimal point attached, H'80' must be added to the value given. For example, H'07' will display '7', while H'87' will display '7.'. Refer to Section 5 for additional information on the use of the monitor's display subroutine.

Audio Cassette Interface

An audio cassette interface lets you load and store programs into and out of RAM. The storage medium is any audio cassette recorder.

S100-Compatible Expansion Bus

The INSTRUCTOR 50 includes an S100-compatible expansion bus connector so that other standard products, such as additional memory or prototyping cards, can be used with the system. This connector carries all of the 2650's I/O signals in addition to control signals required by the S100 bus. (See Section 6.)



( ) INDICATES THE HEX VALUE USED AS THE INTERNAL DISPLAY CODE.

NOTE: IF 80_{16} IS ADDED TO ANY CODE, A DECIMAL POINT WILL APPEAR WITH THE CHARACTER.

Figure 2.2: Instructor 50 Display Font

Monitor Firmware

The USE (User System Executive) monitor supervises operation of the INSTRUCTOR 50 and allows you to enter and alter programs, execute these programs in continuous or single-step modes, and perform a number of auxiliary functions. Monitor commands are entered via the control keys and the hexadecimal keyboard, and responses are displayed on the monitor display.

A basic flowchart of the monitor is shown in Figure 2.3. The monitor normally idles in the scan display and keyboard mode. If a key closure is detected during the scan, the monitor verifies that this is a new key closure (that any previously depressed key had been released), extinguishes the display, performs a keyboard debounce function, and then performs the requested function. The monitor then resumes the display and keyboard scan.

Monitor functions are terminated by depressing a new function key. Interrupts are inhibited while the monitor is running.

Debugging Aids

Two key features incorporated into INSTRUCTOR 50 are designed specifically for program debugging. These features are:

1. The ability to set a breakpoint that automatically interrupts execution of programs at any point without loss of hardware or software status.

2. The ability to step through a program one instruction at a time.

When a breakpoint is encountered during program execution or when a single instruction is executed in the single-step mode, control is returned to the monitor at which time you may examine the 2650 registers, the Program Status Word (PSW), and the program counter to determine the status of the microcomputer. You can then continue execution, set a new breakpoint, or resume the single-step operation. While in the monitor mode, you may change any register value, including the PSW and program counter, and you may alter memory locations.

On-Board User I/O

Both parallel and serial I/O are available in the INSTRUCTOR 50. The parallel I/O port provides 8 switch inputs and 8 individual Light-Emitting Diodes (LEDs) as a latched output port. A single LED is attached to the processor's FLAG output, and the SENS key on the function control keyboard allows you to test the processor's SENSE input. Additionally, you may exercise interrupt operation by using the interrupt (INT) key on the function control keyboard. See Section 5 for a discussion of the INSTRUCTOR 50's I/O capabilities.

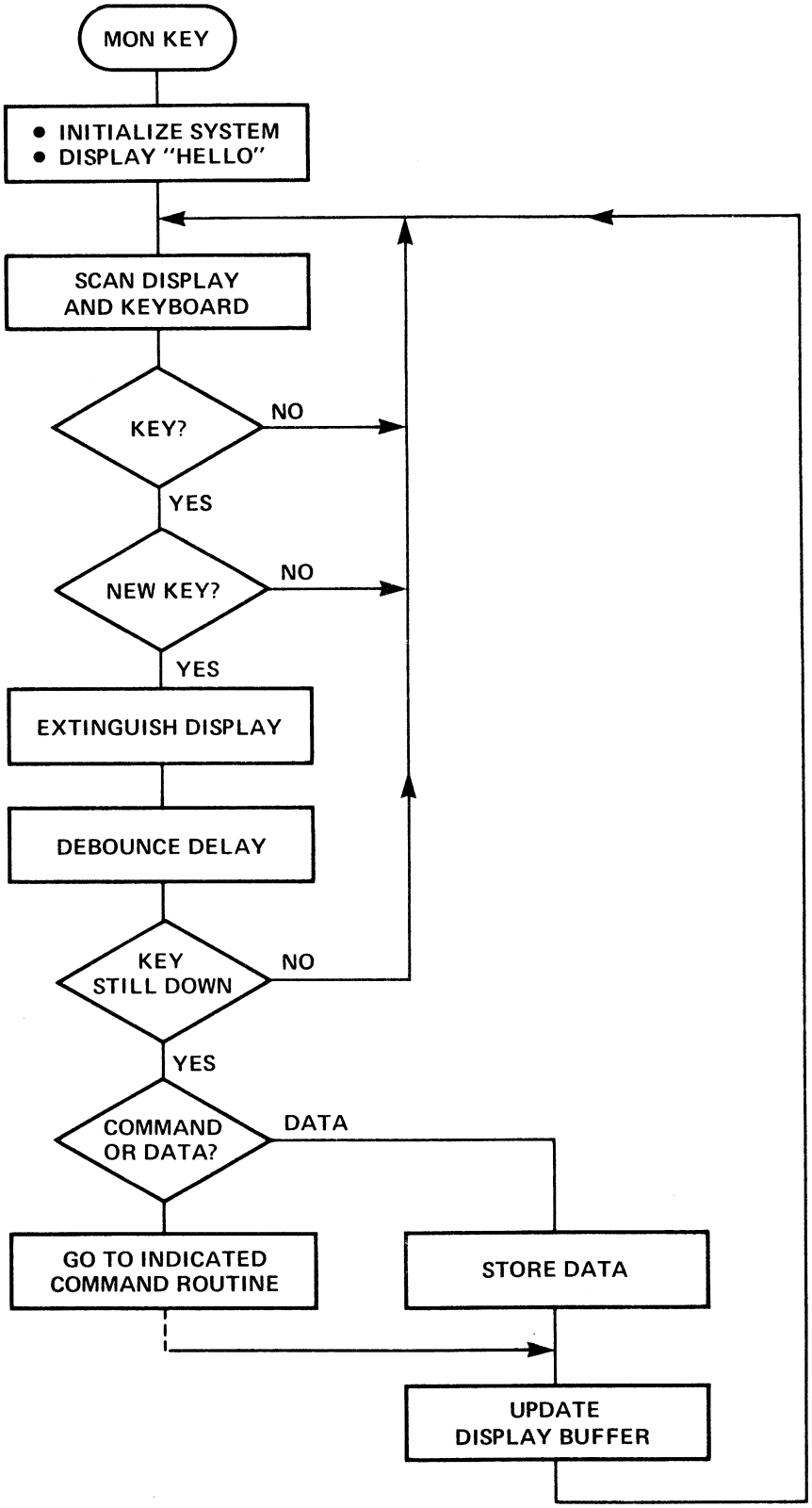


Figure 2.3: Basic Use Monitor Flow Chart

Forced Jump Logic

The Forced Jump Logic performs the following functions:

- Entry into the MONITOR mode when power is applied to the INSTRUCTOR 50 or when the MON key is depressed.
- Re-entry to the MONITOR mode after executing one instruction in single-step operation or upon detection of a breakpoint.

Memory and I/O Organization

512 bytes of RAM storage is provided for storing user programs and data. The RAM area may be expanded via the expansion bus connector.

Partitioning of the INSTRUCTOR 50's memory and I/O locations is illustrated in Figure 2.4. The supplied user memory occupies locations H'0000' to H'01FF' and may be expanded to occupy location H'0200' - H'0FFF' and H'2000' - H'7FFF'. The extended I/O ports from H'00' to H'F7' are available for program use. Ports H'F8' to H'FF' and memory locations H'1000' to H'1FFF' are reserved for the USE monitor.

An additional 64 bytes of RAM storage is available to user programs for storing data values. This additional storage space occupies memory locations H'1780' to H'17BF'. Because of the way the USE monitor operates, instructions should not be stored at these locations.

The INSTRUCTOR 50 I/O data port is assigned one of three locations, depending on the setting of the Port Address Select Switch. These are memory address H'0FFF', extended I/O address H'07', or non-extended Port D.

Clock Circuitry

The 2656 SMI provides the clock circuitry for the INSTRUCTOR 50. A 3.579545 MHz crystal is used to provide the reference frequency.

Internal Power Supply

The INSTRUCTOR 50 uses a self-contained A-C power pack that produces 8 VAC @ 1.5A. An on-board rectifier and regulator reduces this to 5 VDC. A jumper option permits the use of an alternate 8 VDC source. The INSTRUCTOR 50 may be plugged into any standard 115 VAC domestic wall socket. (European models require 220 VAC primary power).

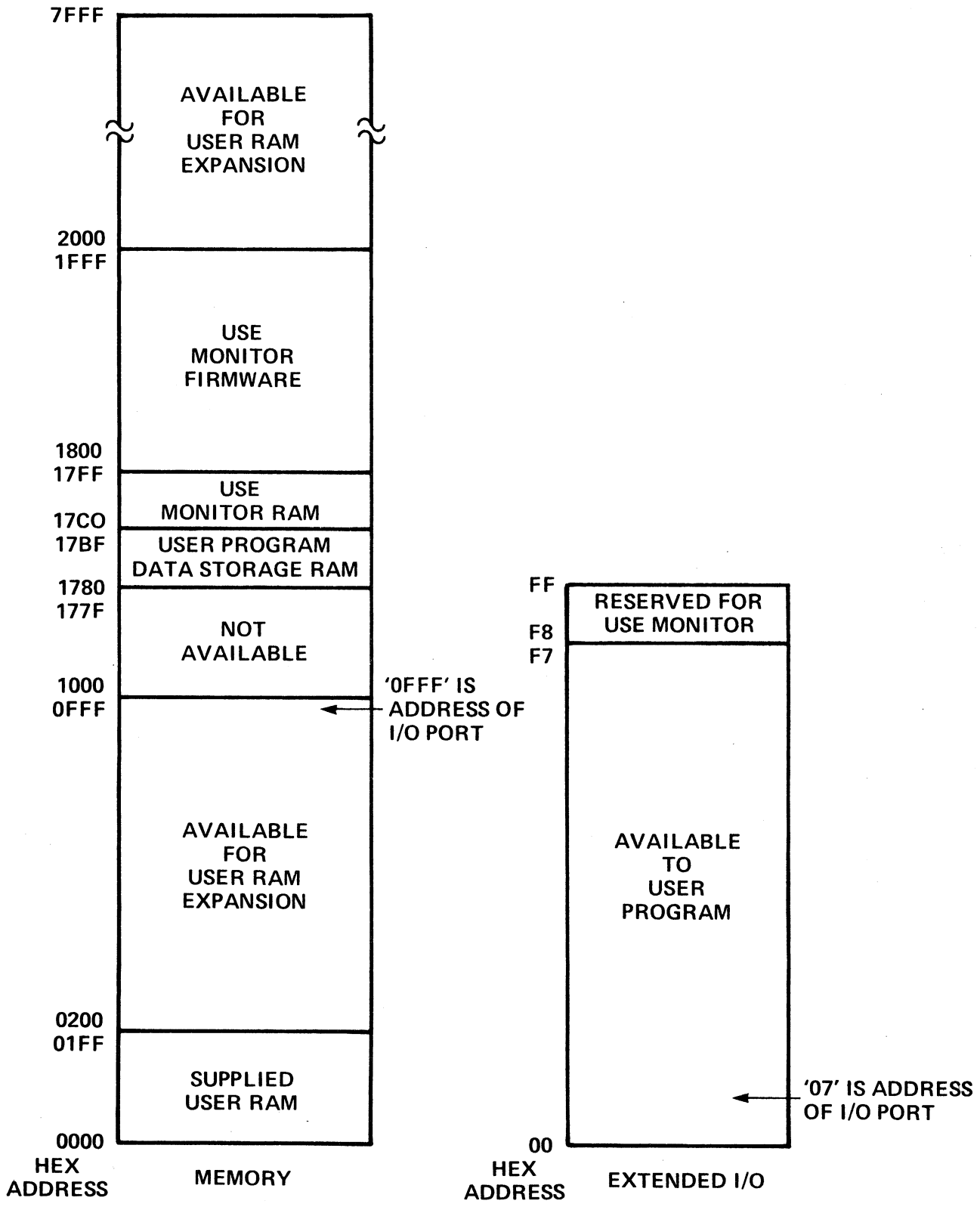


Figure 2.4: Memory And I/O Organization

3. CONTROLS AND INDICATORS

Introduction

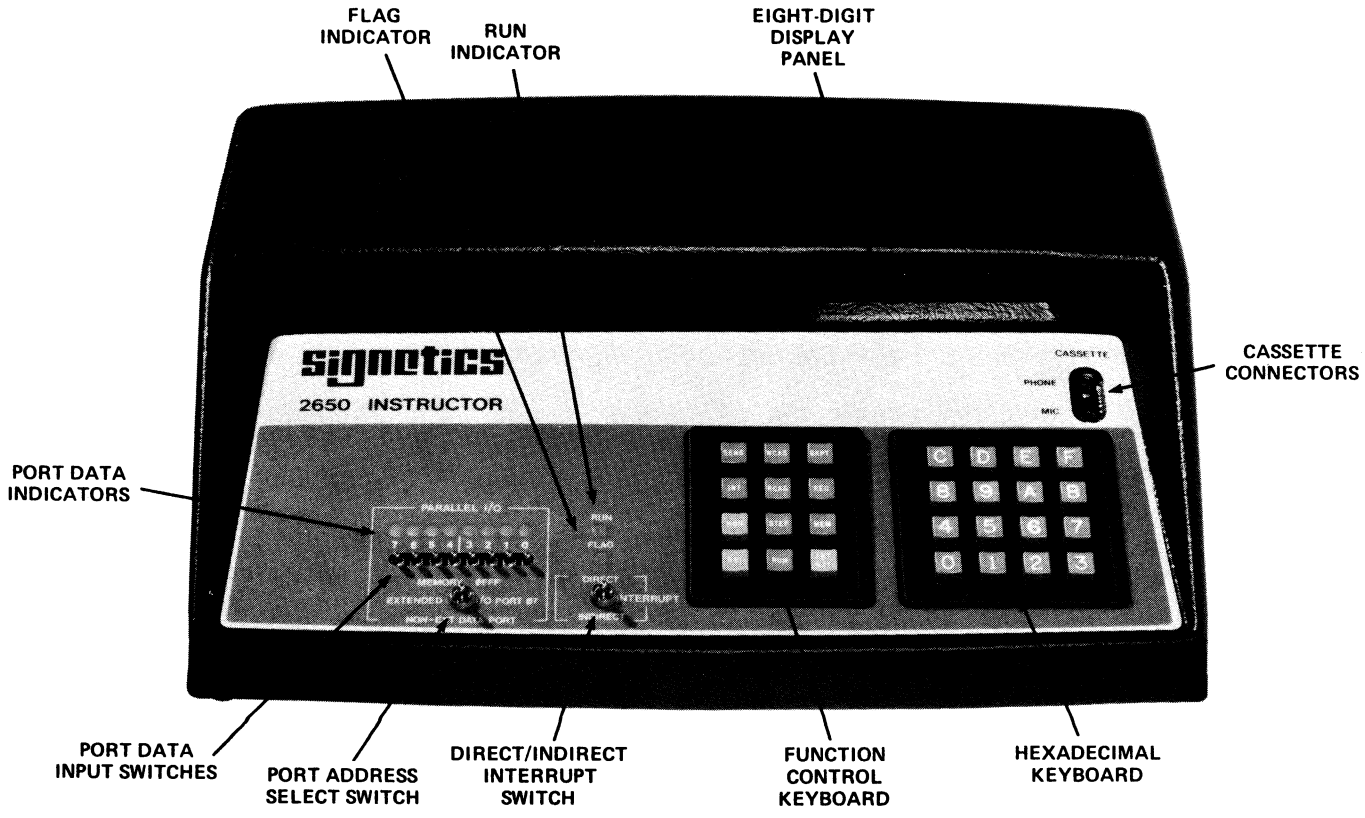
This section provides a brief functional description of the various keys, switches and indicators associated with the INSTRUCTOR 50. See Figure 3.1.

The 12-key Function Control Keyboard and the 16-key Hexadecimal keyboard enable you to communicate with, enter data, and perform the various system functions associated with the INSTRUCTOR 50. The 8-digit display is used by the USE monitor to display responses to keyed input commands. The other switches and indicators are associated with various INSTRUCTOR 50 facilities.

Function Control Keyboard

The keys in the left-most column of the function control keyboard (SENS, INT, MON, and RST) are used primarily for system control. All other keys on this keyboard perform functions associated with entry, execution, and debugging of programs.

The RST and MON keys are active at all times. All other keys except SENS and INT are normally active only during the monitor mode. Depressing these keys while executing your program has no effect. The SENS and INT keys are active only during execution of a program and have no effect on monitor operation.



However, you may take advantage of the INSTRUCTOR 50's keyboard and display facilities by incorporating calls to the monitor subroutines controlling these devices as part of your program. See section 5 for a description of these subroutines.

KEY	FUNCTION
SENS	Controls the SENSE input to the 2650 when executing a user program. The SENSE input is normally a logic '0'. Depressing the SENS key will set the SENSE input to a logic '1'.
INT	Allows you to manually interrupt the processor when executing a program. When this key is depressed, an interrupt sequence begins, resulting in the processor being vectored to or through memory location 07. The Direct/Indirect switch on the INSTRUCTOR 50 panel determines whether an instruction at location 07 is executed (Direct) or whether location 07 contains a branch address to another location in the user memory (Indirect). A switch accessible through a cutout in the bottom panel permits interrupts to be controlled by the AC line input frequency. See Section 5 for more information on INT options.
MON	Terminates any operation in process and causes the forced jump logic to output a jump instruction sequence resulting in an entry to the monitor mode. The response to a depression of the MON key is the message HELLO on the display panel.
RST	When this key is depressed, any current operation is terminated, and a RESET signal is applied to the 2650 causing program execution to begin at address zero. The system does not enter the monitor mode when this key is depressed.
WCAS	Allows programs to be transferred from the INSTRUCTOR 50 memory to audio cassette tape.
RCAS	Allows programs to be transferred from audio cassette tape to the INSTRUCTOR 50 memory.
STEP	Causes the 2650 to execute a single program instruction and return to the monitor mode, displaying the address of the next instruction to be executed on the monitor display.

RUN — Depressing this key terminates the monitor mode and causes program execution to begin at a previously specified address. Program execution continues until (1) a breakpoint is encountered; (2) the RST or MON keys are depressed; or (3) the program executes a WRTC or HALT instruction.

BKPT — Allows you to specify and examine a program breakpoint.

REG — Places the INSTRUCTOR 50 in the Display and Alter Registers mode. In this mode, you may examine and alter the contents of the 2650 general-purpose registers, the program counter value, and the value of the Program Status Word (PSW). This key is also used to initiate entry into the ADJUST CASSETTE and FAST PATCH commands. See Section 4.

MEM — Places the INSTRUCTOR 50 in the Display and Alter Memory Mode. In this mode your may specify memory locations that you wish to examine, and you may alter the contents of these memory locations.

ENT NXT — Enters keyed-in data into memory or registers and also causes the contents of the next sequential memory or register location to be displayed. The use of this key during the various monitor operations is described in the detailed command descriptions, Section 4.

Hexadecimal Keyboard

The 16-key hexadecimal keyboard (0 through 9 and A through F) is used to enter address and data parameters as required. This keyboard is also used in conjunction with the REG key on the function control keyboard to enable certain commands. See detailed command descriptions, Section 4.

Eight-Digit Hex Display Panel

The 8-digit display panel is used by the monitor to display prompting messages and responses to keyed input commands. It also displays prompting messages to guide you in the operation of the INSTRUCTOR 50.

Port Data Input Switches

These eight switches are used to specify a byte of input data at the parallel I/O port. This value is read when the 2650 executes a read I/O port instruction.

Port Data Indicators

The eight I/O port LEDs reflect the current value in the parallel output port latch. This latch is loaded with the contents of an internal register by a write I/O port instruction.

Direct/Indirect Interrupt Switch

This switch determines whether the 2650 executes a direct or indirect branch to subroutine when it acknowledges an interrupt request.

Port Address Select Switch

This switch selects the manner in which the parallel I/O port is addressed. The three modes are: non-extended I/O - Port D, extended I/O at port address 07_{16}, and memory mapped I/O at address $OFFF_{16}$.

FLAG Indicator

This LED indicates the current value of the FLAG bit in the 2650's Program Status Word. If the FLAG bit is a one, the LED is on. If the FLAG bit is a zero, the LED is off.

RUN Indicator

The RUN indicator reflects the operating status of the 2650. When the 2650 is executing either the monitor program or a user program, the RUN light is on. The RUN light is off when the 2650 has executed a HALT instruction or when the PAUSE line of the S100 interface has been driven low.

4. COMMAND DESCRIPTIONS

Introduction

This section describes the various commands available to the INSTRUCTOR 50 user. These commands include:

DISPLAY AND ALTER REGISTERS

DISPLAY AND ALTER MEMORY

FAST PATCH

DISPLAY AND ALTER PROGRAM COUNTER

BREAKPOINT

STEP

WRITE CASSETTE

ADJUST CASSETTE

READ CASSETTE

RUN

RESET

In this section, each pair of facing pages discusses a single command. The left-hand page is devoted to text, while the right-hand page actually shows what is displayed on the monitor display panel when specific keys are depressed. The circled numbers imbedded in the text on the left-hand page correspond with the circled numbers on the right-hand page.

A discussion of the INSTRUCTOR 50's error messages is presented at the end of this section.

DISPLAY AND ALTER REGISTERS

FUNCTION: This command allows you to inspect and alter, if desired, the contents of the 2650's general-purpose registers and/or Program Status Word (PSW).

PROCEDURE:

1. Depress the [REG] key (1) followed by the register address corresponding to the first register to be inspected, (2) according to the following table:

REGISTER ADDRESS	REGISTER
0	R0
1	R1, bank 0
2	R2, bank 0
3	R3, bank 0
4	R1, bank 1
5	R2, bank 1
6	R3, bank 1
7	PSU
8	PSL

2. The contents of the register are displayed as two hex digits in the data field of the display. (2)
3. The register contents may be modified at this time by keying in a new value followed by [ENT/NXT] . The numbers keyed in and appearing in the DATA display field are displayed there only and can be edited by simply keying in the correct characters (9) . The display shifts to the left each time a new character is entered, and characters shifted out of the two-digit field are lost. The hex value appearing on the display is deposited in the register when the [ENT/NXT] key is depressed. (10)
4. When the [ENT/NXT] key is depressed after step 2 or 3, the next higher register in sequence will be displayed as in step 2 (3) unless the PSL is being displayed, in which case R0 will be the next register displayed. (10)
5. The command is terminated by initiating any other command.
6. If the keys 9, B, D, or E are depressed following [REG] in step 1, the key depression will be ignored. If the keys A, C, or F are depressed, the INSTRUCTOR 50 will enter the ADJUST CASSETTE, DISPLAY AND ALTER PROGRAM COUNTER, or FAST PATCH commands, respectively. See appropriate command descriptions.

DISPLAY AND ALTER REGISTERS

EXAMPLES

	KEY	DISPLAY	COMMENTS
1	REG	r =	Awaiting register address
2	4	. r4 = 7E	R1, bank 1 = H'7E'
3	ENT NXT	. r5 = OF	R2, bank 2 = H'OF'
4	ENT NXT	. r6 = 13	R3, bank 1 = H'13'

Example A: Examine contents of R1 - R3 of bank 1

	KEY	DISPLAY	COMMENTS
5	REG	r =	Awaiting register address
6	7	. PU = 04	PSU = H'04'
7	ENT NXT	. PL = 53	PSL = H'53'
8	4 8	. PL = 48	Wrong data entered
9	4 0	. PL = 40	Correct data entered
10	ENT NXT	. r0 = 72	Entered data deposited in PSL and RO contents displayed.

Example B: Examine contents of PSW and change contents of PSL to H'40'

DISPLAY AND ALTER MEMORY

FUNCTION: Allows you to examine and optionally alter the contents of memory locations individually. This command is particularly useful when you are debugging your program and wish to examine, verify and/or change the contents of memory locations.

PROCEDURE:

1. Depress the [MEM] key (1) followed by the address of the memory location to be inspected. (2) If fewer than four digits are entered, the digits entered are used as the least-significant hexadecimal digits of the address. (2) If more then four digits are entered, the last four digits are used as the address.
2. Depress the [ENT/NXT] key (3) to display the contents of the specified memory location. The contents is displayed as two hexadecimal digits in the data field of the display.
3. You may continue to examine the contents of sequential memory locations by depressing the [ENT/NXT] key. (4) If you wish to alter the content of any memory location, enter the new data via the hexadecimal keyboard. (8) Only the last two digits entered are retained, so that an error in entry can be corrected by entering the correct data. To deposit the new data into the specified memory location, you may either depress the [ENT/NXT] key or transfer control to a new function by depressing a function key. (9)

   Each time new data is specified, the monitor performs a read-after-write check to verify that you are not attempting to write into a ROM area or into nonexistent memory. If the check fails, error message 3 is displayed. To recover from this error, depress the [MEM] key and repeat the cycle correctly.

DISPLAY AND ALTER MEMORY

EXAMPLES

	KEY	DISPLAY	COMMENTS
(1)	MEM	.Ad. =	Awaiting memory address
(2)	1 0	.Ad. = 10	10 = Address of memory location to be examined
(3)	ENT NXT	.0010 02	H'02' = contents of memory location 0010
(4)	ENT NXT	.0011 FF	Address and contents of next sequential memory location

Example A = Examine contents of memory location 0010 and move to next sequential memory location

	KEY	DISPLAY	COMMENTS
(5)	MEM	.Ad. =	Awaiting memory address
(6)	2 2	.Ad. = 22	Address of memory location to be examined
(7)	ENT NXT	.0022 06	H'06' = Contents of memory location 0022
(8)	0 5	.0022 05	Desired contents of memory location 0022 entered and displayed.
(9)	REG	.r =	H'05' deposited into memory location 0022, Display and Alter Memory Command is terminated, and monitor enters Display and Alter Registers Command.

Example B: Examine contents of memory location 0022, change data, and transfer control to another function.

FAST PATCH

FUNCTION: The FAST PATCH command allows you to enter long strings of data into memory from the hexadecimal keyboard. Once the starting address is selected, data is loaded into memory sequentially--one byte for every two hex keys depressed. Once keyed in, data may not be changed in the FAST PATCH mode. To change data, you must use the DISPLAY AND ALTER MEMORY command or re-enter the FAST PATCH command starting at the address where the change is required.

PROCEDURE

1. To enter the FAST PATCH command, depress the [REG] key (1) on the function control keyboard followed by [F] on the hexadecimal keyboard. (2)
2. Enter the desired starting address on the hexadecimal keyboard. (3)

   NOTE: You may bypass this step and go directly to step 3 to begin at a known starting address. The starting address is known under any one of the following conditions:

   a) When a file has been read into memory from a cassette tape by the INSTRUCTOR 50. The file's starting address will be the beginning address for the FAST PATCH.

   b) The address from which the last exit from the DISPLAY AND ALTER MEMORY or FAST PATCH command took place.
3. Depress the [ENT/NXT] key (4) on the function control keyboard to set the starting address. Data may now be entered into the specified address.
4. Enter desired data for the displayed address as two hex digits. (5) Continue entering data in this manner until all data is entered. The INSTRUCTOR 50 automatically increments the memory address as data is entered. (6) (7) (8) (9)
5. Exit the FAST PATCH mode by depressing [ENT/NXT] or another function key. (10)
6. A read-after-write check is performed as each byte is deposited. The INSTRUCTOR 50 will display Error 3 if data cannot be stored.

FAST PATCH

EXAMPLE

	KEY(s)	DISPLAY	COMMENTS
1	REG	r =	
2	F	.Ad. =	Awaiting starting memory address
3	1 0	.Ad. = 10	Starting address entered
4	ENT NXT	.0010	Starting address set
5	1 2	.0010 12	Data entry
6	1 3	.0011 13	
7	1 4	.0012 14	
8	1 5	.0013 15	
9	1 6	.0014 16	
10	MEM	.Ad. =	Exit from FAST PATCH mode

Enter Data String "12 13 14 15 16" into
Successive Memory Locations Starting at Address H'10'

DISPLAY AND ALTER PROGRAM COUNTER

FUNCTION: The DISPLAY AND ALTER PROGRAM COUNTER command allows you to examine or change the address of the first instruction to be executed by the 2650 during execution of a RUN or STEP command.

PROCEDURE:

1. To enter the DISPLAY AND ALTER PROGRAM COUNTER command, depress the [REG] key (1) on the function control keyboard followed by [C] on the hexadecimal keyboard. (2)
2. The display will show the current Program Counter (PC) value as four hexadecimal digits. (2)
3. If you want to change the PC address, enter the desired address on the hexadecimal keyboard. (3)

   NOTE: For a multiple-byte instruction, the address entered is the address of the first byte.
4. Depress any command key (4) on the function control keyboard to set the desired starting address. If the [ENT/NXT] key is used, the INSTRUCTOR 50 transfers control to the DISPLAY AND ALTER REGISTERS command.

DISPLAY AND ALTER PROGRAM COUNTER

EXAMPLE

	KEY	DISPLAY	COMMENTS
(1)	REG	r =	
(2)	C	.PC = 0015	0015 = present contents of Program Counter
(3)	1 7	.PC = 17	Starting address changed to 0017
(4)	ENT NXT	r =	Sets new starting address, and transfers control to DISPLAY AND ALTER REGISTERS Command

Set Starting Address for RUN Command to H'0017'

BREAKPOINT

The BREAKPOINT COMMAND allows you to enter, clear, or examine a program breakpoint. A breakpoint returns system control from the executing program to the monitor and enables you to examine the state of the memory and processor registers, make modifications, if desired, and continue program execution from the point of interruption.

PROCEDURE:

1. Depress the [BKPT] key on the function control keyboard (1) to place the INSTRUCTOR 50 in the breakpoint mode.
2. The monitor will display either:
   a) A blank data field if a breakpoint address was not specified previously. (1)
   b) The address of the breakpoint previously entered. (5)
3. Enter the desired breakpoint address on the hexadecimal keyboard. (2) If the desired address is already displayed, as in step (2b), re-entry is not required.

   NOTE: If a breakpoint is set at a multiple-byte instruction, the address specified for the breakpoint should be the address of the first byte.
4. Depress the [ENT/NXT] key (3) or another function key (4) to set the breakpoint at the address displayed.
5. To clear a breakpoint, depress the [BKPT] key twice in succession. (5) (6)

NOTE: The breakpoint is inserted into your program when you enter the execution mode via the RUN command. When the breakpoint is encountered during program execution, the breakpoint address and contents are displayed, preceded by a "-" (minus) sign. The instruction at the breakpoint address is restored and executed prior to this display, and the Program Counter is updated to the address of the instruction following the breakpoint.

ERROR MESSAGES

During specification of the breakpoint address, the INSTRUCTOR 50 may display one of the following error messages:

ERROR 1 If the user attempts to specify a breakpoint address in the INSTRUCTOR 50's ROM address space or in non-existent memory. To clear this error, depress [BKPT] once.

ERROR 2 If the user attempts to enter a new breakpoint address after having set a previous breakpoint address by depression of the [ENT/NXT] key. To clear this error, depress any function key. The original breakpoint address will be saved.

BREAKPOINT

EXAMPLE

	KEY(s)	DISPLAY	COMMENTS
(1)	BKPT	.b.P =	No previous breakpoint specified. Waiting for breakpoint address.
(2)	4 4	.b.P = 44	Breakpoint address entered.
(3)	ENT NXT	b.P = 0044	Breakpoint address set.
(4)	REG	r =	Breakpoint address set by exiting to another function.
(5)	BKPT	.b.P = 0044	Breakpoint address displayed.
(6)	BKPT	b.P =	Breakpoint cleared.

Set Breakpoint at Address H'0044' and then clear it.

STEP

FUNCTION: Causes the 2650 to execute a single instruction and return to the MONITOR mode, displaying the address of the next instruction to be executed on the monitor display.

PROCEDURE:

1. Enter the address of the first instruction to be executed as described under DISPLAY AND ALTER PROGRAM COUNTER command. (1)
2. Depress the [STEP] key. (2) The INSTRUCTOR 50 will execute a single instruction and display the address of the next instruction to be executed and the data at that address.
3. At this point you may examine and alter memory and/or register values if desired by using the appropriate commands.
4. Continue as in step 2 to repeat the single-step operation. (3) (4)
5. To exit the single-step mode, depress any function key. (5)
6. Note that a breakpoint, if entered, is ignored during single-step operation.

The single-step sequencer and the forced jump logic are used in this mode of operation. Following is the sequence of operations executed by the monitor when the [STEP] key is depressed:

a) The monitor SINGLE STEP flag is set.

b) Register contents previously stored upon entry to the monitor are restored to the 2650.

c) The monitor executes a "hidden single step" to determine how many cycles are contained in the instruction to be stepped.

d) The monitor permits execution of one user program instruction by counting the predetermined number of cycles.

e) The registers (R0 - R3, R1' - R3' and PSW) are saved.

f) The Program Counter is updated to the next instruction.

g) The address in the Program Counter and data at that address are displayed. The SINGLE STEP flag is cleared.

h) The monitor exits to the KBD SCAN routine to await user's input.

EXAMPLE

	KEY(s)	DISPLAY	COMMENTS
1	REG C 8 ENT NXT	r =	Starting address H'0008' entered
2	STEP	000A 42	Single step executed.* Next instruction is at H'000A', and op-code is H'42' (ANDZ, R2)
3	STEP	000B CC	Next instruction op-code is H'CC' (STRA, R0)
4	STEP	000E 20	Next instruction op-code is H'20' (EORZ, R0)
5	REG	r =	Exit single step

Single step three instructions starting at address H'0008

*Since the displayed address is two greater than the starting address (H'000A' - H'0008' = 2), the first instruction executed was a two-byte instruction.

WRITE CASSETTE

FUNCTION: The WRITE CASSETTE command allows you to write programs and data from memory onto cassette tape. Any good quality audio cassette tape recorder may be used as the output device. The data transfer rate is approximately 300 bits per second.

PROCEDURE:

General Installation

- Connect the INSTRUCTOR 50's Cassette-Out Jack to the microphone (MIC) input of the cassette deck using the appropriate cable supplied with the INSTRUCTOR 50 package.
- Install tape in transport.
- Make certain that the tape is positioned so that previously recorded files will not be destroyed when the WCAS command is issued.
- Adjust recorder's input level control, if one is provided, to normal recording level.

Operation

1. Depress the [WCAS] key (1) to place the INSTRUCTOR 50 in the WRITE CASSETTE mode.
2. Enter the lower (beginning) address of the file to be written. (2)
3. Depress the [ENT/NXT] key (3) to set the lower address.
4. Enter the upper (ending) address of the file to be written. (4)
5. Depress the [ENT/NXT] key (5) to set the upper address.
6. Enter the program start address (the address at which you want your program to begin executing). (6)
7. Depress the [ENT/NXT] key (7) to set the start address.
8. Enter the file identification (ID) number. (8)
   NOTE: The file ID may be any hex value between 00 and FF. If no ID is entered, the default file number is 00.
9. Place the cassette deck in the RECORD mode.
10. Depress [ENT/NXT] key. (9) This starts a five second delay prior to actual memory dump to tape. The INSTRUCTOR 50 flashes the FLAG Indicator at one-second intervals during this delay. The message HELLO is displayed (9) when data transfer to tape is completed.
11. During the recording process, a visual indication of the 'dump' can be observed on the I/O port indicators by placing the I/O Port Address Select Switch in the EXTENDED (center) position.

Tape Deck Shutdown

- Turn the audio tape recorder off.
- If the tape deck has a counter, note its value for future reference.
- Disconnect tape deck and remove and store tape cartridge.

Error Messages

The INSTRUCTOR 50 will display the message 'Error 7' if the value of the specified upper address is less than the value of the lower address.

EXAMPLE

	Key(s)	Display	Comments
①	WCAS	L.Ad. =	Waiting for lower address of file to be written onto tape
②	0	L.Ad. = 0	Lower address entered
③	ENT NXT	U.Ad. =	Lower address set. Waiting for upper address.
④	7 6	U.Ad. = 76	Upper address entered
⑤	ENT NXT	S.Ad. =	Upper address set. Waiting for start address.
⑥	10	S.Ad.= 10	Start address entered.
⑦	ENT NXT	. F =	Start address set. Waiting for file number.
⑧	1	. F = 1	File ID entered.
⑨	ENT NXT	HELLO	File address set. Write data to cassette tape completed.

Write a file to tape with the following parameters:

File Number = 1

Beginning Address = 0

Ending Address = H'76'

Program Start Address = H'10'

ADJUST CASSETTE

FUNCTION: The ADJUST CASSETTE command allows you to adjust the output level of a cassette recorder for proper interface to the INSTRUCTOR 50 during a READ CASSETTE operation.

While most conventional audio cassette recorders are compatible for use with the INSTRUCTOR 50, the playback volume control must be accurately adjusted to ensure proper detection of data by the INSTRUCTOR 50. Otherwise, the data signal may be distorted (volume too high) or may drop below detection thresholds (volume too low).

PROCEDURE:

General Installation

1. Check to ensure that the cassette recorder's playback heads and transport mechanism are clean and free from any obstructions.

2. Install tape in transport and rewind to an area known to contain a previously recorded file. Use of the sample tape supplied with the INSTRUCTOR 50 is recommended.

3. Connect the INSTRUCTOR 50's PHONE jack to the cassette deck's PHONE or SPEAKER output jack using the appropriate cable supplied with the INSTRUCTOR 50 package.

Operation

1. Place the INSTRUCTOR 50 in the ADJUST CASSETTE mode by depressing the [REG] key on the function control keyboard followed by [A] on the hexadecimal keyboard. (1)

2. Start playback of previously recorded data.

3. Adjust tape deck VOLUME or LEVEL control. The following three digits will be displayed intermittently during the adjustment process:

   U Increase volume

   d. Decrease volume

   - volume control adjusted correctly

4. When a minus sign (-) (3) is displayed, the audio cassette's playback volume is properly adjusted.

5. During the adjust process, the I/O Port indicators can also be used to observe data being read by the INSTRUCTOR 50 if the I/O Port Address Switch is placed in the EXTENDED (center) position. The display has the following significance:

All LEDs OFF	Indicates proper operation or no data.
Some negative number (LED bit 7 ON)	Indicates that the playback level is too low - not enough pulses
Some positive number (LED bit 7 OFF)	Indicates that the playback level is too high. Tape "noise" is being detected - too many pulses.

6. When level is properly set, turn off the cassette deck.

7. Depress the [MON] key ④ to exit from the ADJUST CASSETTE routine.

EXAMPLE

	Key(s)	Display	Comments
①	[REG] [A]	U	Places INSTRUCTOR 50 in the ADJUST CASSETTE mode. Increase playback level.
②		d.	Decrease playback level
③		-	Playback level properly set
④	[MON]	HELLO	Exit ADJUST CASSETTE mode

READ CASSETTE

FUNCTION: The READ CASSETTE command allows you to read files previously stored on cassette tape using the WRITE CASSETTE command and store these files in the specified RAM locations.

PROCEDURE:

General Installation

1. Check to ensure theat the cassette recorder's playback heads and transport mechanism are clean and free from any obstructions.

2. Install tape in transport and rewind to desired file location.

3. Connect the INSTRUCTOR 50's PHONE jack to the cassette deck's PHONE or SPEAKER output jack using the appropriate cable supplied with the INSTRUCTOR 50 package.

4. Adjust playback level to setting previously determined to be proper by ADJUST CASSETTE operation (See ADJUST CASSETTE command).

Operation

1. Depress the [RCAS] key ① to place the INSTRUCTOR 50 in the READ CASSETTE mode.

2. Depress one or two hex digits ② corresponding to the file number desired to be read back.

   NOTE: the user may elect to read the first file encountered by omitting this step.

3. Depress the [ENT/NXT] key ③ to set the file ID number.

4. Start the cassette deck in playback mode. The reading of data by the INSTRUCTOR 50 can be visually observed on the I/O Port indicators by placing the I/O Port Address Switch in the EXTENDED (center) position.

5. When the reading of the specified file is completed, the INSTRUCTOR 50 will display the HELLO message. ④

6. Turn off the audio cassette deck.

7. Data read from tape will be placed at consecutive memory locations starting at the beginning address specified when the file was created. The Program Counter (PC) will be set to the address specified as the program start address when the file was created.

Error Messages

During the read-in process, any one of the following error messages may be displayed:

- Error 4 - Cassette Block Check Character (BCC) error
- Error 5 - Read Cassette Memory Write Error
- Error 6 - Read Cassette character from tape not ASCII HEX

EXAMPLE

	Key(s)	Display	Comments
①	RCAS	.F=	Places the INSTRUCTOR 50 in the READ CASSETTE mode. Waiting for file ID number
②	1	.F= 1	File ID number entered
③	ENT/NXT		Sets file ID number. Begins reading data into memory*
④		HELLO	File is fully loaded into memory

* Flashing I/O Port indicators at this point indicate that the file is being read.

RUN

FUNCTION: Terminates the monitor mode and causes program execution to begin at the address specified in the Program Counter. Program execution continues until 1) a breakpoint is encountered, 2) the RST or MON key is depressed, or 3) the user program executes a WRTC (Write to Port C) or HALT instruction.

The RUN command allows program execution to begin at any point in the user program. It is particularly valuable, when used in conjunction with a set breakpoint, for debugging sections of a program. When the RUN key is depressed, the INSTRUCTOR 50 performs the following actions:

1. If a breakpoint was set, the WRTC code is inserted at the specified breakpoint address and a monitor 'BREAKPOINT ENABLED' flag is set. This flag distinguishes a breakpoint 'WRTC' from any other 'WRTC' in the user program when control is returned to the USE monitor by the forced jump logic upon execution of a WRTC instruction.

2. The processor registers are restored to the last values existing when control was returned to the USE monitor after a breakpoint or single step, or to the values specified by you in a DISPLAY AND ALTER REGISTERS operation.

3. The INSTRUCTOR 50 switches to the execution mode by jumping to the address specified in the Program Counter. This address will be the address of the next instruction following a breakpoint or single step, or the address specified by you in a DISPLAY AND ALTER PROGRAM COUNTER operation.

RESET

FUNCTION: When the RST (RESET) key is depressed, current INSTRUCTOR 50 activity is terminated immediately, and the processor begins program execution at address H'0000'. Breakpoint and single-step flags, if set, are ignored. A high (logic one) level appears on the expansion connector RESET pin for as long as the key remains depressed.

When the RESET key is used to initiate program execution from location H'0000', the initial processor register values are unknown, and a breakpoint, if previously specified, is not inserted in the user program. Program execution continues until any one of the following occurs:

1. The RESET key is depressed again.

2. A HALT instruction (H'40') is executed. Upon detection of a HALT instruction, the processor halts until the RESET key is depressed again or, if the Interrupt Inhibit PSW bit was not set, until an interrupt occurs.

3. A WRTC instruction is executed or the MON key is depressed. Control is transferred to the USE monitor and the HELLO message is displayed. When control is returned to the monitor, the address of the last memory fetch is saved in the Program Counter, and register values are saved in monitor RAM. These may be examined by using the appropriate commands.

4. The processor's PAUSE input is raised high via the expansion connector. When this occurs, the RUN indicator light is extinguished. Program execution will begin at the next instruction when PAUSE goes low.

ERROR MESSAGES

The USE monitor incorporates extensive error checking firmware. If an error is encountered while attempting to execute a command, a message of the form 'Error n' is presented on the monitor display. Error messages are summarized in Table 4.1.

• Error 1	BREAKPOINT CANNOT BE SET
• Error 2	INVALID COMMAND
• Error 3	ALTER OR PATCH MEMORY WRITE ERROR
• Error 4	CASSETTE BCC ERROR
• Error 5	READ CASSETTE MEMORY WRITE ERROR
• Error 6	CHARACTER FROM TAPE NOT ASCII HEX
• Error 7	START ADDRESS GREATER THAN STOP ADDRESS
• Error 8	KEYBOARD HAS 2 KEYS IN COLUMN DOWN
• Error 9	NEXT SINGLE STEP IS INTO MONITOR

TABLE 4.1 Error Messages

Additional information on each of the above error messages is presented in the following paragraphs.

Error 1 *BREAKPOINT CANNOT BE SET*

The display message Error 1 indicates that an attempt was made to set a breakpoint at a memory address which is not RAM. A breakpoint is entered by inserting the WRTC,R0 code H'B0' into the memory address specified. A read-after-write check is then performed. If this test fails, the error message is displayed.

Error 2 *INVALID COMMAND*

The display message Error 2 indicates that an incorrect command sequence was entered via the keyboard.

Error 3 *ALTER OR PATCH MEMORY ERROR*

The display message Error 3 indicates that an attempt was made to change the data at a memory address which is not RAM. When changing memory data during an Alter Memory or Patch Memory operation, a read-after-write check is performed. If this test fails, the error message is displayed.

Error 4 *CASSETTE BCC ERROR*

When data is written on tape with the WRITE CASSETTE command, a Block Check Character (BCC) is appended to the end of the file. The BCC is re-calculated when data is read back with a READ CASSETTE command and compared with the BCC recovered from the tape. If the BCC's do not match, the message Error 4 is displayed, indicating that some problem has occurred in reading the tape.

Error 5 *READ CASSETTE MEMORY WRITE ERROR*

Data read back from the tape is stored in the INSTRUCTOR 50 at consecutive memory locations starting at the address specified in the tape file. A read-after-write check is performed on each byte stored. If the test fails, the message Error 5 is displayed.

Error 6 *CHARACTER FROM TAPE NOT ASCII HEX*

Data written on tape uses the ASCII code for the characters 0 through F. The display message Error 6 indicates that a non-hex character was recovered from the tape. Correct adjustment of playback level should be verified using the ADJUST CASSETTE command.

Error 7 *START ADDRESS GREATER THAN STOP ADDRESS*

The display message Error 7 indicates that the start address in the WRITE CASSETTE command is greater than the specified stop address. The operation cannot be performed. See Section 4.

Error 8 *KEYBOARD HAS 2 KEYS IN COLUMN DOWN*

The Error 8 message is displayed when the monitor detects that two keys are depressed simultaneously. The monitor cannot decode the action desired.

Error 9 *NEXT SINGLE STEP IS INTO MONITOR*

Single-step operation in the memory area reserved for the USE monitor (H'1000' - H'1FFF') is not permitted and will cause unpredictable results if executed. The display message Error 9 is a warning that such a single-step operation was attempted.

5. USING THE INSTRUCTOR 50

Restrictions on Using the 2650 Instruction Set

When writing programs, the INSTRUCTOR 50 user has the complete 2650 microprocessor instruction set at his disposal. However, because of the interaction between the USE monitor and user hardware and software, certain restrictions must be observed:

1) The USE monitor reserves the WRTC, Rx instruction (H'B0' - H'B3') to indicate the location of a breakpoint in a user program. If this instruction is executed in a user program, control of the system will return to the monitor, and the message HELLO will be displayed.

2) If a HALT instruction (H'40') is executed, processor operation will terminate. This is indicated by the RUN indicator being extinguished. The only ways to reinitiate operation are to depress the [RST] key or, if interrupts were not inhibited, to cause an interrupt by depressing the [INT] key.

   If a breakpoint is set at a HALT instruction location, the monitor will prevent execution of the HALT, and normal operation will continue.

3) The top of memory page zero is occupied by the USE monitor program. Therefore, the ZBSR and ZBRR instructions with negative displacements should not be used unless entry into the monitor program is desired. The same applies to interrupt vectors with negative displacements.

4) The USE monitor uses three levels of the 2650 subroutine Return Address Stack (RAS) in its operation. Since the RAS is limited to eight levels, user programs being developed under control of the USE monitor should be limited to a maximum of five levels of subroutines, including interrupt levels.

Using Interrupts

Interrupts provide a method of interfacing a synchronous program to asynchronous external events. An Interrupt Request forces the 2650 to temporarily suspend execution of the program currently running and branch to an interrupt service routine. Upon completion of the interrupt service routine, the 2650 resumes execution of the interrupted program.

The INSTRUCTOR 50 provides three methods of interrupting the 2650. The first method is a manual interrupt using the [INT] key on the function keyboard. The second method uses a 60Hz signal derived from the INSTRUCTOR 50's power supply to generate interrupt requests once every 16.7 ms. This option accommodates user programs that require a real-time clock. (For European systems, the real-time clock interrupts occur at a 50Hz rate or once every 20 ms). The third method of interrupting the INSTRUCTOR 50 is via the S100 bus interface. This section describes the 2650's interrupt mechanism and provides details on selecting the interrupt options.

The 2650's interrupt mechanism can be selectively enabled or disabled at various points in a user program by setting or clearing the Interrupt Inhibit (II) bit of the processor's Program Status Word (PSW). If the Interrupt Inhibit bit has been set, the 2650 ignores interrupt requests. The Interrupt Inhibit bit may be cleared (thus enabling interrupts) in any of the following four ways:

1) By reseting the processor (depressing the RST key);

2) By executing a Clear Program Status Upper (CPSU) instruction with the proper mask value;

3) By executing a Return from Subroutine and Enable Interrupt (RETE) instruction; or

4) By executing a Load Program Status, Upper (LPSU) intruction.

The interrupt mechanism of the 2650 operates with a vectored interrupt. When the processor accepts an interrupt request, it responds by issuing an INTerrupt ACKnowledge (INTACK). Upon receipt of INTACK, the interrupting device responds by placing an "interrupt vector" on the 2650 data bus. This vector is used as the address, relative to byte zero, page zero, of a branch to subroutine instruction. The interrupt vector may specify either direct or indirect addressing. A vector that specifies direct addressing causes the 2650 to execute a subroutine branch to the address specified by the vector. If an indirect address

is specified, the interrupt vector points to the first of two successive memory locations (interrupt vector and interrupt vector + 1) where the address of the interrupt subroutine is stored. In this case, the processor first fetches the two interrupt subroutine address bytes and then branches to the subroutine. Thus, a direct interrupt vector transfers the program to any location from -64 to +63 relative to byte zero, page zero, and an indirect interrupt vector can transfer the program to any location within the 2650's 32K addressing range.

If the Interrupt Inhibit bit has been cleared, the INSTRUCTOR 50 responds to an interrupt request with the following sequence of events:

1) The 2650 completes execution of the current instruction.

2) The processor sets the Interrupt Inhibit bit of the PSW (=1)

3) The first byte of a Zero Branch to Subroutine Relative (ZBSR) instruction is inserted in the 2650's internal instruction register.

4) The processor issues INTACK and waits for an interrupt vector to be returned on the data bus.

5) The INSTRUCTOR 50's interrupt logic places the interrupt vector (H'07' or H'87') on the data bus. Whether the interrupt vector specifies direct (H'07') or indirect (H'87') addressing is determined by the setting of the Direct/Indirect switch on the front panel. If the switch is in Direct position, the next instruction executed is the instruction at address H'07'. If the switch is in the Indirect position, the next instruction executed is at the address contained in H'07' and H'08'.

6) The 2650 executes the ZBSR instruction. The address of the next instruction in the interrupted program is stored in the 2650's internal subroutine address stack, and the stack pointer is incremented.

7) When the interrupt subroutine is terminated with an RETE or RETC instruction, the 2650 decrements the stack pointer, replaces the current value of the Program Counter with the address previously stored in the subroutine stack, and resumes execution of the interrupted program.

Since the INSTRUCTOR 50 interrupt logic vectors interrupt requests through memory address H'07', user programs that support direct interrupts must place the first byte of the interrupt subroutine at this address. If indirect subroutines are used, the address of the interrupt subroutine must be stored at memory locations H'07' and H'08'.

As interrupts may occur at any point in a user program, it is entirely possible that they will affect the contents of the 2650's internal registers with unpredictable results for the interrupted program. This problem can be solved in two ways. The first way is to tightly control the portions of a user program that can be interrupted by selectively setting and clearing the Interrupt Inhibit bit in the PSW. The second method is to save the 2650's internal registers and Program Status Word upon entering the interrupt subroutine and restoring them before returning from the subroutine.

The INSTRUCTOR 50's interrupt modes can be selected by a combination of switch settings and a jumper option on the printed circuit board. As mentioned previously, the Direct/Indirect swtich on the INSTRUCTOR 50's front panel determines whether the interrupt vector generated by the interrupt logic specifies direct or indirect addressing. Whether the 2650 responds to the INT key or the 60 Hz real-time clock is determined by the setting of a slide switch located at the bottom of the INSTRUCTOR 50 case. Optionally, devices external to the INSTRUCTOR 50 can generate interrupt requests via the S100 bus interface. To accomplish this, a jumper option described in the last part of this section is used. Following are two programming examples that make use of the INSTRUCTOR 50's interrupt facilities:

FOLLOWING ARE TWO PROGRAMMING EXAMPLES THAT MAKE USE OF THE INSTRUCTOR 50's INTERRUPT FACILITIES:

Example 1 - Direct Interrupt

This example is a complete program that first clears the parallel I/O port lights and then maintains a binary counter at the I/O port lights. The count is incremented each time the [INT] key is depressed. Prior to running this program, you must place the Direct/Indirect switch in the Direct position, and the I/O port address select switch in the Non-Extended position.

Address	Data	Instruction Mnemonic	Comment
0000	76,20	PPSU H'20'	Set II - inhibit interrupts
0002	75,08	CPSL H'08'	Operations without carry
0004	1F,00,0A	BCTA,UN,H'000A'	Branch over interrupt subroutine
0007	84,01	ADD1,R0,H'01'	Increment R0 (counter)
0009	17	RETC,UN	Return from interrupt subroutine
000A	20	EORZ,R0	Clear R0 (counter)
000B	F0	WRTD,R0	Write R0 to the lights (non-extended)
000C	74,20	CPSU H'20'	Clear II (open interrupt window)
000E	76,20	PPSU H'20'	Set II (close interrupt window)
0010	1F,00,0B	BCTA,UN H'000B'	Branch back to WRTD

Example 2 - Indirect Interrupt

This example performs the same function as above but uses indirect interrupts. The interrupt subroutine starts at address H'100'. This address is contained in program locations H'07' and H'08'.

Address	Data	Instruction Mnemonic	Comments
0000	76,20	PPSU H'20'	Set II - Inhibit Interrupts.
0002	75,08	CPSL H'08'	Operations without carry.
0004	1F,00, 09	BCTA,UN H'0009'	Branch over interrupt address.
0007	01,00	ACON H'0100'	Interrupt routine address.
0009	20	EORZ,R0	Clear counter.
000A	F0	WRTD,R0	Write R0 to the lights.
000B	74,20	CPSU H'20'	Clear II - enable interrupts.
000D	1F,00, 0D	BCTA,UN H'000D'	Loop forever.
0100	84,01	ADDI,R0 H'01'	Increment counter.
0102	F0	WRTD,R0	Write R0 to the lights.
0103	37	RETE,UN	Return and enable interrupts.

Using the I/O Switches and Lights

The 2650 provides several methods for monitoring the status of and controlling the operation of external I/O devices. One such method unique to the 2650 is a serial I/O port formed by the SENSE input pin and the FLAG output pin on the processor. The 2650 also has provisions for two types of parallel I/O instructions, called extended and non-extended. The non-extended I/O instructions are one-byte instructions that allow a user program to read from and write to two eight-bit I/O ports: port C and port D. The two byte extended I/O instructions expand the 2650's I/O capabilities to 256 bidirectional I/O ports.

In addition to the 2650 instructions specifically intended for I/O operations, you may choose to use the memory mapped I/O mode. This mode is implemented by assigning a memory address to an I/O device. While a memory mapped I/O port requires more decode logic than either an extended or a non-extended port, it can be accessed by the full range of 2650 memory referencing instructions. (Refer to the 2650 Microprocessor Manual for a description of the 2650 I/O control modes.)

The INSTRUCTOR 50 includes features that demonstrate all of the 2650's I/O modes. These features are described as follows.

FLAG and SENSE I/O

The 2650's FLAG and SENSE pins are associated with the flag and sense bits of the processor's internal Program Status Word (PSW). The sense bit of the PSW always reflects the signal level on the SENSE pin. Likewise, the level on the FLAG pin always reflects the current value of the flag bit in the PSW.

The user may manually control the value of the sense bit in the PSW using the [SENS] key on the function control keyboard. When the [SENS] key is depressed, the sense bit is a one. Otherwise, the SENSE bit is a zero.

The INSTRUCTOR 50's FLAG indicator on the front panel is driven by the FLAG pin on the 2650, providing a visual indication of the flag bit's current value. The FLAG light is on if the flag bit is a one, and the light is off if the flag bit is a zero.

Non-Extended I/O

The 2650 can control two bidirectional I/O ports with four single-byte instructions: WRTC, WRTD, REDC and REDD. These instructions move data between port C, port D and the 2650's internal registers.

The INSTRUCTOR 50's front panel parallel I/O port can be assigned as non-extended port D by placing the Port Address select switch in the NON-EXTENDED position. In this position, the I/O port can be accessed with the WRTD and REDD instructions. This allows you to manually enter data with the input switches by including a REDD instruction in your program. Similarly, your program can write a data value to the output LEDs by executing a WRTD instruction.

Extended I/O

The 2650 can control up to 256 bidirectional I/O ports with the double-byte instructions WRTE and REDE. The second byte of these instructions specifies the extended I/O port address. The INSTRUCTOR 50's parallel I/O port can be assigned as extended address H'07' by placing the Port Address switch in the EXTENDED position. In this mode, the parallel I/O port can accessed with WRTE and REDE instructions that specify an extended address H'07' in their second byte.

Memory Mapped I/O

Memory mapped I/O is simply a matter of decoding a memory address for enabling an I/O port. To demonstrate this I/O mode, the INSTRUCTOR 50's Port Address select switch can be placed in the MEMORY position. This assigns the parallel I/O port a memory address of H'0FFF'. Thus, any memory reference instruction that specifies H'0FFF' as the source or destination will access the front panel parallel I/O port. When an instruction reads location H'0FFF', the value set in the port input switches is loaded into the specified register. If an instruction writes to memory location H'0FFF', the value contained in the specified register will appear in the port output LEDs.

CALLING MONITOR SUBROUTINES

Now that you are familiar with the 2650 instruction set and have successfully entered a few simple programs, you are undoubtedly ready and anxious to make use of some of the more powerful features provided by the INSTRUCTOR 50. For example, you might want to write a decimal add program using the INSTRUCTOR 50's keyboard and eight-digit display. By calling subroutines within the monitor program, the display can be used to request the two numbers to be added, and the hex keyboard can be used to enter the numbers. After the two numbers have been entered, and their sum calculated, another monitor subroutine can be called to display the results of the addition. This section describes these subroutines and provides examples in their use.

In addition to subroutines that provide easy access to the INSTRUCTOR 50's keyboard and display, the monitor program contains other subroutines that are useful in writing application programs. Refer to the program listing in the appendices for additional information on other subroutines.

The monitor subroutines are called with Zero Branch to Subroutine Relative (ZBSR) instructions. The ZBSR instruction specifies a subroutine relative to byte zero, page zero. The relative addressing range is -64 to +63. Since the 2650 uses an 8K page addressing scheme, ZBSR instructions with a negative offset (relative address) wraps back around to the top of the first 8K page. The top of the first 8K page in the INSTRUCTOR 50 is located within the monitor program and contains a table of indirect subroutine addresses. Thus, the monitor subroutines can be called by ZBSR instructions that specify indirect addressing and have the negative offset that points to the desired subroutine. The addresses required to call the various monitor subroutines are included in the description of each subroutine.

The subroutine descriptions include a list of the 2650 registers used in their execution. Unless otherwise specified, the contents of these registers will contain meaningless data when the subroutine returns control to the user program. Therefore, registers that contain important user program information must be stored in a memory location before the monitor subroutine is called.

When calling monitor subroutines, caution must be exercised to avoid overflowing the 2650's internal 8-level subroutine stack. Since some of the user-accessible subroutines call other subroutines within the monitor program, each subroutine description includes the number of other subroutines called during its execution. This information allows you to calculate the number of subroutine stack levels required by your program and insures that this number never exceeds eight.

MOVE SUBROUTINE

Calling Instruction:

Mnemonic	Hex Value
ZBSR *MOV	BB,FE

Registers Used:

R1 = Message Pointer -1 (high-order byte)
R2 = Message Pointer -1 (low-order byte)

Subroutine Levels Used: 0

Function:

MOVE fetches an eight-byte message within the user's program and stores the eight bytes in the monitor's display buffer. When combined with the DISPLAY subroutine, MOVE allows you to write messages on the INSTRUCTOR 50's eight-digit display. Any of the INSTRUCTOR 50's characters can be used in assembling a message.

Operation:

Before calling MOVE, you must store an eight-byte message within your program. The location of the sequential message bytes is transferred to MOVE by storing the address of the first message byte in R1 and R2 prior to calling the subroutine. Because of the algorithm used to implement the MOVE subroutine, it is necessary to subtract one from the message pointer before it is stored in R1 and R2. Following is an example of the MOVE subroutine call and a list of the hexadecimal values for the INSTRUCTOR 50's display characters.

EXAMPLE OF MOVE SUBROUTINE CALL

Address	Data	Instruction Mnemonic	Comments
•			
•			
•			
0010	05,00	LODI, R1 H'00'	Load message pointer -1 in R1 and R2
0012	06,FF	LODI, R2 H'FF'	(H'100' -1 = H'00FF').
0014	BB,FE	ZBSR *MOV	Call MOVE. The message bytes stored in locations 0100-0107 are transferred to the monitor's display buffer.
•			
•			
•			
0100	17		= blank **(first byte of message)**
0101	14		= H
0102	0E		= E
0103	11		= L
0104	11		= L
0105	00		= O
0106	17		= blank
0107	17		= blank (last byte of message)

Hex Values of Display Characters

Character	Value	Character	Value	Character	Value
*0.0	H'00'	*A	H'0A'	*H	H'14'
*1.I	H'01'	*B	H'0B'	*0	H'15'
*2	H'02'	*C	H'0C'	*=	H'16'
*3	H'03'	*D	H'0D'	*BLANK	H'17'
*4	H'04'	*E	H'0E'	*J	H'18'
*5.5	H'05'	*F	H'0F'	*-	H'19'
*6.G	H'06'	*P	H'10'	*	H'1A'
*7	H'07'	*L	H'11'	*Y	H'1B'
*8	H'08'	*U	H'12'	*N	H'1C'
*9	H'09'	*R	H'13'		

DISPLAY SUBROUTINE

Calling Instruction:

Mnemonic	Hex Value
ZBSR *DISPLY	BB,EC

Registers Used:

R0,R1,R2,R3

On entry R0 = Display Command
On exit R0 = Key Value (optional)

Subroutine Levels Used: 0

Function:

When used with the MOVE subroutine, DISPLAY writes messages on the INSTRUCTOR 50's eight-digit display. DISPLAY reads the message stored in the monitor's display buffer with MOVE and writes the message on the display. Optionally, DISPLAY can be used to read the function and data keyboards and return the value of a depressed key.

Operation:

DISPLAY has three modes of operation that are selected by writing a command byte in R0 prior to calling the subroutine. The DISPLAY commands and the functions they specify are summarized below:

Value Placed in RØ	Function
H'00'	Displays message in display buffer until a function or data key is depressed. Returns the value of the depressed key in R0.
H'01'	Makes one pass through the DISPLAY subroutine and does not read the keyboards. A single pass through the DISPLAY subroutine will not produce a visible display. Hence, when this command is used, it should be part of a loop that calls DISPLAY a sufficient number of times to illuminate the message.
H'80'	This command is identical to the H'00' command except that the decimal point of the most-significant (far-left) digit is illuminated.

The function and data key values returned in R0 when operating in response to commands H'00' and H'80' are listed in the following table. This is followed by an example of the MOVE and DISPLAY subroutine calls that displays the message HELLO until the [RUN] key is depressed.

Data Values for Command and Data Keys

Key	Value	Key	Value	Key	Value
Ø	H'00'	8	H'08'	WCAS	H'80'
1	H'01'	9	H'09'	BKP	H'81'
2	H'02'	A	H'0A'	RCAS	H'82'
3	H'03'	B	H'0B'	REG	H'83'
4	H'04'	C	H'0C'	STEP	H'84'
5	H'05'	D	H'0D'	MEM	H'85'
6	H'06'	E	H'0E'	RUN	H'86'
7	H'07'	F	H'0F'	ENT/NXT	H'87'

Example of Move and Display Subroutine Calls

Address	Data	Instruction Mnemonic	Comment
0010	05,00	LODI,R1 H'00'	Place message table pointer minus one in R1 and R2.
0012	06,FF	LODI,R2 H'FF'	
0014	BB,FE	ZBSR *MOV	Call the Move subroutine.
0016	04,00	LODI,R0 H'00'	Place command byte in R0.
0018	BB,EC	ZBSR *DISPLY	Call the DISPLAY subroutine.
001A	E4,86	COMI,R0 H'86'	Compare returned key code to RUN code. If equal,
001C	1C,XX,XX	BCTA,EQ H'XXXX'	branch to address H'XXXX'.
001F	1F,00,16	BCTA,UN H'0016'	If not equal, loop back and wait for next key.
⋮			
0100	17		First byte of message table = blank
0101	14		= H
0102	0E		= E
0103	11		= L
0104	11		= L
0105	00		= O
0106	17		= blank
0107	17		Last byte of message table = blank

USER DISPLAY SUBROUTINE

Calling Instruction:

Mnemonic	Hex Value
ZBSR *USRDSP	BB,E6

Registers Used:

R0, R1, R2, R3

On entry R3 = Display Command
R1 = Message Pointer -1 (high order)
R2 = Message Pointer -1 (low order)
On exit R0 = Key value (optional)

Subroutine Levels Used: 2

Function:

USER DISPLAY combines the functions of MOVE and DISPLAY. That is, USER DISPLAY moves an eight-byte message from a user program to the display buffer and then displays the message. As with DISPLAY, this subroutine may be used to read the function and data keyboards.

Operation:

Before calling USER DISPLAY, you must load the first address of your message table (-1) in R1 and R2. Additionally, R3 must be loaded with the desired command as in the DISPLAY subroutine.

The following example of a USER DISPLAY subroutine call displays the message HELLO until the RUN key is depressed. (This example is functionally equivalent to the example for the DISPLAY subroutine).

Example of a USER DISPLAY Subroutine Call

Address	Data	Instruction Mnemonic	Comment
0010	05,00	LODI,R1 H'00'	Place message table pointer -1 in R1 and
0012	06,FF	LODI,R2 H'FF'	R2.
0014	07,00	LODI,R3 H'00'	Place command byte in R3.
0016	BB,E6	ZBSR *USRDSP	Call USER DISPLAY.
0018	E4,86	COMI,R0 H'86'	Compare returned key value to RUN's value.
001A	1C,XX,XX	BCTA,EQ H'XXXX'	Branch to XX,XX if equal.
001D	1F,00,10	BCTA,UN H'0010'	If not equal, loop back and get another key.
.	.		
.	.		
.	.		
0100	17		First byte of message table = blank
0101	14		= H
0102	0E		= E
0103	11		= L
0104	11		= L
0105	00		= O
0106	17		= blank
0107	17		Last byte of message table = blank

NIBBLE SUBROUTINE

Calling Instruction:

Mnemonic	Hex Value
ZBSR *DISLSD	BB,F4

Registers Used:

R0 and R2
On entry:. R0 = byte (high-order nibble, low-order nibble)
on exit: R0 = high-order nibble
R1 = low-order nibble

Subroutine Levels Used: 1

Function:

NIBBLE takes an eight-bit byte and separates it into two bytes, each containing one of the original four-bit nibbles. This subroutine is useful in user programs that display a register or memory data value on the INSTRUCTOR 50 display.

Operation:

The byte to be separated is passed to NIBBLE in R0. NIBBLE then takes the least-significant four bits (low-order nibble) from R0 and places them in the four least-significant bits of R1. When NIBBLE returns program control to your program, R0 contains the low-order nibble, and R1 contains the high-order nibble. The most-significant four bits of both R0 and R1 contain zeros. A functional example of NIBBLE is shown below. This is followed by an example of a NIBBLE subroutine call.

Functional Example of NIBBLE

On entry: R0 = F3

On exit: R0 = 0F
R1 = 03

Example of NIBBLE Subroutine Call

Address	Data	Instruction Mnemonic	Comment
0000	70	REDD,R0	Read I/O port (Non-Extended) into R0.
0001	BB,F4	ZBSR *DISLSD	Call NIBBLE subroutine.
0003	CD,01,07	STRA,R1 H'01,07'	Store low-order nibble in message table.
0006	CC,01,06	STRA,R0 H'01,06'	Store high-order nibble in message table.
0009	05,00	LODI,R1 H'00'	Place message table pointer (-1) in R1 and R2.
000B	06,FF	LODI,R2 H'FF'	Place display command in R0.
000D	04,00	LODI,R0 H'00'	
000F	BB,E6	ZBSR*USRDSP	Call USER DISPLAY subroutine. Displays previous Port D value. Allows new I/O value to be set up in switches. Exits when any key is depressed.
0011	1B,6D	BCTR,UN H'6D'	Loop back to 0000 and get new I/O value.
:			
0100	13		= "R" (first byte of message table).
0101	0E		= "E"
0102	0D		= "D"
0103	0D		= "D"
0104	16		= "="
0105	17		= "blank"
0106	17		= "blank" (high-order nibble will be stored here).
0107	17		= "blank" (low-order nibble will be stored here).

INPUT DATA SUBROUTINE

Calling Instruction:

Mnemonic	Hex Value
ZBSR *GNP	BB,FA

Registers Used:

On entry: R0 = Input Command

On exit: R0 = Two Data Key Values
R1 = Two Data Key Values (optional)
R2 = Function Key Value
R3 = Data Entered Indicator

Subroutine Levels Used: 1

Function:

INPUT DATA displays the contents of the display buffer and scans the data keyboard for data entry. As data is keyed in, the subroutine writes the input data in the least-significant digits of the display. When a function key is depressed, USER DISPLAY returns to the main program with the input data and function key values in the 2650's internal registers.

Operation:

INPUT DATA has two selectable modes of operation. Mode selection is made by writing an input command byte in R0 before calling the subroutine. The input command bytes and the functions they specify are listed as follows:

Value Placed in R0	Function
H'00'	Displays a four-digit message and accepts four digits of data. As each data value is keyed in, it is displayed in the least-significant (right-most) display digit, and previously entered values are shifted left. Data entry is terminated and program control is returned to the user program when a function key is depressed. If less than four data values are entered, zeros are inserted in the non-entered digit positions.
H'01'	Identical to H'00' except that a five-digit message is displayed, and two digits of data are input from the data keyboard.

The four or five-digit message to be displayed by INPUT DATA must be placed in the monitor's display buffer before INPUT DATA is called. The message characters displayed are taken from the first four or five bytes of the eight-byte message table transferred to the display buffer by the MOVE subroutine.

The data values input to INPUT DATA are returned to the main program in R0 for the two-digit input mode and to R0 and R1 for the four-digit input mode. In the two-digit input mode, the most-significant data value entered is returned in the most-significant nibble of R0, and the least-significant data value is returned in the least-significant nibble of R0. In the four-digit input mode, the two most-significant data values are returned in R1, and the two least-significant data values are returned in R0.

When data entry is terminated with a function key depression, the value of the function key is returned to R2, and a data entered indicator value is returned to R3. If no data has been entered before a function key is depressed, R3 will contain the value H'7F'. If data has been entered, R3 will contain a value of H'00'. This allows you to insure that the data returned in R0 and R1 is valid data. The following example illustrates how data is returned to the user program. (This is followed by an example of an INPUT DATA call.)

Example of Data Entry and Register Contents on Return From Input Data Subroutine

Key	Display	Comments
[1][2][3][4]	P L U S	Initial display on subroutine entry.
[RUN]	P L U S 1 2 3 4	Data values entered.
		Data entry terminated and program control returned to user program.

Register Contents on Return from Input Data Subroutine

Register	Contents	Comments
R0	H'34'	Least-significant data values
R1	H'12'	Most-significant data values
R2	H'86'	Value of RUN key
R3	H'00'	Indicates valid data in R0 and R1

Example of Input Data Subroutine Call

Address	Data	Instruction Mnemonic	Comment
0050	05,00	LODI,R1 H'00'	Place message table
0052	06,FF	LODI,R2 H'FF'	pointer (-1) in R1 and R2.
0054	BB,FE	ZBSR *MOV	Call MOVE to transfer message table to display buffer.
0056	04,00	LODI,R0 H'00'	Place input command in R0 (H'00' = 4 digits).
0058	BB,FA	ZBSR *GNP	Call INPUT DATA subroutine.
⋮			
0100	10		First byte of message table = P
0101	11		= L
0102	12		= U
0103	05		= S
0104	17		= blank
0105	17		= blank
0106	17		= blank
0107	17		Last byte of message table = blank

NOTE: Since the input command requests four digits of input data, only the first four message table bytes (0100 - 0103) are displayed.

MODIFY DATA SUBROUTINE

Calling Instruction:

Mnemonic	Hex Value
ZBSR *GNPA	BB,FC

Registers Used:

R0,R1,R2,R3

On entry: R0 = Input command

On exit: R0 = Two Data Key Values
R1 = Two Data Key Values
R2 = Function Key Value
R3 = Data Entered Indicator

Subroutine Levels Used: 1

Function:

MODIFY DATA is very similar to INPUT DATA. The major difference is that the initial display message can use all eight digit positions on the INSTRUCTOR 50 display panel. MODIFY DATA enables a program to display data values that were previously entered with INPUT DATA and allows these data values to be modified.

Operation:

As with INPUT DATA, MODIFY DATA has two modes of operation that are selected by writing an input command byte in R0 prior to calling the subroutine. The input commands and their respective functions are listed below:

Value Placed in R0	Resulting Function
H'00'	Displays an eight-digit message and accepts four digits of data. After the first data key is depressed, the four least-significant digits of the display are cleared. Each new data value entered is then displayed in the least-significant display digit, and previously entered values are shifted left. Data entry is terminated when a function key is depressed.
H'01'	Identical to H'00' except that when the first data key is depressed, the three least-significant display digits are cleared, and two digits of data may be entered.

The eight-digit message to be displayed must be transferred to the monitor's display buffer with MOVE before MODIFY DATA is called. The values for the data entered indicator are the same for MODIFY DATA as for INPUT DATA. That is, R3 contains H'00' if R0 and R1 contains valid data and H'7F' if a function key was depressed before data was entered. The following example illustrates operation of MODIFY DATA. This is followed by an example of a MODIFY DATA subroutine call.

Data Entry and Register Contents on Return

From MODIFY DATA

Data Input Key	Display	Comments
	J O B = 0 1	Initial display on subroutine entry.
[2]	J O B = 2	Least-significant three digits are cleared and new data is displayed.
[R U N]		Data entry is terminated, and program control is returned to user program.

Register Contents on Return from MODIFY DATA

Register	Contents	Comments
R0	H'02'	Data entered is returned in R0.
R1	H'XX'	Data in R1 is meaningless.
R2	H'86'	Value of RUN key.
R3	H'00'	Indicates valid data in R0.

Example of MODIFY DATA Subroutine Call

Address	Data	Instruction Mnemonic	Comment
0034	05,00	LODI,R1 H'00'	Place message table pointer (-1) in R1 and R2.
0036	06,FF	LODI,R2 H'FF'	
0038	BB,FE	ZBSR *MOV	Call MOVE to transfer the message table to the display buffer.
003A	04,01	LODI,R0 H'01'	Place input command in R0 (H'01' = 2 digits).
003C	BB,FC	ZBSR *GNPA	Call MODIFY DATA subroutine.
⋮			
0100	17		First byte of message table = "blank"
0101	18		= "J"
0102	15		= "O"
0103	0B		= "B"
0104	16		= "="
0105	17		= "blank"
0106	00		= "0" previously entered data value
0107	01		= "1" previously entered data value

Jumper Options

The INSTRUCTOR 50's versatility is enhanced by jumper options on the printed circuit board. These options allow you to modify the system's basic configuration. The jumpers are accessible through cut-outs at the bottom of the INSTRUCTOR 50's plastic housing. Figure 5.1 identifies the location of the various jumpers and their configuration. The factory supplied configurations are identified by asterisks (*) in the jumper pin description tables.

Jumper A - Interrupt Selection

As described previously, a switch at the bottom of the INSTRUCTOR 50 allows you to select interrupts from the interrupt key on the function keyboard or from the input line frequency clock. Jumper 'A' provides additional interrupt flexibility by allowing interrupt requests from external logic via the bus interface connector. If this option is exercised, interrupt requests from external logic will result in a vectored interrupt through memory address H'0007'. The setting of the DIRECT/INDIRECT switch on the front panel determines whether an externally generated interrupt request results in a direct or indirect subroutine branch. The pin descriptions for jumper 'A' are defined in the following table:

JUMPER A Pin Descriptions

Pins Connected	Description
1-2*	Normal operation. The 2650 recognizes interrupt requests from the interrupt key or the real-time clock, depending on the position of S6.
2-4	Bus interface. The 2650 recognizes interrupt requests from the interface signal VIO (pin 4). The interrupt latch is set on the rising edge of VIO.
2-39 3-4	Bus interface inverted. This configuration is identical to the 2-4 option except that the interrupt latch is set on the falling edge of VIO.

Jumper B - S100 Clock Select

The bus interface includes three pins for S100 interface clock requirements. The jumper 'B' option allows you to select between two clock signals generated by the INSTRUCTOR 50. The first clock is the same 895 KHz clock available to the 2650.

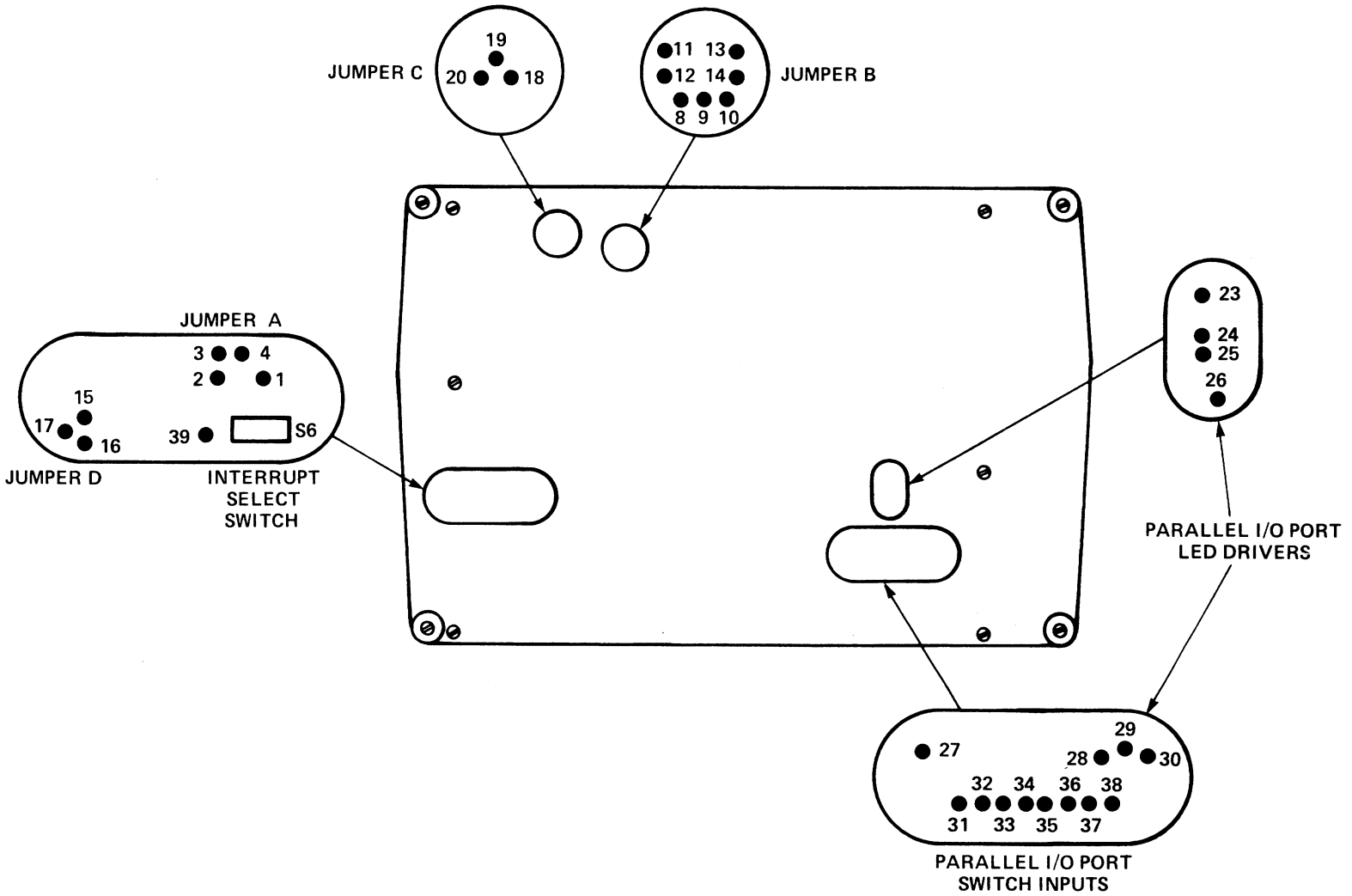


Figure 5.1: Jumper Locations

The second clock is the OPREQ signal generated by the 2650 gated by the forced jump logic enable (i.e., the OPREQ clock is inhibited whenever the forced jump logic has control of the 2650's address and data busses). The pin descriptions for jumper 'B' are defined in the following table:

JUMPER B Pin Definitions

Clock Source Pins

Pin Numbers	Description
11,12	These pins are driven by the INSTRUCTOR 50 895 KHz system clock.
13,14	These pins are driven by the conditioned OPREQ signal. The frequency is approximately 303 KHz. (NOTE: This clock is not a continuous frequency. Some 2650 instructions are executed without generating OPREQ).

S100 Clock Pins

Pin Numbers	Description
8	This pin is connected to the S100 bus signal Ø1, pin 25.
9	This pin is connected to the S100 bus signal Ø2, pin 24.
10	This pin is connected to the S100 bus signal CLOCK, pin 49.

Jumper C - Power Source Select

The INSTRUCTOR 50 is designed to operate with its own internal power supply used in conjunction with the wall transformer supplied with the system. Optionally, the input to the INSTRUCTOR 50's 5-volt regulator can be supplied from the interface bus connector. Jumper 'C' supports this option. The pin descriptions for jumper 'C' are defined in the following table:

JUMPER 'C' Pin Definitions

Pin Connected	Description
18-20*	Normal operation. The INSTRUCTOR 50 power requirements are supplied by the wall transformer.
18-19	The INSTRUCTOR 50 power requirements are supplied by an 8-volt unregulated D-C source applied via the bus interface connector.

Jumper D - Cassette Output Selection

The INSTRUCTOR 50's cassette interface provides two recording signal levels. Jumper 'D' selects between a 30mV rms record level and a 300mV rms record level. The pin descriptions for jumper 'D' are defined in the following table:

JUMPER 'D' Pin Definitions

Pins Connected	Description
15-17*	This option provides a 30mV rms record level to the cassette.
16-17	This option provides a 300mV rms record level to the cassette.

6. SYSTEM EXPANSION

Introduction

Microprocessors have had a tremendous impact on the hobbyist computer market. Beginning with Altair's 8800 home computer, the hobbyist market has literally exploded with new products. These new products include not only basic computers but a host of small support systems or peripheral boards. The first peripheral boards were simple memory expansion boards, but today there are a wide variety of peripherals available. There are television interfaces for computer graphics, floppy disc interfaces for mass storage, and even a board that synthesizes human speech.

The majority of these peripheral boards are designed to be compatible with the Altair 8800 bus. As more and more Altair 8800-compatible systems were introduced, this microcomputer bus was given an industry wide name, the S100 bus.

The INSTRUCTOR 50's S100 interface (an edge connector at the back of the unit) transforms a simple learning device into a small system computer limited only by the number and type of peripheral boards used. Moreover, the powerful program/data entry and debug facilities of the basic INSTRUCTOR 50 are extended to any device connected to the S100 bus interface.

Because the Altair 8800 home computer was based on the 8080, many of the S100 bus signals are essentially 8080 signals. Many of these signals, such as the two-phase clock and negative supply voltage, are not required by state-of-the-art microprocessors like the 2650. Hence, the INSTRUCTOR 50's S100 interface bus is not pin-for-pin compatible with Altair's original bus. However, the INSTRUCTOR 50's interface bus contains the most commonly used signals and can be easily connected to the majority of S100 peripherals. In addition to the common S100 bus signals, spare pins on the S100 pin bus have been assigned 2650 signals (e.g., OPREQ R/W and M/IO). Thus, custom interfaces can be designed with the 2650 control logic, instead of the more cumbersome 8080 interface logic. In short, the INSTRUCTOR 50's interface bus opens up the entire universe of home computer peripherals to owners of the INSTRUCTOR 50 training system.

The INSTRUCTOR 50 bus interface signals are described in Table 6.1.

TABLE 6.1

INSTRUCTOR 50 INTERFACE BUS SIGNALS

(* indicates a 2650 bus signal)

Pin #	Mnemonic	Signal Description
1	+8V	Positive 8 volts, unregulated. This line provides +8 volts to the INSTRUCTOR 50 when Jumper C selects the interface bus as the system power source.
2	+16V	Positive 16 volts. This line is reserved for +16 volts that may be required for a S100 peripheral board. +16V is neither required for or generated by the INSTRUCTOR 50.
3	XRDY	External Ready. XRDY is returned by an external device when it has completed a data transfer with the 2650. On board the INSTRUCTOR 50 XRDY becomes OPACK for the 2650.
4	VIO	Vectored Interrupt #0. VIO provides an external interrupt request when Jumper A is wired for external interrupts. VIO is latched and generates either an indirect or direct interrupt (selected by the DIRECT/INDIRECT switch) through address H'0007'.
5	not used	
6	not used	
7	not used	
8	not used	
9	not used	
10	not used	
11	not used	

12	R/W*	Read/Write. A 2650 control signal that indicates whether the processor is performing a read or write operation with an external peripheral board. As with all of the 2650 control signals, R/W is valid only when OPREQ is true. * NOTE: Indicates non-S100 2650 control signals.
13	WRP*	Write Pulse. A 2650 control signal that is generated during memory or I/O write sequences. WRP may be used to strobe data into the selected device.
14	M/IO*	Memory/Input-Output. A 2650 signal that indicates whether the address bus contains a memory or I/O address during a data transfer operation.
15	RESET*	Reset. When driven high, RESET performs the same operation as depressing the RST switch on the INSTRUCTOR 50 front panel. That is, the 2650 is reset and begins executing the user program at location H'0000'.
16	RUN/WAIT*	Run/Wait. A 2650 control signal that indicates whether the 2650 is in the wait state or is executing a program.
17	PAUSE*	Pause. This 2650 control signal input is provided for Direct Memory Access (DMA) operations. When driven high, this signal causes the 2650 to enter the WAIT state after completing the instruction currently being executed.
18	not used	
19	not used	
20	not used	
21	not used	
22	not used	
23	not used	
24	Ø1	Phase 1 Clock. Ø1 may be driven by the 895 KHz system clock or the 2650 OPREQ signal depending on the configuration of the Jumper B option.

25	Ø2	Phase 2 Clock. Ø2 may be driven by the system clock or OPREQ depending on the configuration of Jumper B.
26	not used	
27	not used	
28	not used	
29	A5	Address Bit 5
30	A4	Address Bit 4
31	A3	Address Bit 3
32	A15	Address Bit 15. Since the 2650 has an address range of 32K, this line is grounded.
33	A12	Address Bit 12
34	A9	Address Bit 9
35	D01	Data Out Bit 1
36	D00	Data Out Bit 0
37	A10	Address Bit 10
38	D04	Data Out Bit 4
39	D05	Data Out Bit 5
40	D06	Data Out Bit 6
41	D12	Data In Bit 2
42	D13	Data In Bit 3
43	D17	Data In Bit 7
44	not used	
45	SOUT	Output. SOUT indicates that the address bus contains the address of an output I/O device. The addressed device may accept the value on the data bus when PWR (pin 77) is active.
46	SINP	Input. SINP indicates that the address bus contains the address of an input I/O device. The selected device should return its data when PDBIN (pin 78) is active.

47	SMEMR	Memory Read. This signal indicates that the address bus contains the address of a memory location and that the 2650 is performing a memory read operation.
48	not used	
49	CLOCK	System Clock. Depending on the configuration of Jumper B, this line is driven by the 895 KHz system clock or the 2650 OPREQ output.
50	GND	System Ground.
51	+8V	Positive 8 volts. This line provides +8V to the INSTRUCTOR 50 when Jumper C selects the interface bus as the system power source.
52	-16V	Negative 16 volts. This line is reserved for -16 volts that may be required by a S100 peripheral board. Not supplied by INSTRUCTOR 50.
53	not used	
54	not used	
55	DØ*	Data Bus Bit Ø -
56	D1*	Data Bus Bit 1 -
57	D2*	Data Bus Bit 2 -
58	D3*	Data Bus Bit 3 -
59	D4*	Data Bus Bit 4 -
60	not used	
61	D5*	Data Bus Bit 5 -
62	D6*	Data Bus Bit 6 -
63	D7*	Data Bus Bit 7 -

In addition to the 2650 control signals, the INSTRUCTOR 50 Interface bus also includes a bidirectional data bus. The 2650 signals form a subset of the Interface Bus that can be used to interface the INSTRUCTOR 50 to breadboard peripherals with a minimum of interconnect wires.

64	UOPREQ*	User Operation Request - OPREQ is a 2650 control signal that indicates that the processor's address bus, data bus, and other control signals are valid. OPREQ may be used to latch the data bus for write operations and enable input device bus drivers for read operations.

65	INTACK*	INTERRUPT ACKNOWLEDGE. The 2650 returns INTACK to an interrupting device in response to an INTERRUPT REQUEST. Upon receipt of INTACK, the interrupting device drives the data bus with a relative branch address and asserts either XRDY or PRDY (these signals become the 2650 status signal OPACK).
66	FLAG*	FLAG. This line contains the 2650 single bit output port.
67	USENSE*	USER SENSE. USENSE is the 2650 single bit input port. FLAG and SENSE are part of the PROGRAM STATUS WORD.
68	MWRITE	MEMORY WRITE. MWRITE indicates that data is to be written into the memory location addressed by the current value of the ADDRESS BUS.
69	not used	
70	not used	
71	not used	
72	PRDY	PROCESSOR READY. PRDY is logically OR'd with XRDY to form the 2650 status signal OPACK. PRDY is returned by an addressed device (either memory or I/O) or an interrupting device when the requested data transfer has been completed.
73	PINT	PROCESSOR INTERRUPT. PINT is an S100 signal that corresponds to the 2650 INTERRUPT REQUEST signal. The 2650 acknowledges PINT when it completes the instruction it was executing when PINT was driven low. The 2650 does not recognize PINT if it is in the WAIT state or if the INTERRUPT INHIBIT bit of the PSW is reset. PINT can be used to release the 2650 from the HALT state.
74	not used	
75	not used	
76	not used	
77	PWR	PROCESSOR WRITE. PWR indicates that the data bus is valid and may be accepted by the addressed memory location or output device.
78	PDBIN	PROCESSOR DATA BUS IN. PDBIN indicates that the 2650 is reading data from the addressed memory location or input device. PDBIN may be used to enable the selected device's data bus drivers.

79	AØ	Address Bit Ø
80	A1	Address Bit 1
81	A2	Address Bit 2
82	A6	Address Bit 6
83	A7	Address Bit 7
84	A8	Address Bit 8
85	A13	Address Bit 13
86	A14	Address Bit 14
87	A11	Address Bit 11
88	D02	Data Out Bit 2
89	D03	Data Out Bit 3
90	D07	Data Out Bit 7
91	D14	Data In Bit 4
92	D15	Data In Bit 5
93	D16	Data In Bit 6
94	D11	Data In Bit 1
95	D10	Data In Bit Ø
96	not used	
97	not used	
98	not used	
99	POR	POWER ON RESET. POR is an output signal that indicates that power has been applied to the INSTRUCTOR 50 and the system is being reset. POR may be used to reset peripheral boards on the Interface Bus
100	GND	GROUND. System Ground.

7. THEORY OF OPERATION

Introduction

The INSTRUCTOR 50 is typical of modern microcomputers, reflecting many of the recent advances in microprocessor technology. For example, the current trend in microcomputer design is to replace logic functions implemented with SSI and MSI circuits with complex LSI microprocessor support circuits. This trend is exemplified in the INSTRUCTOR 50 which makes use of the 2650 microprocessor and the 2656 System Memory Interface. These two chips alone constitute a basic microcomputer. Beyond this two-chip microcomputer, the remainder of the circuits on the INSTRUCTOR 50 Printed Circuit Board are devoted to providing the microcomputer with man-machine and machine-machine interfaces.

This section describes the hardware and software associated with the INSTRUCTOR 50 system. The intent is not to give a detailed exposition for maintenance purposes. The INSTRUCTOR 50 comes fully assembled and debugged ready to be plugged in and used and requires little or no maintenance. Rather, the intent is to introduce you to the basic fundamentals of modern microcomputer design.

Basic Concept

The functional heart of computers in general and microcomputers in particular is the system program. The program is a logical sequence of machine instructions that monitor system status, and, based on that status, decides what control actions to take. A computer's Central Processing Unit (CPU) is a device that reads instructions from program storage and, by executing the instructions, performs all of the arithmetic and logical operations required by the system program. The CPU also provides the system program with the physical means to access and control the system's I/O functions. The INSTRUCTOR 50's CPU is the 2650 microprocessor.

The 2650 fetches instructions from program storage and communicates with the system I/O circuits via its address bus, control bus, and data bus. As the 2650 executes each instruction, the address and control bus values specify the device to be communicated with (memory location, I/O device, etc.), and the data bus serves as an information conduit between the processor and the selected device. This information transfer scheme defines the system's basic architecture illustrated in Figure 7.1. Considerable savings in parts count was realized by decoding the I/O device addresses within the 2656 SMI.

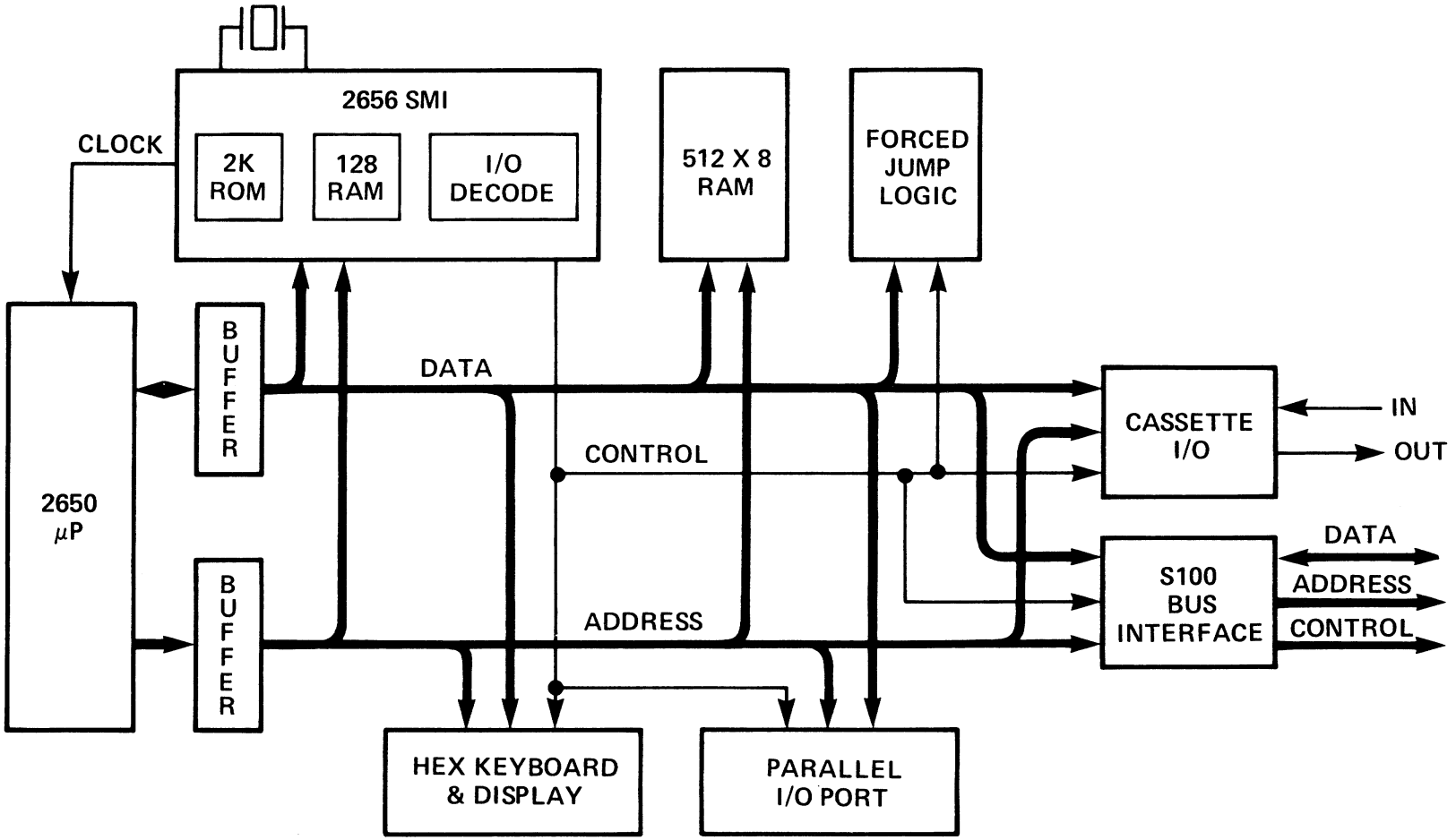


Figure 7.1: Basic Instructor 50 Architecture

Thus, when the 2650 executes an instruction that references an I/O device (e.g., the parallel I/O port), that device's address is asserted on the address bus, and the Programmable Gate Array within the SMI decodes the address and generates the I/O device's enable signal. Thus enabled, the selected I/O device either accepts data from or returns data to the 2650 over the data bus. As the 2650 executes each instruction, it selects the device specified by the instruction (program storage, user RAM, an I/O device, etc.) with the address bus and communicates with the selected device via the data bus.

Detailed Block Diagram Description

A detailed block diagram of the INSTRUCTOR 50 is presented in Figure 7.2. This section gives a description of each of the major functional blocks illustrated in Figure 7.2

The Microcomputer

As mentioned previously, the basic microcomputer consists of the 2650 microprocessor and the 2656 SMI. The 2650 provides the following functions:

8-bit ALU	The Arithmetic Logic Unit performs all of the arithmetic and logical operations required for program execution.
Program Counter	The program counter is used to generate program storage addresses.
Interrupt Logic	The interrupt logic performs all functions required to respond to an interrupt request from an external device.
Internal Registers	The 2650's seven internal registers provide temporary data storage and serve as a link between the ALU and external data storage, such as RAM locations and I/O devices.
Bus Interface Logic	The bus interface logic distinguishes between memory and I/O device addresses and specifies the direction of data transfers between the processor and external data storage.

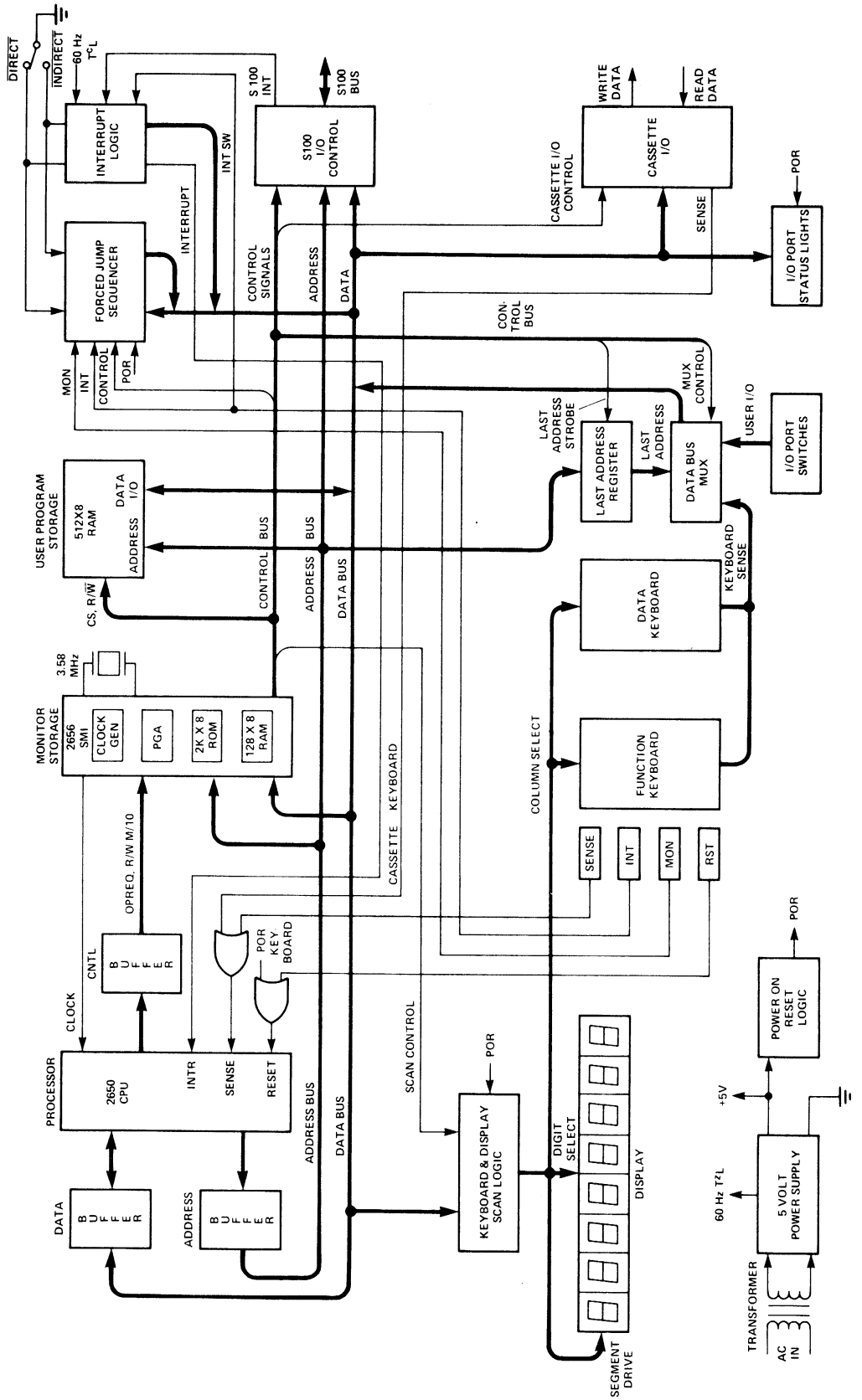


Figure 7.2: Instructor 50 Detailed Block Diagram

The 2650 microprocessor is surrounded with bus drivers (buffers). Because the 2650 is fabricated using an MOS process, its output pins can drive only one TTL load. The bus drivers buffer the 2650 outputs and are able to drive all of the loads on the INSTRUCTOR 50's busses.

The buffered 2650 data, address and control busses are connected directly to the 2650 SMI. The SMI contains the 2K monitor program, 128 bytes of scratch-pad RAM, a system clock generator, and an eight-bit I/O port. The eight-bit I/O port is controlled by a mask Programmable Gate Array (PGA). As configured for the INSTRUCTOR 50, the PGA decodes the address bus and provides eight I/O chip enables for the user RAM and I/O devices. Table 7.1 lists the functions of these outputs.

All of the monitor program's scratch-pad memory requirements are met by the SMI's 128 byte RAM. In fact, the monitor only requires 64 bytes, thus leaving the remaining 64 bytes for user storage. It should be noted, however, that while the INSTRUCTOR 50 enables the user to access these 64 bytes of the SMI's RAM with the DISPLAY AND ALTER MEMORY command and the FAST PATCH command, the SINGLE STEP and BREAKPOINT commands are not supported within this memory space. Hence, these 64 bytes should be used for data storage only. That is, user programs should be stored in user RAM or on an S100 memory expansion board.

INSTRUCTOR 50 Memory Allocation

Figure 7.3 is a memory map of the INSTRUCTOR 50's addressable memory space. The memory map is divided into four 8K pages reflecting the addressing architecture of the 2650. The first page, page zero, contains the user RAM and the SMI ROM and RAM. The second, third, and fourth pages are available for user memory expansion or memory mapped I/O.

The user RAM is formed by four 256 x 4 RAMs (Signetics 2112's) that are enabled by the SMI chip-enable lines mentioned previously. Section 6 describes how S100 memory boards can be added to the INSTRUCTOR 50.

TABLE 7.1
CONTROL SIGNALS GENERATED BY THE SMI

Signal	Function
RAM0CE	RAM 0 chip enable: this signal enables the lower 256 bytes of user RAM.
RAM1CE	RAM 1 chip enable: this signal enables the upper 256 bytes of user RAM.
PORTFX	PORTFX goes low whenever the 2650 executes an extended I/O instruction with an address between H'F8' and H'FF' inclusive. This signal enables the INSTRUCTOR 50's I/O device addresses to be decoded with just three address bits.
USRPORT	USRPORT goes low whenever the 2650 accesses the parallel I/O port with an extended I/O instruction (address H'07').
USRMEM	This signal goes low when the 2650 executes a memory reference instruction that specifies address H'0FFF' USRMEM enables the parallel I/O port when the port address select switch is in the MEMORY position.
DI/O	DI/O goes low when the 2650 executes a non-extended I/O instruction that specifies port D. If the port address select switch is in the NON-EXTENDED position, DI/O enables the parallel I/O port.
CI/O	CI/O goes low when the 2650 executes a WRTC instruction. This signal is used by the forced jump logic for breakpoint detection.
MON	MON goes low whenever the 2650 fetches an instruction or data value within the monitor's address space (H'17C0' and H'1FFF').

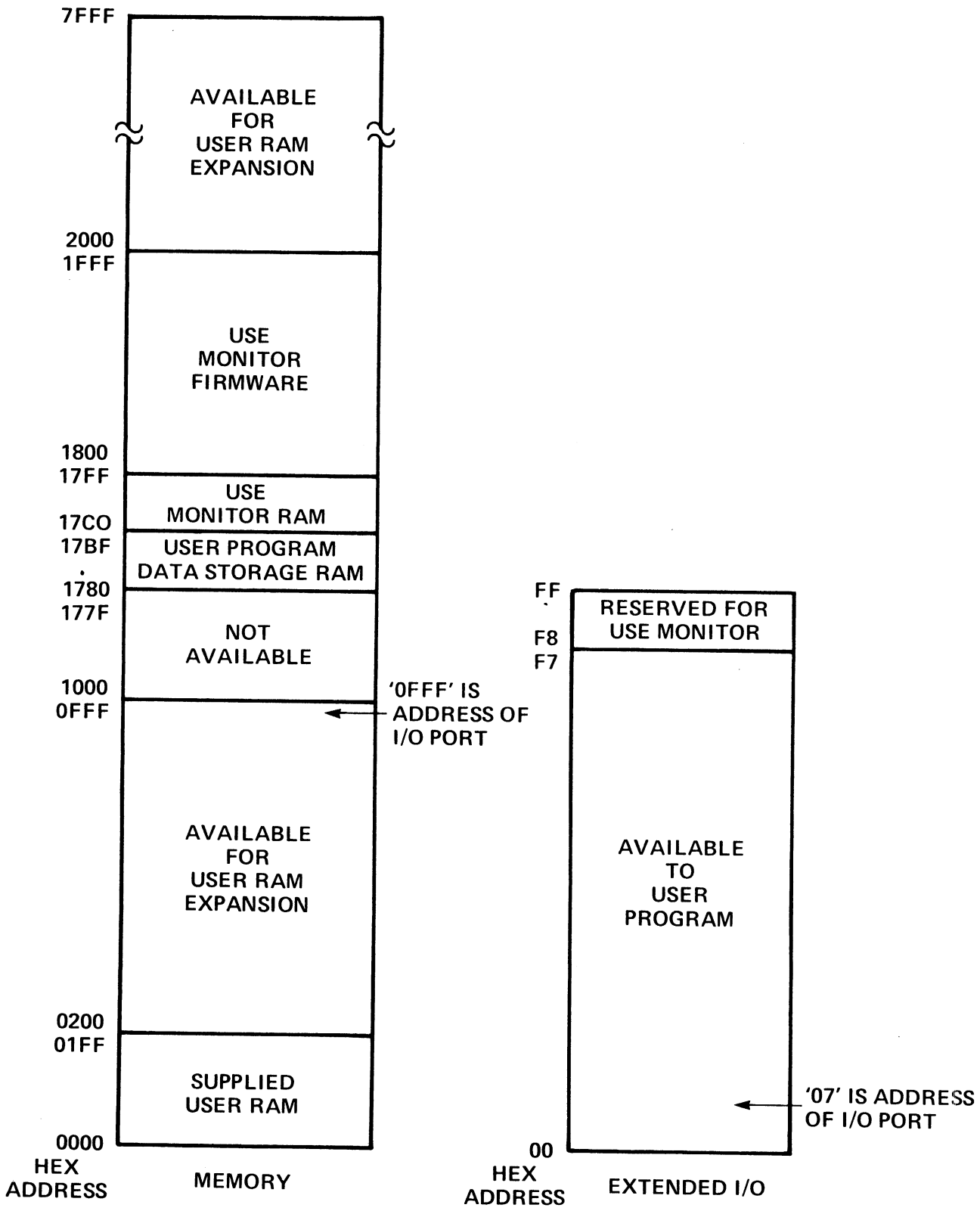


Figure 7.3: INSTRUCTOR 50 Memory Map

Parallel I/O Port

The parallel I/O port consists of an output latch, input switches, and port address decode logic. The port address decode logic generates a port enable whenever one of the three following conditions are met.

1) The 2650 executes a WRTD or REDD instruction.

2) The 2650 executes either a WRTE or REDE instruction that specifies H'07' as an extended I/O address.

3) The 2650 executes a memory reference instruction that specifies location H'0FFF'.

The Port Address switch selects one of these signals as the parallel I/O port enable.

Whenever the I/O port is enabled and the R/W control line specifies a write operation, the value on the data bus is strobed into the I/O port output latch. This latch drives the I/O port indicator LEDs.

The I/O port switches are one of four inputs to a data bus multiplexer. Whenever the I/O port is enabled and the R/W line indicates a read operation, the I/O switch levels are asserted on the data bus via the data bus multiplexer.

Keyboard and Display Logic

The INSTRUCTOR 50's primary man-machine interface consists of an output device, the eight-digit display, and two input devices — the function and data entry keyboards. Together they provide an inexpensive human interface to the microcomputer.

The display digits consist of seven discrete LEDs arranged in a rectangular array or bars and an eighth LED that serves as a decimal point. There are several methods of driving a seven-segment display with a microprocessor. The most straightforward approach is to provide a separate output port latch to drive each individual display. With this approach, the microprocessor simply writes a byte to each output port, corresponding to the segments required to form the desired character. While the direct drive approach is the simplest to conceptualize, it also requires the most hardware to implement. However, the basic rule of thumb in microcomputer design is to eliminate as much system hardware as possible with program logic. Toward this end, an alternate display drive method that requires only two output ports is used in the INSTRUCTOR 50.

The first output port is a latch that drives the segment select lines connected in parallel to each of the eight digits. The second output port, an eight-bit latch, enables only one digit at a time. With this structure, the segment select lines can be time shared among the eight digits. The 2650 first enables a digit with the digit select output port and then writes that digit's character segments in the segment select output port. The process is repeated for each digit in a sequential fashion. If each digit is illuminated at a sufficiently fast frequency, about 100 Hz, the entire eight-digit display appears flicker free. Thus, considerable savings in display drive hardware is realized by substituting program complexity for output ports.

Because of the display's high-current requirements, the two output port latches require current buffering. A darlington transistor array on the output of each latch supplies the required current.

There are several methods of interfacing a microcomputer to an input keyboard. Here again the primary objective is to minimize the system hardware by placing as much of the control logic in the program as possible. The keyboard scan approach used by the INSTRUCTOR 50 arranges the two keyboards in a matrix. Since each function and data key is actually a two-terminal switch, a matrix can be formed by grouping the terminals of each switch into columns and rows. This organization is illustrated in Figure 7.4.

Referring to Figure 7.4, the column select signals, COL 1-COL 6, are driven by an output port, and the two sense signals, KR0-KR3, serve as the inputs to an input port. Given this structure, the 2650 can scan the keyboards to detect a switch closure as follows:

1) The processor writes a byte to the column select output port that drives one of the column select lines low.

2) The processor reads the row sense input port. If any of the keys in the selected column are depressed, a low is sensed on the corresponding row sense line.

3) The process is repeated for each column.

The keyboard interface column select operation is identical to that of the display digit select. Hence, a single output port serves both interfaces. The row sense input port is another input to the data bus multiplexer. When the 2650 executes an REDE instruction that specifies the row sense input port, the row sense signals are returned to the processor on the data bus via the multiplexer.

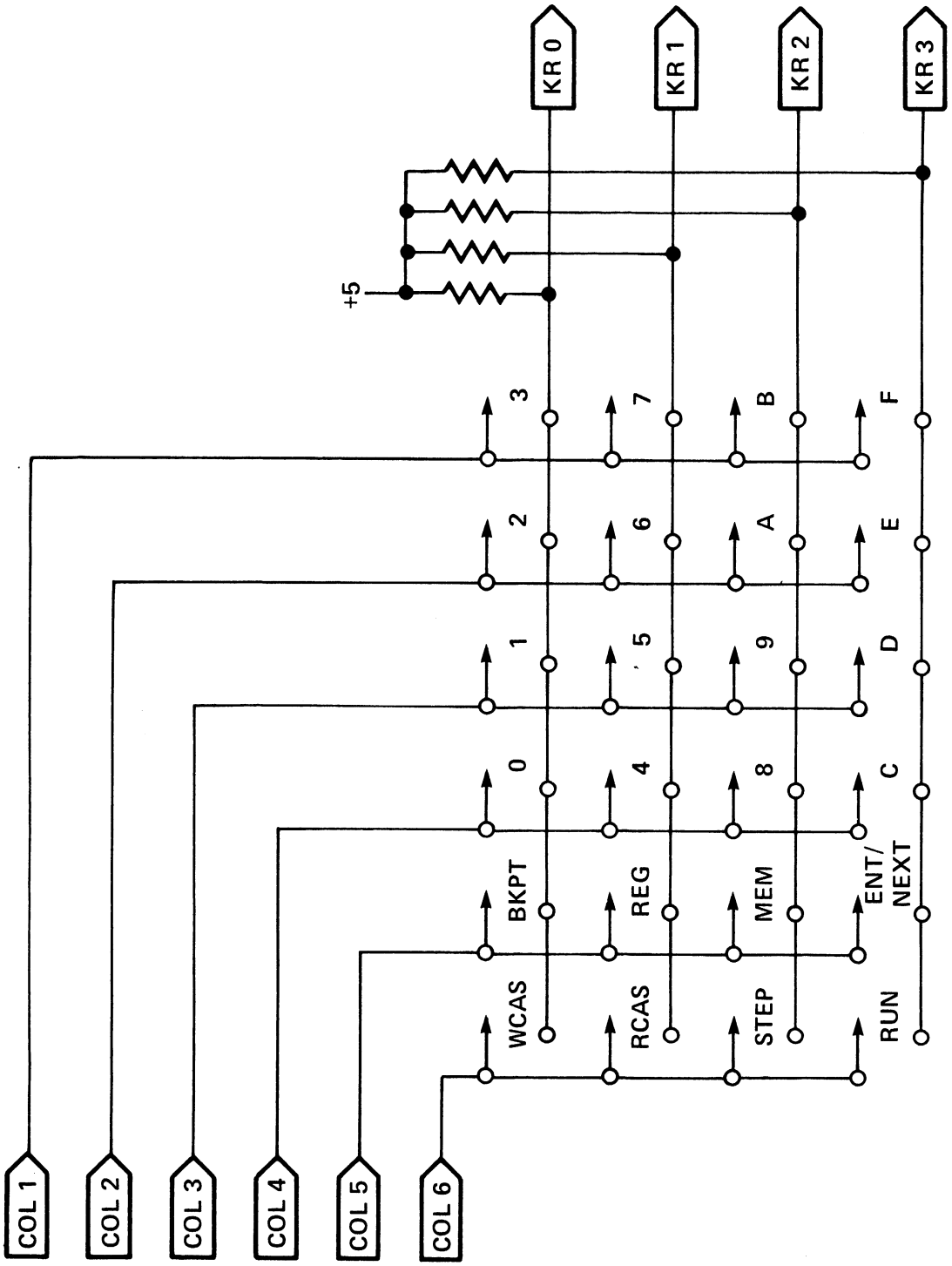


Figure 7.4: Keyboard Layout

Referring again to Figure 7.4, you will notice that four of the function keys, SENS, INT, MON, and RST, are not included in the switch matrix. The reason for their absence is that the functions they perform are independent of the monitor program. Since RST resets the 2650, this switch is connected to the 2650's RESET pin (after being OR'ed with the power on reset signal). Likewise, the SENS key is connected to the 2650 SENSE input pin. (Actually the 2650 SENSE pin is used for both the SENS key and the audio cassette interface. The signal presented to the 2650 depends on whether or not the 2650 is reading data from cassette). The INT key is connected directly to the INSTRUCTOR 50 interrupt logic, and the MON key is connected to the forced jump logic. The operation of these two keys is described under forced jump logic.

The Cassette Interface

The cassette interface is unique among the INSTRUCTOR 50's I/O devices in that it communicates with an analog system, a cassette tape recorder. In converts microprocessor-generated logic signals into an audio waveform for recording data, and converts the audio waveform returned from the recorder into a digital pulse stream that can be decoded by the processor when data is being read from the cassette.

The INSTRUCTOR 50 uses a two-bit output port for recording data onto cassette tape and a single-bit input port for reading the data back. Figure 7.5 illustrates the record waveforms required by this technique. The two signals, FREQ and ENV, are provided by a two-bit output port. These signals are combined with an open-collector NAND gate to form the write signal for the cassette. As shown in Figure 7.5, six pulses are used to record a 'zero' on the cassette, and three pulses to record a 'one'. The only exception to this recording format is the last bit of a byte. Six additional pulses are recorded for the last bit of a byte to mark byte boundaries (i.e., a one is nine pulses and a zero is twelve pulses).

Since only a single bit input port is required to read data back from cassette, the 2650's SENSE pin is used for this purpose. However, before the audio input is presented to the SENSE pin, it is digitized by a Schmidt trigger. The Schmidt trigger has about 1.5 volts of hysteresis that provides the read logic with necessary noise immunity.

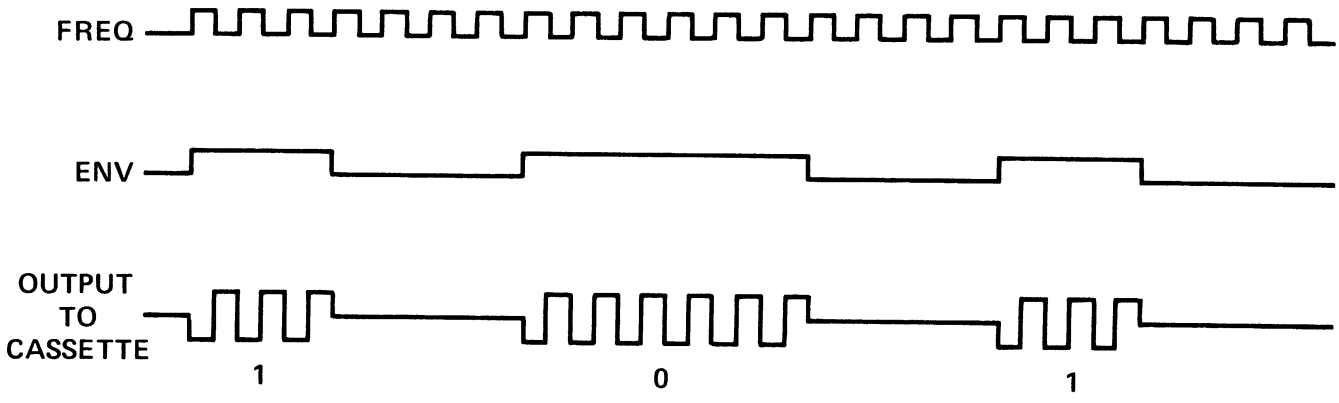


FIGURE 7.5: CASSETTE RECORD WAVEFORMS

Interrupt Logic

The INSTRUCTOR 50 can respond to interrupt requests from three possible sources: the INT key, the real-time clock derived from the power supply line frequency, or the S100 bus interface. As mentioned previously, interrupt source is determined by a switch located at the bottom of the INSTRUCTOR 50 case. This switch selects between the INT key and the real-time clock. A jumper option enables interrupt requests from the S100 bus interface.

The selected interrupt request source is input to a flip-flop that is set when an interrrupt request is received. The output of the flip-flop is connected to the INTREQ pin on the 2650. The 2650 responds to an interrupt request by asserting INTACK. INTACK, in turn, enables a tri-state drive that places the interrupt vector H'07' or H'87', depending on the position of the DIRECT/INDIRECT switch on the data bus. INTACK also resets the interrupt request flip-flop.

Forced Jump Logic

The INSTRUCTOR 50's Breakpoint and Single Step commands are implemented with a combination of firmware and hardware control. This hardware portion is called the forced jump logic. The forced jump logic returns program control to the monitor whenever a breakpoint is detected, after a single user instruction has been executed in the step mode, when the MON key is depressed, and when power is initially applied to the INSTRUCTOR 50.

The forced jump logic consists of the following logical elements:

1) The Return to Monitor Sequencer - This sequencer is responsible for returning program control to the monitor when the 2650 is executing a user program. The sequencer consists of a programmable counter and a 32 x 8 PROM. The PROM contains the data values of an absolute branch instruction. When the sequencer is active, the forced jump logic disables the INSTRUCTOR 50's normal instruction fetch mechanism and returns the absolute branch instruction stored in the PROM. The 2650 initializes the sequencer by loading the counter via an extended output port.

2) The Last Address Register - The Last Address Register (LAR) saves the last address issued by a user program before program control is returned to the monitor. This address points to the next instruction that the user program would execute if the return to monitor had not been activated. The monitor program reads the LAR to determine where the user program should resume execution after a STEP command has been completed or when a breakpoint is encountered.

3) Control Logic - The control logic performs general housekeeping functions such as loading the LAR, integrating interrupt requests with the return to monitor state sequencer, and loading the programmable counter.

The forced jump logic is enabled when power is first applied to the INSTRUCTOR 50, when the MON key is depressed, when a breakpoint is detected, and when the monitor program executes the STEP command. The resulting action taken by the forced jump logic when one of these events occurs is described below.

Power On (POR) or MON Key Depression

When power is applied to the INSTRUCTOR 50 or when the MON key is depressed, the 2650 is reset. The 2650 responds to a reset by clearing its internal program counter and fetching the instruction located at byte zero, page zero. However, when the 2650 places address H'0000' on the address bus, the forced jump logic disables the normal memory access mechanism and returns a NOP instruction value to the 2650 via the data bus. The 2650 executes the NOP and attempts to fetch an instruction at the next sequential address H'0001'. This instruction fetch generates an operation request (OPREQ). OPREQ is used to increment the sequencer counter. In this state, the return to monitor sequencer places the first byte of an unconditional branch instruction on the data bus. When the 2650 receives the BCTA, UN op-code, it generates two more OPREQs to fetch the branch address. Each OPREQ increments the counter and the PROM places the beginning address of the monitor, H'1800', on the data bus. At this point the 2650 executes the branch to monitor, and the forced jump logic returns to the idle state.

Breakpoint Detection

If the user has specified a breakpoint, the monitor program inserts a WRTC instruction at the breakpoint address specified. When the 2650 executes the WRTC instruction, a control signal is generated that produces the same results as the POR signal, and program control is returned to the monitor. A monitor software flag distinguishes this entry from a POR or MON key entry and causes a branch to the breakpoint routine.

Single Step

The execution of a single 2650 instruction in response to the STEP key is an excellent example of combined firmware/hardware control. When the STEP key is depressed, the monitor program fetches the instruction pointed to by the Program Counter and

calculates the number of OPREQs required to execute the instruction. The OPREQ counter (an extended I/O port) is then loaded with a value that corresponds to the number of OPREQs. The monitor then restores the user's program registers and status and branches to the instruction to be stepped. When the 2650 executes the instruction, the OPREQ counter, beginning at the present count, addresses "dummy states" of the return to monitor sequencer. That is, the locations addressed are not output on the data bus. When the last OPREQ of the instruction occurs, the output of the return to monitor PROM is enabled, and subsequent OPREQs return the unconditional branch to monitor instruction bytes to the processor.

If an interrupt request should occur during execution of the STEP instruction, the 2650 waits until the instruction has been completed before asserting INTACK. Conditioned by the forced jump control logic, INTACK becomes an address bit for the return to monitor PROM. While INTACK is high, another address bit reflects the position of the DIRECT/INDIRECT switch. In concert, these two address bits force the sequencer into one of two interrupt handling sequences: one for direct interrupts and another for indirect interrupts.

S100 Bus Interface

The S100 bus interface consists of tri-state drivers and receivers and a Field Programmable Gate Array (FPGA) which produces the S100 bus signals from logical combinations of 2650 control signals. Unfortunately, the S100 bus is far from standardized. Many of the signals are repetitious and different peripheral manufacturers make different demands of the bus. The FPGA enables you to modify the bus interface to meet any specific needs you may encounter. A detailed description of the S100 bus interface is given in Section 6.

System Power

The INSTRUCTOR 50 obtains its system power from one of two possible sources. The first source is an A-C wall transformer supplied with the INSTRUCTOR 50. The transformer provides the INSTRUCTOR 50 with 8 VAC (rms). On board, the A-C input is rectified, and the resulting D-C voltage is applied to a three-terminal regulator. The regulator supplies 5 VDC at 1.5 amps - the system power requirements of the INSTRUCTOR 50. The user may optionally change a wire jumper at the bottom of the printed circuit board to select unregulated 8 VDC from the S100 bus interface as input to the regulator.

In addition to the rectifier, the A-C input to the system is also applied to the resistive divider network. The reduced A-C voltage is input to a comparator that outputs a 60 Hz real-time clock (50 Hz in Europe and Japan). This real-time clock is available to the interrupt request logic via a select switch at the bottom of the printed circuit board. The wall transformer can be used to drive the real-time clock even if system power is derived from the S100 bus interface.

The USE Monitor

Without question, the most important component of any microcomputer (or any computer for that matter) is the system program. Every function or operation performed by a microcomputer is accomplished by executing a sequence of instructions within the system program.

Basically, the USE monitor is a collection of separate routines — one routine for each system command. A brief functional description of several routines with illustrative examples is provided in Section 4. This section provides a brief description of the command executive - a section of the monitor program that links the various command routines into a cohesive system program.

Figure 7.6 is a flowchart of the command routine executive section of USE. Whenever the forced jump logic returns program control to the monitor, monitor execution begins at H'1800', the first address of the executive. Beginning at this address, the first operation is to save the 2650 registers and Program Status Word. (These values are restored before program control is transferred to the user program). The next operation is to check certain software flags to determine how the forced jump logic was enabled. If it was tirggered by a breakpoint (WRTC instruction), program control is returned to the breakpoint routine. Similarly, if the forced jump logic was activated by the completion of a single-step sequence, program control is returned to the single-step routine. The alternatives to these two entry modes are power on and MON key depression. If the executive was entered via either of these two modes, the executive clears the breakpoint and step flags, since they may be on even if entry to the monitor was via power-on. Next, the display buffer pointer is set to the "HELLO" message table, and the DISPLAY subroutine is called. The monitor remains in this routine until a function key is depressed.

Upon returning from the DISPLAY subroutine, R0 contains the function key value. This value is used as an index to fetch a command routine address from the command address table. The address thus accessed is used for an absolute branch to one of the command routines. The executive is re-entered from any command routine when a function key is depressed. Hence, a new

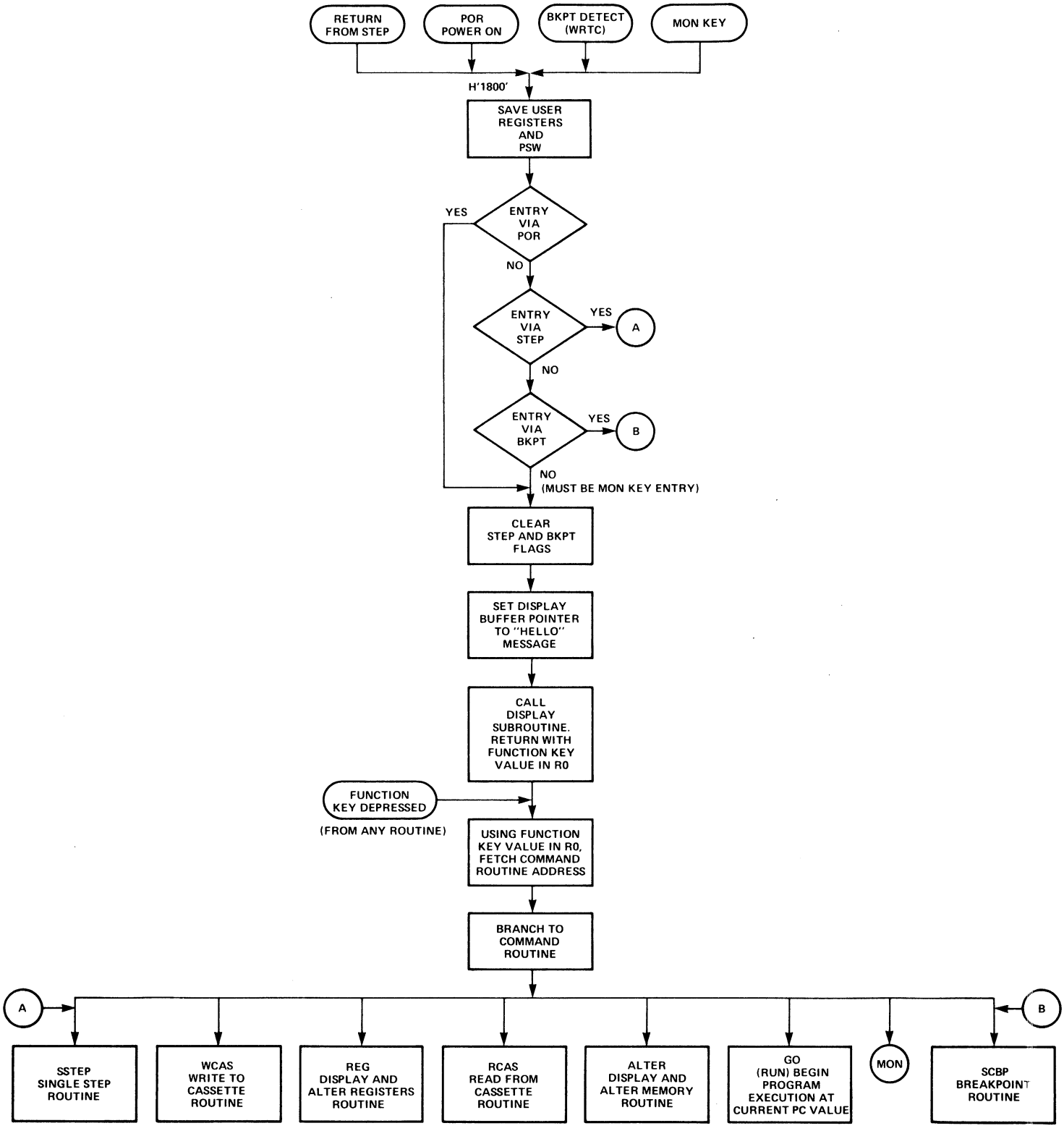


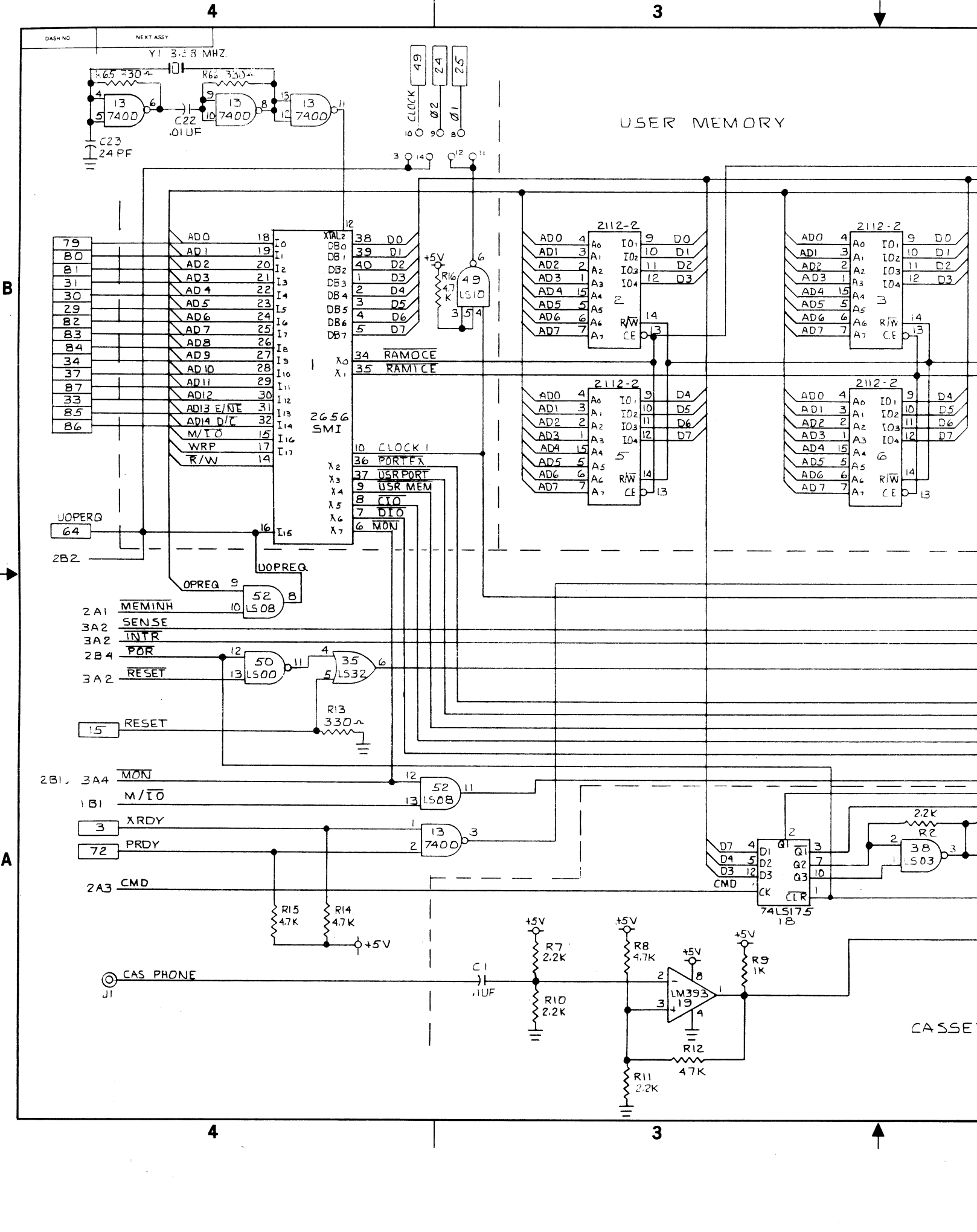
Figure 7.6: USE Command And Routine Executive

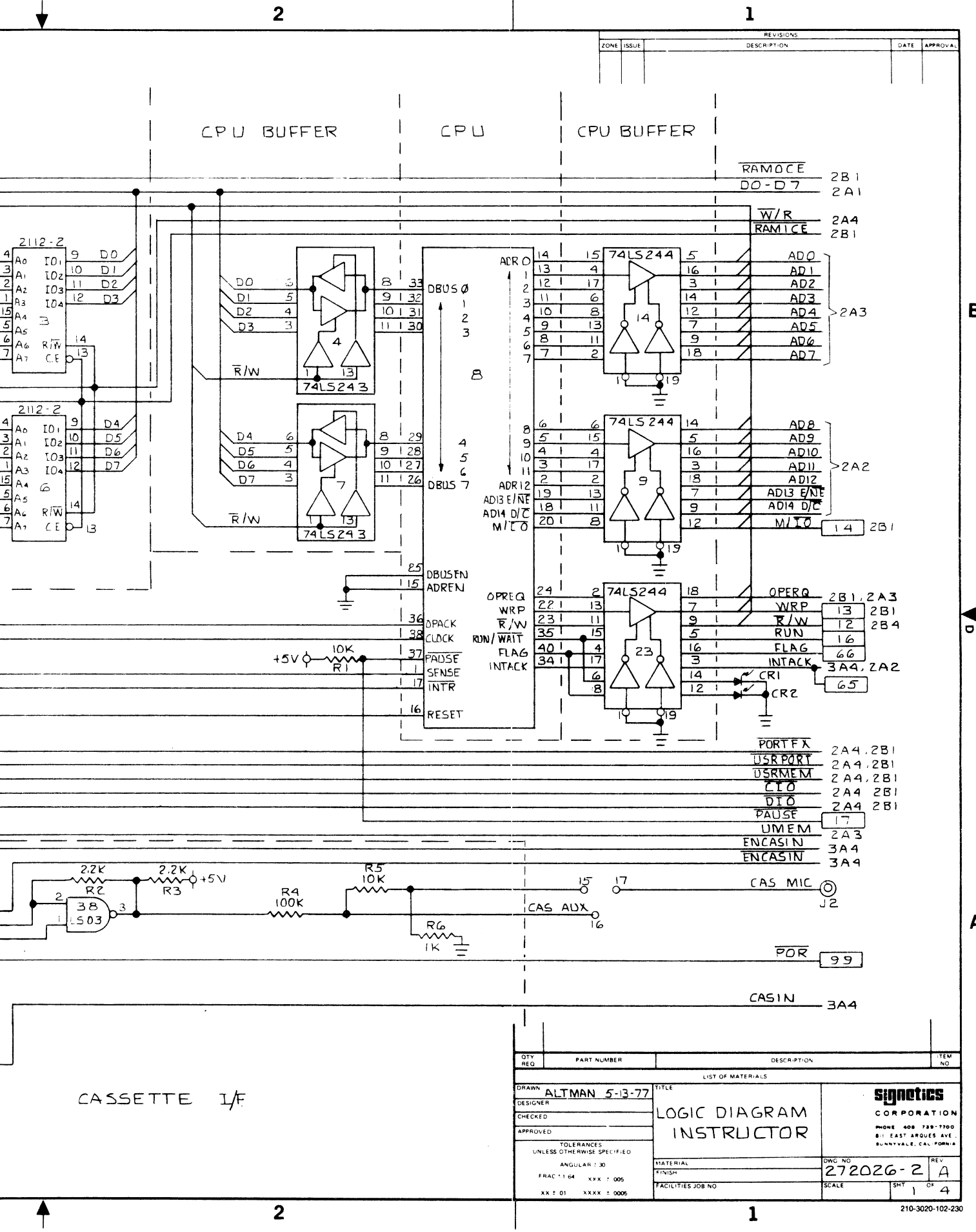
command address is accessed, and the monitor again branches to the specified command routine. Refer to the USE Program Listing in the appendices for detailed information on the USE routines.

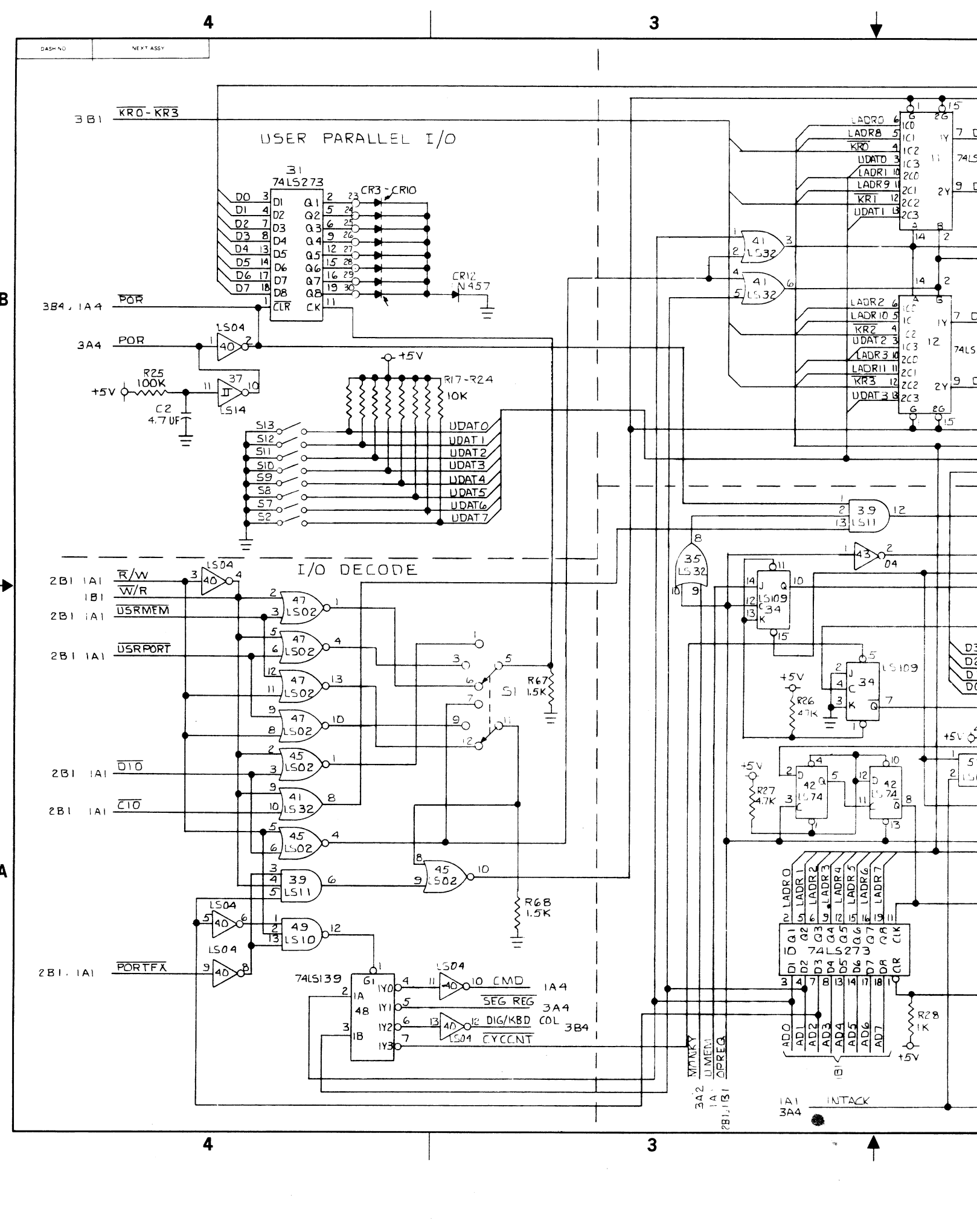
APPENDIX A — THE 2650 MICROPROCESSOR

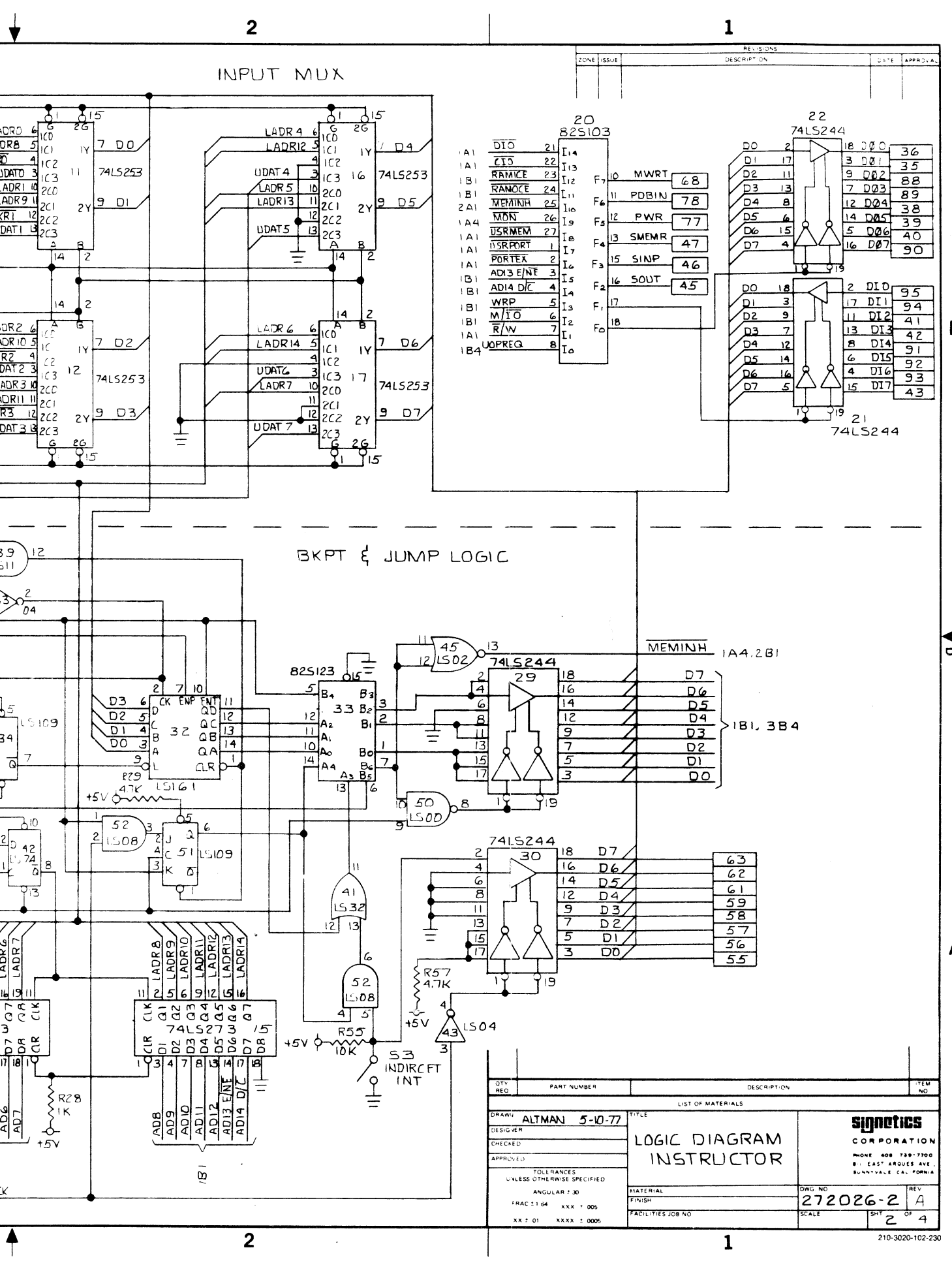
NOTE: Refer to the Signetics 2650 Microprocessor Manual accompanying this document.

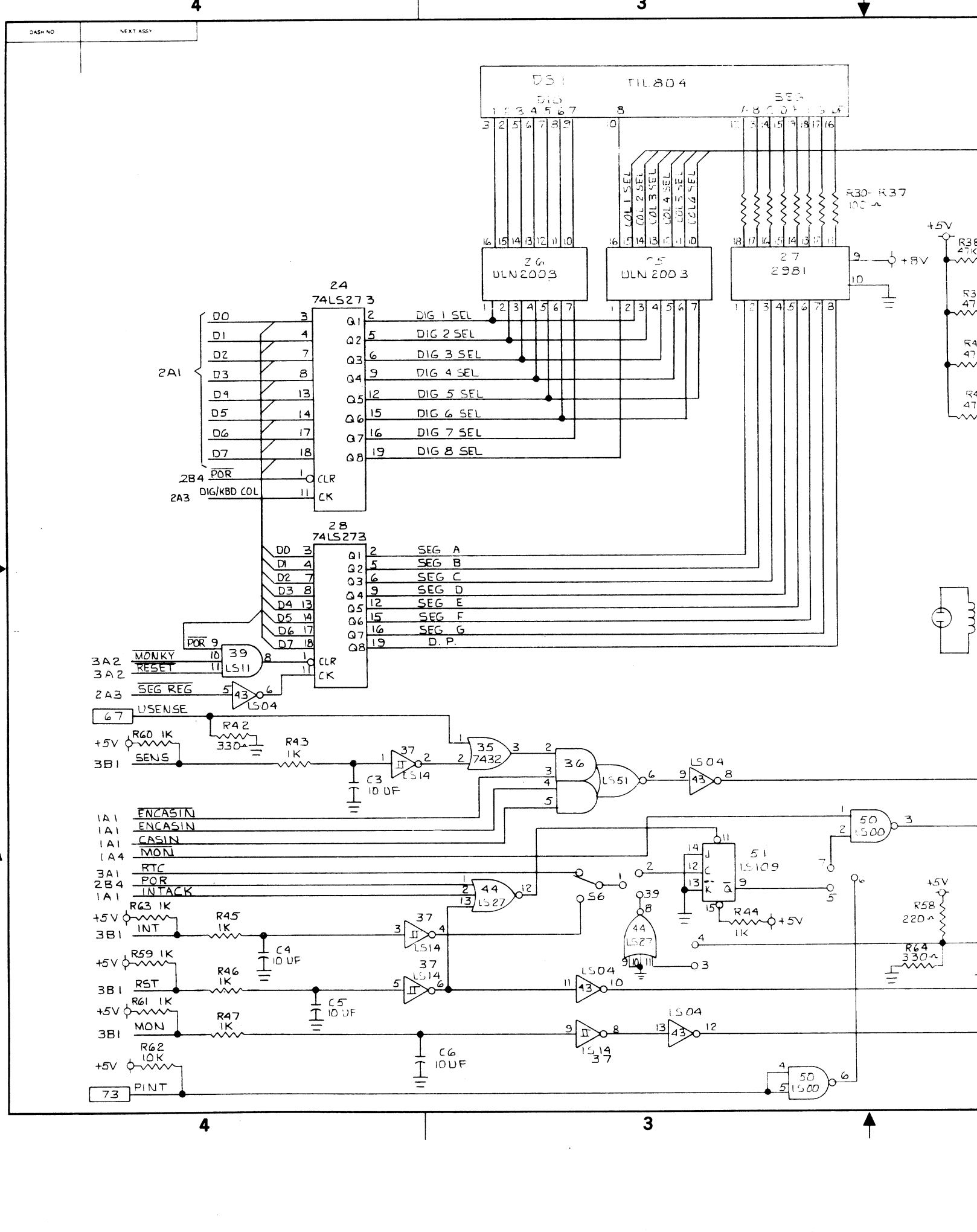
APPENDIX B —
INSTRUCTOR 50 SYSTEM SCHEMATICS


DASH NO
NEXT ASSY
Y1 3.58 MHZ
R65 330
R66 330
13 7400
C22 .01UF
C23 24 PF
CLOCK
Ø2
Ø1
USER MEMORY
2112-2
ADO
AD1
AD2
AD3
AD4
AD5
AD6
AD7
DO
D1
D2
D3
D4
D5
D6
D7
R/W
CE
2656 SMI
AD8
AD9
AD10
AD11
AD12
AD13 E/NE
AD14 D/C
M/IO
WRP
R/W
XTAL2
DB0
DB1
DB2
DB3
DB4
DB5
DB6
DB7
RAMOCE
RAMICE
CLOCK 1
PORTEX
USRPORT
USR MEM
CIO
DIO
MON
R16 4.7K
49 LS10
UOPERQ
64
2B2
UOPREQ
OPREQ
52 LS08
2A1 MEMINH
3A2 SENSE
3A2 INTR
2B4 POR
3A2 RESET
50 LS00
35 LS32
R13 330
15 RESET
2B1, 3A4 MON
1B1 M/IO
3 XRDY
72 PRDY
2A3 CMD
R15 4.7K
R14 4.7K
+5V
CAS PHONE
J1
C1 .1UF
R7 2.2K
R10 2.2K
R8 4.7K
LM393
R9 1K
R12 47K
R11 2.2K
74LS175
R2 2.2K
38 LS03
CASSE
B
A
4
3

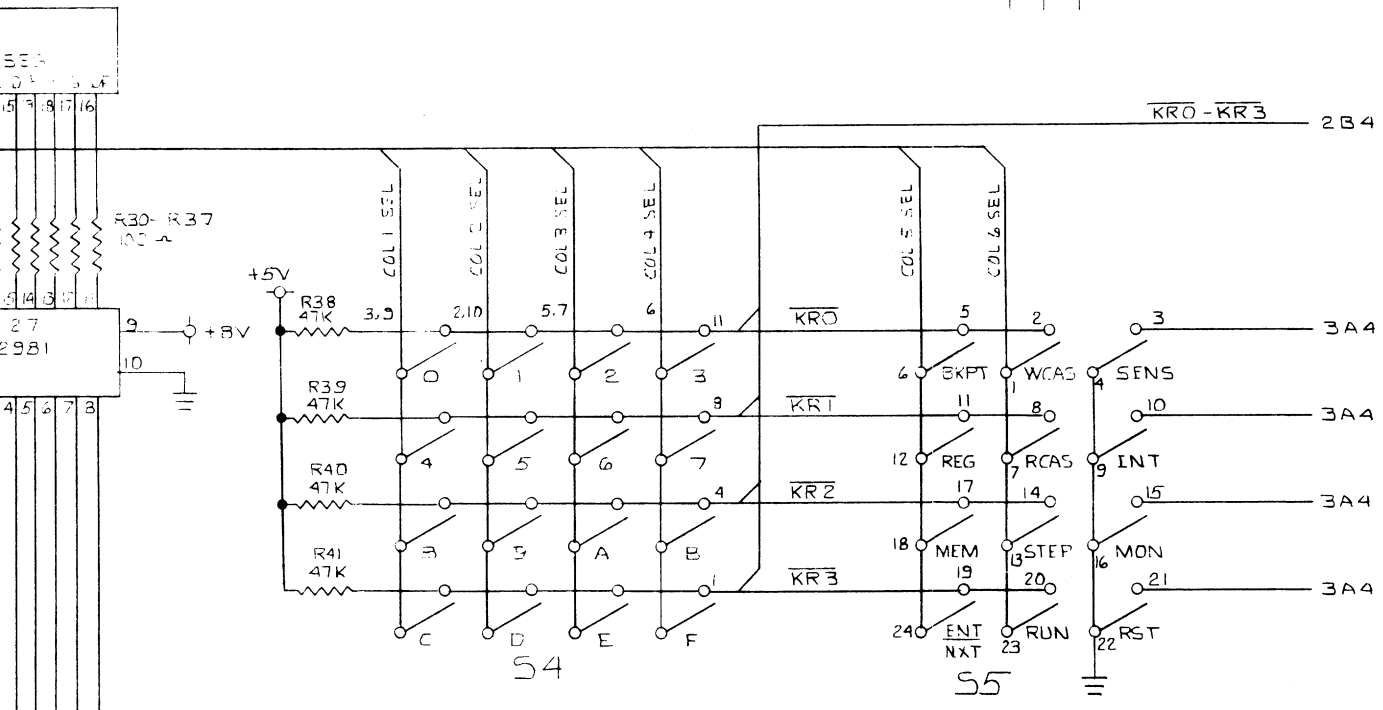

REVISIONS
ZONE ISSUE DESCRIPTION DATE APPROVAL
CPU BUFFER
CPU
CPU BUFFER
RAMOCE 2B1
DO-D7 2A1
W/R 2A4
RAMICE 2B1
2112-2
74LS243
74LS244
DBUS Ø
DBUS 7
DBUSEN
ADREN
OPACK
CLOCK
PAUSE
SENSE
INTR
RESET
ADR Ø
ADR12
ADI3 E/NE
ADI4 D/C
M/IO
OPREQ
WRP
R/W
RUN/WAIT
FLAG
INTACK
AD0
AD1
AD2
AD3
AD4
AD5
AD6
AD7
AD8
AD9
AD10
AD11
AD12
ADI3 E/NE
ADI4 D/C
M/IO
OPERQ
WRP
R/W
RUN
FLAG
INTACK
CR1
CR2
+5V
10K
R1
2.2K
R2
2.2K
R3
R4
100K
R5
10K
R6
1K
38 LS03
CAS MIC
J2
CAS AUX
PORTFX 2A4,2B1
USRPORT 2A4,2B1
USRMEM 2A4,2B1
CIO 2A4 2B1
DIO 2A4 2B1
PAUSE 17
UMEM 2A3
ENCASIN 3A4
ENCASIN 3A4
POR 99
CASIN 3A4
CASSETTE I/F
QTY REQ PART NUMBER DESCRIPTION ITEM NO
LIST OF MATERIALS
DRAWN ALTMAN 5-13-77
DESIGNER
CHECKED
APPROVED
TOLERANCES UNLESS OTHERWISE SPECIFIED
ANGULAR ± 30
FRAC ± 1/64 XXX ± .005
XX ± .01 XXXX ± .0005
TITLE
LOGIC DIAGRAM INSTRUCTOR
MATERIAL
FINISH
FACILITIES JOB NO.
signetics CORPORATION
PHONE 408 739-7700
811 EAST ARQUES AVE.
SUNNYVALE, CALIFORNIA
DWG NO 272026-2
REV A
SCALE
SHT 1 OF 4
DWG. NO.
210-3020-102-230

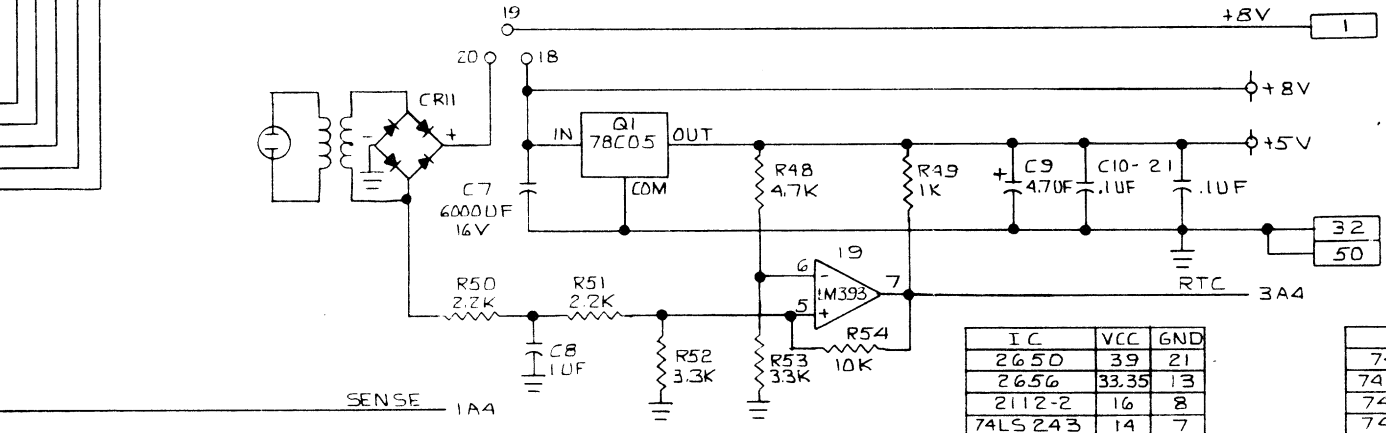

DASH NO
NEXT ASSY
4
3
USER PARALLEL I/O
3B1 KR0-KR3
31 74LS273
D0 D1 D2 D3 D4 D5 D6 D7
Q1 Q2 Q3 Q4 Q5 Q6 Q7 Q8
CLR CK
CR3-CR10
CR12 1N457
3B4, 1A4 POR
3A4 POR
LS04
R25 100K
+5V
C2 4.7 UF
LS14
+5V
R17-R24 10K
S13 S12 S11 S10 S9 S8 S7 S2
UDAT0 UDAT1 UDAT2 UDAT3 UDAT4 UDAT5 UDAT6 UDAT7
I/O DECODE
2B1 1A1 R/W
1B1 W/R
2B1 1A1 USRMEM
2B1 1A1 USRPORT
2B1 1A1 D10
2B1 1A1 C10
2B1 1A1 PORTFX
47 LS02
45 LS02
41 LS32
39 LS11
49 LS10
S1
R67 1.5K
R68 1.5K
74LS139
1G 1A 1B 1Y0 1Y1 1Y2 1Y3
LS04
10 CMD 1A4
SEG REG 3A4
12 DIG/KBD COL 3B4
CYCCNT
LADR0 LADR8 KR0 UDAT0 LADR1 LADR9 KR1 UDAT1
LADR2 LADR10 KR2 UDAT2 LADR3 LADR11 KR3 UDAT3
74LS
41 LS32
39 LS11
43
35 LS32
LS109
34
R26 4.7K
R27 4.7K
42 LS74
LADR0 LADR1 LADR2 LADR3 LADR4 LADR5 LADR6 LADR7
1D 74LS273
AD0 AD1 AD2 AD3 AD4 AD5 AD6 AD7
R28 1K
MMEM
UMEM
OPREQ
3A2
2B1,1B1
1A1 INTACK
3A4

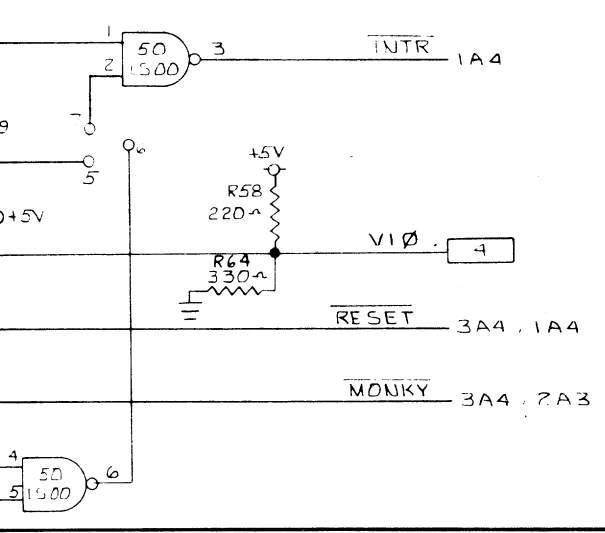

INPUT MUX
BKPT & JUMP LOGIC
74LS253
74LS244
82S103
82S123
74LS27
74LS08
74LS04
74LS32
74LS109
74LS161
MEMINH
1A4,2B1
1B1, 3B4
MWRT
PDBIN
PWR
SMEMR
SINP
SOUT
INDIRCFT INT
LOGIC DIAGRAM INSTRUCTOR
ALTMAN 5-10-77
272026-2
SHT 2 OF 4
DWG. NO.
signetics CORPORATION


DASH NO
NEXT ASSY
4
3
DS1
TIL804
DIG
SEG
R30- R37
10K Ω
+5V
R38
47K
R3
47
R4
47
R4
47
+8V
26
ULN2003
25
ULN2003
27
2981
COL 1 SEL
COL 2 SEL
COL 3 SEL
COL 4 SEL
COL 5 SEL
COL 6 SEL
24
74LS273
2A1
D0
D1
D2
D3
D4
D5
D6
D7
Q1
Q2
Q3
Q4
Q5
Q6
Q7
Q8
DIG 1 SEL
DIG 2 SEL
DIG 3 SEL
DIG 4 SEL
DIG 5 SEL
DIG 6 SEL
DIG 7 SEL
DIG 8 SEL
2B4 POR
2A3 DIG/KBD COL
CLR
CK
28
74LS273
SEG A
SEG B
SEG C
SEG D
SEG E
SEG F
SEG G
D. P.
POR 9
39
LS11
3A2 MONKY
3A2 RESET
2A3 SEG REG
43
LS04
67 USENSE
R42
330Ω
R60 1K
+5V
3B1 SENS
R43
1K
37
LS14
C3
10 UF
35
7432
36
LS51
LS04
1A1 ENCASIN
1A1 ENCASIN
1A1 CASIN
1A4 MON
3A1 RTC
2B4 POR
1A1 INTACK
44
LS27
S6
50
LS00
51
LS109
R44
1K
+5V
R58
220Ω
R64
330Ω
R63 1K
+5V
3B1 INT
R45
1K
C4
10 UF
R59 1K
+5V
3B1 RST
R46
1K
C5
10 UF
R61 1K
+5V
3B1 MON
R47
1K
C6
10 UF
R62
10K
+5V
73 PINT

REVISIONS				
ZONE	ISSUE	DESCRIPTION	DATE	APPROVAL







IC	VCC	GND
2650	39	21
2656	33.35	13
2112-2	16	8
74LS243	14	7
74LS244	20	10
74LS08	14	7
74LS00	14	7
74LS03	14	7
74LS175	16	8
LM393	8	4
74LS273	20	10
74LS253	16	8
74LS04	14	7
74LS02	14	7
74LS32	14	7
74LS11	14	7

IC	VCC	GND
74LS10	14	7
74LS139	16	8
74LS109	16	8
74LS161	16	8
82S123	16	8
74LS14	14	7
ULN2003	—	8
74LS27	14	7
74LS51	14	7
82S103	28	14
74LS74	14	7
7400	14	7

IC	+8V	GND
2981	9	10

QTY REQ	PART NUMBER	DESCRIPTION	ITEM NO

LIST OF MATERIALS

DRAWN ALTMAN 6-27-77	TITLE LOGIC DIAGRAM INSTRUCTOR	signetics CORPORATION	
DESIGNED			
CHECKED			
APPROVED			
TOLERANCES UNLESS OTHERWISE SPECIFIED ANGULAR ±30 FRACT ±1/64 XXX ± .005 XX ± .01 XXXX ± .0005	MATERIAL	DWG NO 272026-2	REV 2
	FINISH		
	FACILITIES JOB NO	SCALE	SHT 3 OF 4

210-3020-102-230

DWG. NO.

DASH NO	NEXT ASSY

2656 SMI PROGRAM CODING

FUNCTION SELECT	X0 E	X1 E	X2 E	X3 E	X4 E	X5 E	X6 E	X7 E			
	0	1	2	3	4	5	6	7	8	9	10
A0	X	X	X	1	1	X	X	X	1	X	X
A1	X	X	X	1	1	X	X	X	1	X	X
A2	X	X	X	1	1	X	X	X	1	X	X
A3	X	X	1	0	1	X	X	X	1	X	X
A4	X	X	1	0	1	X	X	X	1	X	X
A5	X	X	1	0	1	X	X	X	1	X	X
A6	X	X	1	0	1	X	X	X	1	X	X
A7	X	X	1	0	1	X	X	X	1	1	X
A8	0	1	X	X	1	X	X	X	1	1	X
A9	0	0	X	X	1	X	X	X	1	1	X
A10	0	0	X	X	1	X	X	X	1	1	X
A11	0	0	X	X	1	X	X	X	0	0	1
A12	0	0	X	X	0	X	X	1	1	1	1
A13	0	0	1	1	0	0	0	0	0	0	0
A14	0	0	X	X	0	0	1	0	0	0	0
OPREQ	1	1	1	1	1	1	1	X	1	1	1
M IO	1	1	0	0	1	0	0	X	1	1	1
WRP	1	1	1	1	1	1	1	X	1	1	1
	$\overline{\text{RAM 0 CE}}$	$\overline{\text{RAM 1 CE}}$	$\overline{\text{PORT FX}}$	$\overline{\text{USR PORT}}$	$\overline{\text{USR MEM}}$	$\overline{\text{CIO}}$	$\overline{\text{DIO}}$	$\overline{\text{MON}}$	SMI PORT	SMI RAM	SMI ROM

CLOCK SOURCE XTAL
CLOCK DIVIDE BY/4
NO DISABLE

82S123 PROGRAM TABLE

ADDRESS HEX	ADDRESS OCTAL	A4	A3	A2	A1	A0	OUTPUT B7	B6	B5	B4	B3	B2	B1	B0	
00	00	0	0	0	0	0	0	1	1	1	0	1	0	0	CØ
01	01	0	0	0	0	1	0	1	0	1	0	0	1	1	1F
02	02	0	0	0	1	0	0	1	0	1	0	0	1	0	18
03	03	0	0	0	1	1	0	1	0	1	0	0	0	0	ØØ
04	04	0	0	1	0	0	0	0	0	0	0	0	0	0	IDLE
05	05	0	0	1	0	1	0	0	0	0	0	0	0	0	IDLE
06	06	0	0	1	1	0	0	0	0	0	0	0	0	0	
07	07	0	0	1	1	1	0	0	0	0	0	0	0	0	IDLE
08	10	0	1	0	0	0	0	0	0	1	0	0	0	0	
09	11	0	1	0	0	1	0	0	0	1	0	0	0	0	
0A	12	0	1	0	1	0	0	0	0	1	0	0	0	0	
0B	13	0	1	0	1	1	0	0	0	1	0	0	0	0	
0C	14	0	1	1	0	0	0	0	0	1	0	0	0	0	
0D	15	0	1	1	0	1	0	0	0	1	0	0	0	0	
0E	16	0	1	1	1	0	0	0	0	1	0	0	0	0	
0F	17	0	1	1	1	1	0	0	0	1	0	0	0	0	
10	20	1	0	0	0	0	0	0	0	1	0	0	0	0	
11	21	1	0	0	0	1	0	0	0	1	0	0	0	0	DIRECT INT VECT
12	22	1	0	0	1	0	0	1	1	1	0	0	1	1	1F
13	23	1	0	0	1	1	0	1	0	1	0	0	1	0	18
14	24	1	0	1	0	0	0	1	0	1	0	0	0	0	ØØ
15	25	1	0	1	0	1	0	0	0	0	0	0	0	0	
16	26	1	0	1	1	0	0	0	0	0	0	0	0	0	
17	27	1	0	1	1	1	0	0	0	0	0	0	0	0	
18	30	1	1	0	0	0	0	0	0	0	0	0	0	0	
19	31	1	1	0	0	1	0	0	0	1	0	0	0	0	INDIRECT INT VECTO
1A	32	1	1	0	1	0	0	0	0	1	0	0	0	0	HI ADDR BYTE
1B	33	1	1	0	1	1	0	0	0	1	0	0	0	0	LO ADDR BYTE
1C	34	1	1	1	0	0	0	1	1	1	0	0	1	1	1F
1D	35	1	1	1	0	1	0	1	0	1	0	0	1	0	18
1E	36	1	1	1	1	0	0	1	0	1	0	0	0	0	ØØ
1F	37	1	1	1	1	1	0	0	0	0	0	0	0	0	
							DATA BUS ENABLE		LAST ADDR LOAD	$\overline{\text{CNTR STOP}}$		D6, D7	D3, D4	D0, D1, D2	

SINGLE STEP

REVISIONS				
ZONE	ISSUE	DESCRIPTION	DATE	APPROVAL

TABLE

TPUT

B4	B3	B2	B1	B0	
	0	1	0	0	CØ
	0	0	1	1	1F
	0	0	1	0	18
	0	0	0	0	ØØ
0	0	0	0	0	IDLE
0	0	0	0	0	IDLE
0	0	0	0	0	
0	0	0	0	0	IDLE
	0	0	0	0	
	0	0	0	0	
	0	0	0	0	
	0	0	0	0	
	0	0	0	0	
	0	0	0	0	
	0	0	0	0	
	0	0	0	0	
	0	0	0	0	DIRECT INT VECTOR
	0	0	1	1	1F
	0	0	1	0	18
	0	0	0	0	ØØ
0	0	0	0	0	
	0	0	0	0	
	0	0	0	0	
	0	0	0	0	
	0	0	0	0	INDIRECT INT VECTOR
	0	0	0	0	HI ADDR BYTE
	0	0	0	0	LO ADDR BYTE
	0	0	1	1	1F
	0	0	1	0	18
	0	0	0	0	ØØ
0	0	0	0	0	

SINGLE STEP

DIRECT

INTERRUPT SPACE

INDIRECT

D6, D7 | D3, D4 | D0, D1, D2

8251Ø3 PROGRAM TABLE

OUTPUT POLARITY		INPUT VARIABLE															
		Iø	I1	I2	I3	I4	I5	I6	I7	I8	I9	IA	IB	IC	ID	IE	IF
FØ	L	H	H	–	–	–	–	H	H	H	H	H	H	H	H	H	–	ENABLE OUTPUT DATA
F1	L	H	L	–	–	–	–	H	H	H	H	H	H	H	H	H	–	ENABLE INPUT DATA
F2	H	–	H	L	–	–	–	H	H	H	H	H	H	H	H	H	–	SOUT
F3	H	–	L	L	–	–	–	H	H	H	H	H	H	H	H	H	–	SINP
F4	H	–	L	H	–	–	–	H	H	H	H	H	H	H	H	H	–	SMEMR
F5	L	H	H	–	H	–	–	H	H	H	H	H	H	H	H	H	–	PWR
F6	H	H	L	–	–	–	–	H	H	H	H	H	H	H	H	H	–	PDBIN
F7	H	H	H	H	–	–	–	H	H	H	H	H	H	H	H	H	–	MWRT
		CHAR/$\overline{KK}$	K/$\overline{D}$	$\overline{MRT}$	WRK	ALT C/$\overline{D}$	ADR E/$\overline{NE}$	PORTFX	$\overline{PORT}$	DENMEM	$\overline{MOD}$	MEMINH	$\overline{HAMACE}$	$\overline{HAMACE}$	$\overline{CID}$	DID	

ACTIVE HIGH = H
ACTIVE LOW = L

QTY REQ	PART NUMBER	DESCRIPTION	ITEM NO
		LIST OF MATERIALS	

DRAWN	TITLE	signetics CORPORATION	
DESIGNER	INSTRUCTOR PROGRAM TABLES	PHONE 408 739-7700 811 EAST ARQUES AVE. SUNNYVALE, CALIFORNIA	
CHECKED			
APPROVED			
TOLERANCES UNLESS OTHERWISE SPECIFIED ANGULAR ± 30 FRAC ± 1/64 XXX ± .005 XX ± .01 XXXX ± .0005	MATERIAL	DWG NO 272062-2	REV A
	FINISH		
	FACILITIES JOB NO	SCALE	SHT 4 OF 9

210-3020-102-230

APPENDIX C — USE PROGRAM LISTINGS

```
LINE ADDR  OBJECT  E SOURCE

0002               *PROGRAM WRITTEN BY DAVE WOTRING
0003               ********************************************************************************
0004               *   EQUATE TABLES
0005               *
0006               *   REGISTER EQUATES
0007 0000          R0     EQU     0       REGISTER 0
0008 0001          R1     EQU     1       REGISTER 1
0009 0002          R2     EQU     2       REGISTER 2
0010 0003          R3     EQU     3       REGISTER 3
0011               *  CONDITION CODES
0012 0001          P      EQU     1       POSITIVE RESULT
0013 0000          Z      EQU     0       ZERO RESULT
0014 0002          NG     EQU     2       NEGATIVE RESULT
0015 0002          LT     EQU     2       LESS THAN
0016 0000          EQ     EQU     0       EQUAL TO
0017 0001          GT     EQU     1       GREATER THAN
0018 0003          UN     EQU     3       UNCONDITIONAL
0019               *  PSW LOWER EQUATES
0020 00C0          CC     EQU     H'C0'   CONDITIONAL CODES
0021 0020          IDC    EQU     H'20'   INTERDIGIT CARRY
0022 0010          RS     EQU     H'10'   REGISTER BANK
0023 0008          WC     EQU     H'08'   1=WITH 0=WITHOUT CARRY
0024 0004          OVF    EQU     H'04'   OVERFLOW
0025 0002          COM    EQU     H'02'   1=LOGIC 0=ARITHMETIC COMPARE
0026 0001          C      EQU     H'01'   CARRY/BORROW
0027               *  PSW UPPER EQUATES
0028 0080          SENS   EQU     H'80'   SENSE BIT
0029 0040          FLAG   EQU     H'40'   FLAG BIT
0030 0020          II     EQU     H'20'   INTERRUPT INHIBIT
0031 0007          SP     EQU     H'07'   STACK POINTER
0032               *  IO PORT DEFINITIONS
0033 0007          LEDS   EQU     H'07'   USER EXTENDED IO PORT
0034               *  INTERUPT VECTORS
0035 0007          UINTV  EQU     H'07'   USER DIRECT INTERUPT VECTOR
0036 0087          UINTVI EQU     H'87'   USER INDIRECT INTERUPT VECTOR
0037               * HARDWARE DEFINITIONS
0038 00FE          KBDIN  EQU     H'FE'   ADDRESS OF KBD IO PORT
0039 00F9          SEG    EQU     H'F9'   IO ADDRESS OF SEGMENT DRIVER
0040 00F9          DISP   EQU     SEG
0041 00FA          DIGIT  EQU     H'FA'   ADDRESS OF DIGIT ENABLE
0042 00F8          CTBYT  EQU     H'F8'   ADDRESS OF CONTROL BYTE
0043 00F8          CAS    EQU     CTBYT   ADDRESS OF CASSETTE INTERFACE
0044 00FB          OPRQCT EQU     H'FB'   ADDRESS OF OPREQ COUNTER
0045 00FD          LADRH  EQU     H'FD'   ADDRESS OF LAST ADDRESS REG HI BYTE
0046 00FC          LADRL  EQU     H'FC'   ADDRESS OF LAST ADDRESS REG LO BYTE
```

```
LINE ADDR  OBJECT  E SOURCE

0048               ************************************************
0049               *
0050 0000                 ORG    H'1800'-64     TRAINING CARD RAM AREA
0051               *
0052 17C0          SCTCH  RES    8       8 BYTE SCRATCH AREA
0053 17C6          TEMP   EQU    SCTCH+6 TEMP STORAGE
0054 17C8          EAD    RES    2       STOP ADDRESS FOR WCAS
0055 17CA          BAD    RES    2       BEGINNING ADDRESS FOR WCAS
0056 17CC          BPD    RES    1       DATA TO BE RESTORED IN BREAK LOC
0057 17CD          BPL    RES    2       ADDRESS OF BREAK POINT LOC
0058 17CF          BPF    RES    1       BREAK POINT SET FLAG
0059 17D0          SSF    RES    1       SINGLE STEP SET FLAG
0060 17D1          DISBUF RES    8       8 BYTE DISPLAY REGISTER
0061 17D9          SAVREG RES    4       A PLACE TO SAVE R0 THRU R3 OF ONE BANK
0062 17DD          MEM    RES    2       ADDRESS FOR ALTER OR PATCH COMMAND
0063 17DF          FID    RES    2       FILE ID FLAG AND FILE ID
0064 17E1          BCC    RES    1       BLOCK CHECK CHAR
0065 17E2          BSTT   RES    1       SAVE UNITS DIGIT
0066 17E3          T      RES    2       TEMP REGISTER
0067 17E5          T1     RES    1       TEMP REGISTER
0068 17E6          T2     RES    1       TEMP REGISTER
0069 17E7          T3     RES    1       TEMP REGISTER
0070 17E8          LADR   RES    2       COPY OF LAST ADDRESS REGISTER
0071 17EA          SLADR  RES    2       SAVE LOCATION FOR LADR
0072 17EC          KFLG   RES    2       KBD SCAN FLAGS
0073 17EE                 RES    1       KBD DEBOUNCE COUNT
0074 17EF          ALTF   RES    1       DISPLAY AND ALTER FLAG
0075 17F0          RESTF  RES    1       RESTORE REGISTERS FLAG
0076 17F1          IFLG   RES    1       INTERUPT INHIBIT FLAG
0077 17F2          UREG   RES    12      STORAGE FOR USER REGISTERS
0078 17FE          PWRON  RES    2       WHEN POWER ON THESE LOC CONTAIN H'5946'
0079               ************************************************
```

```
LINE ADDR  OBJECT  E SOURCE

0081               ************************************************************
0082 1800                ORG H'1800'     BEGINING OF TRAINING CARD ROM AREA
0083               ************************************************************
0084               *
0085               *SAVE ALL REGISTERS UPON ENTRY TO PROGRAM
0086               *
0087               *REGISTERS USED
0088               *
0089               *R0 THRU R3' PSU PSL
0090               *
0091               *SUBROUTINES CALLED
0092               *
0093               *NONE
0094               *
0095               *RAM MEMORY USED
0096               *
0097               *UREG  = R0
0098               *UREG+1        = R1
0099               *UREG+2        = R2
0100               *UREG+3        = R3
0101               *UREG+4        = R1'
0102               *UREG+5        = R2'
0103               *UREG+6        = R3'
0104               *UREG+7        = PSU
0105               *UREG+8        = PSL
0106               *UREG+9        = PPSL INSTRUCTION OPCODE
0107               *UREG+10       = PSL
0108               *UREG+11       = RETC,UN       INSTRUCTION OPCODE
0109               *
0110               ****************************************************************
0111               *
0112 1800 C870     SAVRG  STRR,R0 UREG    SAVE R0
0113 1802 13              SPSL            GET PSL
0114 1803 C875            STRR,R0 UREG+8  SAVE PSL
0115 1805 C875            STRR,R0 UREG+10 SAVE PSL        FOR RESTORE ROUTINE
0116 1807 12              SPSU            GET PSU
0117 1808 C86F            STRR,R0 UREG+7  SAVE PSU
0118 180A 7620            PPSU    II      SET INTERUPT INHIBIT
0119 180C 7510            CPSL    RS      CLEAR REGISTER SWITCH
0120 180E C963            STRR,R1 UREG+1  SAVE R1
0121 1810 CA62            STRR,R2 UREG+2  SAVE R2
0122 1812 CB61            STRR,R3 UREG+3  SAVE R3
0123 1814 7710            PPSL    RS      SET REGISTER SWITCH
0124 1816 C95E            STRR,R1 UREG+4  SAVE R1'
0125 1818 CA5D            STRR,R2 UREG+5  SAVE R2'
0126 181A CB5C            STRR,R3 UREG+6  SAVE R3'
0127 181C 75FF            CPSL    255     CLEAR PSL
0128 181E 7702            PPSL    COM     DO LOGICAL COMPARES
0129 1820 54FD            REDE,R0 LADRH   GET LAST ADDRESS HI BYTE
0130 1822 CC17E8          STRA,R0 LADR    SAVE IN MEMORY
0131 1825 54FC            REDE,R0 LADRL   GET LAST ADDRESS LO BYTE
0132 1827 CC17E9          STRA,R0 LADR+1  SAVE IT
0133 182A 20              EORZ    R0      GET A 0
0134 182B CC17F1          STRA,R0 IFLG    CLEAR INTERUPT INHIBIT FLAG
```

```
LINE ADDR  OBJECT  E SOURCE

0136               ****************************************************************
0137               *
0138               *
0139               *
0140               *PROGRAM ENTRY ROUTINE
0141               *
0142               *
0143               *DECIDES HOW REACHED ENTRY POINT OF PROGRAM
0144               *
0145               *1 POWER ON
0146               *2 SINGLE STEP
0147               *3 MONITOR PUSHBUTTON ON KEY BOARD
0148               *4 BREAKPOINT
0149               *
0150               *
0151               *
0152               *
0153               ****************************************************
0154               *
0155 182E 084E     BEG    LODR,R0 PWRON   CHECK POWER ON FLAG
0156 1830 E459            COMI,R0 H'59'   AFTER POWER VALUE OF FLAG IS
0157               *                      H'5946'
0158 1832 9813     BEG1   BCFR,EQ INIT    IF NOT CORRECT THIS IS POWER ON
0159               *                      GO INITIALIZE THE MONITOR FLAGS
0160 1834 0849     BEG3   LODR,R0 PWRON+1 CHECK LO BYTE OF POWER ON FLAG
0161 1836 E446            COMI,R0 H'46'   IS SECOND BYTE CORRECT
0162 1838 980D            BCFR,EQ INIT IF NOT INITIALIZE THE PROGRAM
0163 183A 0C17D0          LODA,R0 SSF     CHECK THE SINGLE STEP FLAG
0164 183D 9C19C9          BCFA,EQ SGLSTP  IF FLAG THEN GO SINGLE STEP
0165 1840 0899            LODR,R0 *MON+1     SEE IF BREAK POINT ENABLED
0166 1842 9C197A   BEG2   BCFA,EQ BRKPT  GO EXECUTE THE BREAK POINT ROUTINE
0167 1845 1B13            BCTR,UN MON     MUST BE MONITOR KEY
0168               *
```

```
LINE ADDR  OBJECT  E SOURCE

0170                ****************************************************
0171                *
0172                *
0173                *KEY BOARD MONITOR ROUTINE
0174                *
0175                *
0176                *REGISTERS USED
0177                *
0178                *R0 SCRATCH
0179                *R1 SCRATCH
0180                *R2 SCRATCH
0181                *R3 NOT USED
0182                *
0183                *SUBROUTINES USED
0184                *
0185                *MOV MOVE DATA TO DISPLAY BUFFER
0186                *DISPLY DISPLAY MESSAGE AND KEY BOARD SCAN
0187                *
0188                *
0189                *RAM MEMORY USED
0190                *
0191                *PWRON POWER ON FLAG
0192                *SSF   SINGLE STEP FLAG
0193                *BPF   BREAK POINT FLAG
0194                *BPL   BREAK POINT LOCATION
0195                *
0196                ************************************************
0197                *
0198 1847 0459      INIT   LODI,R0 H'59'   SET THE POWER ON FLAG
0199 1849 CC17FE           STRA,R0 PWRON   TO POWER ON VALUE H'5946'
0200 184C 0446             LODI,R0 H'46'
0201 184E CC17FF           STRA,R0 PWRON+1
0202 1851 20               EORZ    R0      GET A 0
0203 1852 CC17DD           STRA,R0 MEM     PRESET INDIRECT ADDRESS MEM
0204 1855 CC17DE           STRA,R0 MEM+1
0205 1858 C881             STRR,R0 *MON+1  CLEAR BREAK POINT FLAG
0206 185A 0C17CF    MON    LODA,R0 BPF     GET BREAK POINT FLAG
0207 185D 180F             BCTR,EQ MON5    BREAK POINT NOT SET
0208 185F 0C17CC           LODA,R0 BPD     GET BREAK POINT DATA
0209 1862 CC97CD           STRA,R0 *BPL    CLEAR BREAK POINT
0210 1865 EC97CD           COMA,R0 *BPL    CHECK DATA STORED CORRECTLY
0211 1868 1804             BCTR,EQ MON5    BREAK POINT CLEARED OK
0212 186A 0701             LODI,R3 1       BREAK POINT DIDN'T CLEAR
0213 186C 9BE8             ZBRR    *ERR    GOTO ERROR
0214 186E 20        MON5   EORZ    R0      GET A 0
0215 186F D4F8             WRTE,R0 CTBYT   CLEAR CONTROL BYTE
0216 1871 CC17D0           STRA,R0 SSF     CLEAR SINGLE STEP FLAG
0217 1874 051F      MON3   LODI,R1 <HELLO-1        GET ADDRESS OF HELLO MESSAGE
0218 1876 0694             LODI,R2 >HELLO-1
0219 1878 BBFE      MON1   ZBSR *MOV     MOVE MESSAGE TO DISBUF
0220 187A 20        MON4   EORZ    R0      SET FLAG TO WAIT FOR ENTRY
0221 187B BBEC             ZBSR    *DISPLY DISPLAY MESSAGE AND SCAN KEYBOARD
0222 187D F680      MON2   TMI,R2  H'80'   CHECK COMMAND FLAG
0223 187F 9816             BCFR,EQ ERR2     IF FLAG NOT SET ERROR
0224 1881 460F             ANDI,R2 H'0F'   MASK COMMAND VALUE
```

```
LINE ADDR  OBJECT  E SOURCE

0225 1883 E607            COMI,R2 7       MAX COMMAND VALUE
0226 1885 1910            BCTR,GT ERR2     ERROR CODE VALUE TO LARGE
0227 1887 D2              RRL,R2          MULTIPLY INDEX BY 2
0228 1888 0E78A4          LODA,R0 CMD,R2  SET UP AN INDIRECT ADDRESS
0229 188B CC17E3          STRA,R0 T       TO THE FUNCTION WANTED
0230 188E 0E78A5          LODA,R0 CMD+1,R2
0231 1891 CC17E4          STRA,R0 T+1
0232 1894 1F97E3          BCTA,UN *T      EXECUTE A COMMAND
0233                 *
0234                 *
0235 1897 0702       ERR2 LODI,R3 2       INVALID COMMAND SEQUENCE
0236 1899 051F       ERRI LODI,R1 <ERROR-1  GET ADDRESS OF ERROR MESSAGE
0237 189B 0684            LODI,R2 >ERROR-1
0238 189D BBFE            ZBSR    *MOV    MOVE MESSAGE TO DISBUF
0239 189F CF17D8          STRA,R3 DISBUF+7       WRITE THE ERROR NUMBER
0240 18A2 1B56            BCTR,UN MON4    GO LOOK FOR NEW COMMAND
0241                 *
0242                 *COMMAND ADDRESS TABLE
0243                 *
0244 18A4 1C91       CMD    ACON    WCAS    WRITE CASSETTE COMMAND
0245 18A6 1D61              ACON    SCBP     BREAK POINT COMMAND
0246 18A8 1BAC              ACON    RCAS    READ CASSETTE COMMAND
0247 18AA 1A7E              ACON    REG     REGISTER DISPLAY AND ALTER COMMAND
0248 18AC 18B4              ACON    SSTEP      SINGLE STEP COMMAND
0249 18AE 1A0C              ACON    ALTER    DISPLAY AND ALTER MEMORY
0250 18B0 1E59              ACON    GO      GOTO COMMAND
0251 18B2 187A              ACON    MON4    ENTR/NEXT KEY IS NOT COMMAND
0252                 *
```

```
LINE ADDR  OBJECT  E SOURCE

0254                ************************************************************
0255                *
0256                *
0257                *SINGLE STEP ROUTINES
0258                *THIS ROUTINE WRITTEN BY BBC
0259                *
0260                * PROCESSOR TRANSFERS CONTROL TO USER PROGRAM
0261                * AFTER COMPUTING THE NUMBER OF OPREQ'S TILL
0262                * THE NEXT INSTRUCTION FETCH
0263                *
0264                *
0265                *REGISTERS USED
0266                *
0267                *R0 THRU R3 SCRATCH
0268                *
0269                *
0270                *SUBROUTINES CALLED
0271                *
0272                *RLADR RESTORE LAST ADDRESS REGISTER
0273                *
0274                *
0275                *RAM LOCATIONS USED
0276                *
0277                *LADR  LAST ADDRESS REGISTER
0278                *T3    TEMP REGISTER
0279                *TEMP  TEMP REGISTER
0280                *SCTCH SCRATCH REGISTER
0281                *
0282                ************************************************************
0283 0000          OVHD   EQU     0       NEGATIVE NUMBER OF OPREQ'S
0284                *
0285                *
0286                *CHECK IF NEXT SINGLE STEP IS IN MONITOR AREA
0287                *
0288 18B4 0C17E8   SSTEP  LODA,R0 LADR    GET MSB OF LADR
0289 18B7 E410            COMI,R0 H'10'   IS ADDRESS LT H'1000'
0290 18B9 1A08            BCTR,LT SSTEP1  GO SINGLE STEP
0291 18BB E420            COMI,R0 H'20'   IS ADDRESS GT OR EQ H'2000'
0292 18BD 9A04            BCFR,LT SSTEP1  GO SINGLE STEP
0293 18BF 0709            LODI,R3 9       NEXT SINGLE STEP ENTERS MONITOR
0294 18C1 9BE8            ZBRR    *ERR    GOTO ERROR
0295 18C3 047F     SSTEP1 LODI,R0 127     SET SINGLE STEP FLAG
0296 18C5 CC17D0          STRA,R0 SSF     STORE IT
0297 18C8 0420            LODI,R0 H'20'   SET THE INTERUPT INHIBIT
0298 18CA CC17F1          STRA,R0 IFLG    SAVE IN INTERUPT INHIBIT FLAG
0299 18CD 7508            CPSL WC         CLEAR WITH CARRY IF SET
0300 18CF 0601     SSTEP2 LODI,R2 1       SET INDEX
0301 18D1 0EF7E8          LODA,R0 *LADR,R2       GET SECOND BYTE OF INSTRUCTION
0302 18D4 CC17E7          STRA,R0 T3      SAVE IT FOR LATER
0303 18D7 0F97E8          LODA,R3 *LADR   GET NEXT INSTRUCTION
0304 18DA 03              LODZ    R3      SAVE INSTRUCTION IN R0
0305 18DB 471C            ANDI,R3 H'1C'   EXTRACT INSTRUCTION CLASS
0306 18DD 0500            LODI,R1 OVHD    SET OVERHEAD OPREQ COUNT
0307 18DF 0605            LODI,R2 5       SHIFT OR MOVE COUNT
0308 18E1 F420            TMI,R0  H'20'   TEST FOR ODD OPCODE IN CLASS4
```

```
LINE ADDR  OBJECT  E SOURCE

0309 18E3 9F1902          BXA    CBRTB,R3       BRANCH TO CLASS PROCESSOR
0310                *
0311                * CLASS 5.     MIXED NUMBER OF OPREQ'S.
0312                *
0313 18E6 FA2F      CL5B  BDRR,R2 CL5A
0314 18E8 4707            ANDI,R3 H'07'  MASK TO OPCODE
0315 18EA 0F7967          LODA,R0 CL5TB,R3       GET NUMBER OF OPREQ'S FROM TABLE.
0316 18ED C1              STRZ   R1
0317                *
0318                * WRITE OPREQ COUNT AND EXIT TO USER
0319                *
0320 18EE 1F195E    EXIT  BCTA,UN EXIT4    GOTO USER
0321                *
0322                * RETURN FROM TEST BRANCH, IF BRANCH TAKEN
0323                *
0324 18F1 3F196F    BRCH  BSTA,UN RLADR    RESTORE LAST ADDRESS REGISTER
0325 18F4 0D17C6    BRCH1 LODA,R1 TEMP     GET OPREQ COUNT BACK AFTER TEST BRANCH
0326 18F7 7508            CPSL   WC       CLEAR PSL WC BIT
0327                *
0328                * ROUTINE TO ADD 2 OPREQ'S IF INDIRECT APPLIES.
0329                *
0330 18F9 0C17E7    CIND  LODA,R0 T3       GET SECOND BYTE OF INSTRUCTION
0331 18FC F480            TMI,R0  H'80'  TEST INDIRECT BIT
0332 18FE 1806            BCTR,0  PLS2   SET, ADD 2 OPREQ'S
0333 1900 1B6C            BCTR,UN EXIT   NOT SET, DO NOT ADD
0334                *
0335                * CLASS PROCESSOR TABLE.
0336                *
0337 1902           CBRTB EQU    $
0338                *
0339 1902 A501      PLS1  SUBI,R1 1        CLASS 0.        1 OPREQ
0340 1904 1B68            BCTR,UN EXIT
0341 1906 A502      PLS2  SUBI,R1 2        CLASS 1.        2 OPREQ'S
0342 1908 1B64            BCTR,UN EXIT
0343 190A A503            SUBI,R1 3        CLASS 2.        3 OPREQ'S + INDIRECT
0344 190C 1B6B            BCTR,UN CIND
0345 190E A504            SUBI,R1 4        CLASS 3.        4 OPREQ'S + INDIRECT
0346 1910 1B67            BCTR,UN CIND
0347 1912 1872            BCTR,EQ PLS2     CLASS4          2 OPREQS IF OPCODE ODD
0348 1914 1B6C            BCTR,UN PLS1                     1 OPREQ IF OPCODE EVEN
0349 1916 C3              STRZ   R3        CLASS 5.        MIXED NUMBER OF OPREQ'S
0350 1917 53        CL5A  RRR,R3           SHIFT OPCODE TO LOW BYTE
0351 1918 1B4C            BCTR,UN CL5B     AND LOOK UP IN TABLE
0352 191A 8501            ADDI,R1 1        CLASS 6.        2 OPREQ'S + IND IF BRANCH TAKEN
0353 191C 6404            IORI,R0 H'04'  CONVERT TO CLASS 7.
0354 191E A503            SUBI,R1 3        CLASS 7.        3 OPREQ'S + IND IF BRANCH TAKEN
0355                *
0356                * CLASS 6 AND 7.
0357                * ADD 2 OPREQ'S IF INDIRECT AND BRANCH IS TAKEN.
0358                *
0359 1920 C9D3            STRR,R1 *BRCH1+1 SAVE PRESENT NUMBER OF OPREQ'S IN TEMP
0360 1922 D5FB            WRTE,R1 OPRQCT   ALSO OUTPUT TO HARDWARE
0361 1924 F440            TMI,R0  H'40'  TEST FOR REGISTER CLASS
0362 1926 1804            BCTR,0  CL67B  IF SO, DO NOT TEST FOR UNCONDITIONAL
0363 1928 F403            TMI,R0  H'03'  IS BRANCH UNCONDITIONAL
0364 192A 184D            BCTR,0  CIND   IF SO, DO NOT TEST BRANCH
```

```
LINE ADDR  OBJECT  E SOURCE

0365 192C F4E0      CL67B  TMI,R0  H'E0'   TEST FOR BDR INST
0366 192E 1802             BCTR,0  CL67C   IF SO, DO NOT REMOVE 'SUBROUTINE' BIT
0367 1930 44DF             ANDI,R0 H'DF'   REMOVE SUBROUTINE BIT FROM OPCODE.
0368 1932 CC17C0    CL67C  STRA,R0 SCTCH   STORE IN TEST AREA
0369 1935 0E7951    MVCODE LODA,R0 BRCD,R2 GET ROM CODE
0370 1938 CE77C0           STRA,R0 SCTCH,R2  STORE IN RAM
0371 193B FA78             BDRR,R2 MVCODE  DO UNTILL ALL HAS BEEN MOVED.
0372                *
0373                *SAVE LADR
0374                *
0375 193D 0C17E8    SLAD1  LODA,R0 LADR    GET LAST ADDRESS REG
0376 1940 C8AE             STRR,R0 *RLADR+1   SAVE IT
0377 1942 0C17E9    SLAD2  LODA,R0 LADR+1
0378 1945 C8AE             STRR,R0 *RLADR1+1
0379 1947 0417             LODI,R0 <SCTCH  GET ADDRESS SCRATCH
0380 1949 C8F3             STRR,R0 *SLAD1+1
0381 194B 04C0             LODI,R0 >SCTCH
0382 194D CC9943           STRA,R0 *SLAD2+1
0383 1950 1B93             BCTR,UN *EXIT3+1      DO TEST BRANCH
0384                *
0385                *THIS IS CODE FOR TEST BRANCH
0386                *
0387 1951           BRCD   EQU     $-1
0388 1952 18F1             ACON    BRCH    ADDRESS FOR TEST BRANCH
0389 1954 1F1957           BCTA,UN EXIT2      RETURN IF BRANCH NOT TAKEN
0390                *
0391 1957 3B16      EXIT2  BSTR,UN RLADR   RESTORE LAST ADDRESS REG
0392 1959 0D17C6           LODA,R1 TEMP    GET OPREQ COUNT
0393 195C 7508             CPSL    WC      CLEAR WITH CARRY
0394 195E D5FB      EXIT4  WRTE,R1 OPRQCT  SET THE OPREQ COUNTER
0395 1960 20               EORZ    R0      CLEAR INTERUPT INHIBIT FLAG
0396 1961 CC17F1           STRA,R0 IFLG    SAVE IN INTERUPT INHIBIT FLAG
0397 1964 1F1E59    EXIT3  BCTA,UN GO
0398                *
0399                * CLASS 5 OPREQ TABLE
0400                *
0401 1967 FF        CL5TB  DATA    OVHD-1       RETC
0402 1968 FF               DATA    OVHD-1       RETE
0403 1969 FD               DATA    OVHD-3       REDE
0404 196A FE               DATA    OVHD-2       C-P PSW
0405 196B FF               DATA    OVHD-1       DAR
0406 196C FE               DATA    OVHD-2       TPSW
0407 196D FD               DATA    OVHD-3       WRTE
0408 196E FE               DATA    OVHD-2       TMI
0409                *
0410                *RLADR RESTORE LAST ADDRESS REG
0411                *
0412 196F 0C17EA    RLADR  LODA,R0 SLADR   GET SAVED LADR
0413 1972 C8CA             STRR,R0 *SLAD1+1
0414 1974 0C17EB    RLADR1 LODA,R0 SLADR+1
0415 1977 C8CA             STRR,R0 *SLAD2+1
0416 1979 17               RETC,UN
0417                *
```

```
LINE ADDR  OBJECT  E SOURCE

0419               ************************************************************************
0420               *
0421               *
0422               *BREAK POINT AND SINGLE STEP RUN TIME CODE
0423               *
0424               *
0425               *SINGLE STEP
0426               *
0427               *WHEN ENTERED AT SINGLE STEP. SINGLE STEP FLAG IS CLEARED
0428               *AND DISPLAY IS ' ADDR DD'
0429               *
0430               *
0431               *WHEN ENTERED AT BREAK POINT AND BREAK POINT IS SET AND MATCHES
0432               *BREAK POINT REGISTER. THE DISPLAY IS '-ADDR DD'
0433               *
0434               *
0435               *REGISTER USED
0436               *
0437               *R0
0438               *
0439               *SUBROUTINE CALLED
0440               *
0441               *DLSLD PREPARE BINARY DATA FOR DISPLAY
0442               *
0443               *
0444               *RAM MEMORY USED
0445               *
0446               *DISBUF DISPLAY BUFFER
0447               *BPF BREAK POINT FLAG
0448               *BPL BREAK POINT LOCATION
0449               *BPD DATA FOR BREAK POINT LOCATION
0450               *LADR COPY OF LAST ADDDRESS REGISTER
0451               *SSF SINGLE STEPFLAG
0452               *
0453               ************************************************************************
0454               *
0455 197A 0C17E8  BRKPT  LODA,R0 LADR    GET LAST ADDRESS REGISTER
0456 197D 0D17E9  BRK3   LODA,R1 LADR+1
0457 1980 7709           PPSL    C+WC    SET CARRY AND WITH CARRY
0458 1982 A501           SUBI,R1 1       DECREMENT LAST ADDRESS REG
0459 1984 A400           SUBI,R0 0       SO CAN COMPARE TO BREAK POINT REGISTER
0460 1986 447F           ANDI,R0 H'7F    MASK OFF UNUSED BIT
0461 1988 7509           CPSL    C+WC    CLEAR CARRY AND WITH CARRY
0462 198A E8AF        BRK2   COMR,R0 *BRKPT2+1 COMPARE WITH BPL
0463 198C 9C185A      BRK1   BCFA,EQ MON     NO COMPARE
0464 198F E9AF           COMR,R1 *BRKPT1+1 COMPARE WITH BPL+1
0465 1991 98FA           BCFR,EQ *BRK1+1   NO COMPARE
0466 1993 C8E6           STRR,R0 *BRKPT+1        IF COMPARE UP DATE PC
0467 1995 C9E7           STRR,R1 *BRK3+1
0468 1997 0C17CC         LODA,R0 BPD     IF COMAPRE CLEAR BREAK POINT
0469 199A CC97CD         STRA,R0 *BPL
0470 199D EC97CD         COMA,R0 *BPL    ERROR CHECK OF DATA WRITTEN
0471 19A0 1804           BCTR,EQ BRK0    DATA STORED OK
0472 19A2 0701           LODI,R3 1       BREAK POINT NOT CLEARED OK
0473 19A4 9BE8           ZBRR    *ERR
```

```
LINE ADDR  OBJECT  E SOURCE

0474 19A6 E440      BRK0   COMI,R0 H'40'   HALT INSTRUCTION OPCODE
0475 19A8 1808             BCTR,EQ BRKPT9  IF HALT DON'T DO HIDDEN SINGLE STEP
0476 19AA 0480             LODI,R0 H'80'   SET FLAG FOR HIDDEN SINGLE STEP
0477 19AC CC17CF    BRKPT3 STRA,R0 BPF     SET FLAG IN BREAK POINT
0478 19AF 1F18B4           BCTA,UN SSTEP   EXECUTE ONE USER INSTRUCTION
0479 19B2 047F      BRKPT9 LODI,R0 127     SET BREAK POINT FLAG
0480 19B4 C8F7             STRR,R0 *BRKPT3+1
0481 19B6 0419             LODI,R0 H'19'   DASH SYMBOL FOR BREAK POINT
0482 19B8 C89A             STRR,R0 *BRKPT8+1  SET THE DASH SYMBOL IN DISBUF
0483 19BA 0C17CD    BRKPT2 LODA,R0 BPL     GET BREAK POINT ADDRESS
0484 19BD 3B32             BSTR,UN BRKPT7  SET THE DISPLAY
0485 19BF 0C17CE    BRKPT1 LODA,R0 BPL+1
0486 19C2 3B36             BSTR,UN BRKPT6
0487 19C4 0C97CD           LODA,R0 *BPL    GET INSTRUCTION OPCODE
0488 19C7 1B1A             BCTR,UN BRKPT5
0489                *
0490                *ENTRY POINT FOR SINGLE STEP
0491                *
0492 19C9 20        SGLSTP EORZ    R0      GET A 0
0493 19CA CC17D0           STRA,R0 SSF     CLEAR SINGLE STEP FLAG
0494 19CD 08DE             LODR,R0 *BRKPT3+1       CHECK BREAK POINT FLAG
0495 19CF 1A61             BCTR,NG BRKPT9  DID A HIDDEN SINGLE STEP
0496                *                      DISPLAY THE BREAK POINT
0497 19D1 0417      SGLST9 LODI,R0 H'17'   BLANK SYMBOL
0498 19D3 CC17D1    BRKPT8 STRA,R0 DISBUF  SET DISPLAY BUFFER
0499 19D6 0C17E8           LODA,R0 LADR    GET ADDRESS
0500 19D9 3B16             BSTR,UN BRKPT7  SET THE DISPLAY
0501 19DB 0C17E9           LODA,R0 LADR+1
0502 19DE 3B1A             BSTR,UN BRKPT6  SET THE DISPLAY
0503 19E0 0C97E8           LODA,R0 *LADR   GET INSTRUCTION DATA
0504 19E3 3B03      BRKPT5  BSTR,UN BRKPTI  SET UP DISPLAY
0505 19E5 1F187A           BCTA,UN MON4     GOTO MONITOR
0506                *
0507                *SET UP DISBUF 6&7
0508                *
0509 19E8 BBF4      BRKPTI ZBSR    *DISLSD CONVERT TO BIN FOR DISPLAY
0510 19EA CC17D7           STRA,R0 DISBUF+6
0511 19ED CD17D8           STRA,R1 DISBUF+7
0512 19F0 17               RETC,UN
0513                *
0514                *SET UP DISBUF 1&2
0515                *
0516 19F1 BBF4      BRKPT7 ZBSR    *DISLSD CONVERT BIN TO DISPLAY
0517 19F3 CC17D2           STRA,R0 DISBUF+1
0518 19F6 CD17D3           STRA,R1 DISBUF+2
0519 19F9 17               RETC,UN
0520                *
0521                *SETUP DISBUF 3&4
0522                *
0523 19FA BBF4      BRKPT6 ZBSR    *DISLSD CONVERT BIN TO DISPLAY
0524 19FC CC17D4           STRA,R0 DISBUF+3        STORE DATA
0525 19FF CD17D5           STRA,R1 DISBUF+4
0526 1A02 0417             LODI,R0 H'17'   BLANK SYMBOL
0527 1A04 CC17D6           STRA,R0 DISBUF+5
0528 1A07 17               RETC,UN
```

```
LINE ADDR  OBJECT  E SOURCE

0530               ******************************************************
0531               *
0532               *
0533               *DISPLAY AND ALTER MEMORY ROUTINE
0534               *PATCH MEMORY ROUTINE
0535               *
0536               *
0537               *REGISTERS USED
0538               *
0539               *R0 SCRATCH
0540               *R1 SCRATCH
0541               *R2 SCRATCH
0542               *R3 SCRATCH
0543               *
0544               *SUBROUTINES CALLED
0545               *
0546               *GAD GET ADDRESS PARAMETER
0547               *GNP GET NUMBER PARAMETER
0548               *ROT ROTATE R0 1 NIBBLE LEFT
0549               *BRKPT4        SETUP DISPLAY 6&7
0550               *BRKPT6        SETUP DISPLAY 3&4
0551               *BRKPT7        SETUP DISPLAY 1&2
0552               *
0553               *RAM MEMORY USED
0554               *
0555               *MEM INDIRECT ADDRESS MEMORY POINTER
0556               *ALTF ALTER FLAG = 1 FOR DISPLAY AND ALTER
0557               *                  3 OR 5 FOR PATCH
0558               *
0559               ******************************************************
0560               *
0561               *
0562               *ENTRY POINT FOR PATCH COMMAND
0563               *
0564 1A08 0403    PTCH   LODI,R0 3       SET ALTER FLAG TO PATCH
0565 1A0A 1B02           BCTR,UN ALTER5
0566               *
0567               *ENTRY POINT FOR DISPLAY AND ALTER COMMAND
0568               *
0569 1A0C 0401    ALTER  LODI,R0 1       SET ALTER FLAG TO ALTER
0570 1A0E C8A4    ALTER5 STRR,R0 *ALTER1+1       STORE IN ALTF
0571 1A10 3F1B04         BSTA,UN GAD     DISPLAY AD= AND WAIT TILL DIGITS ENTERED
0572 1A13 E687           COMI,R2 H'87'   ENTR/NXT?
0573 1A15 9C187D         BCFA,EQ MON2    NEW FUNCTION ABORT ALTER COMMAND
0574 1A18 5B0E           BRNR,R3 ALTER4  NO ADDRESS ENTERED CONTINUE FROM LAST LOCATION
0575 1A1A C88D           STRR,R0 *ALTER4+1 MEM+1   SAVE ADDRESS DATA
0576 1A1C C981           STRR,R1 *AL1+1
0577 1A1E 0E17DD  AL1    LODA,R2 MEM     GET DATA
0578 1A21 0717           LODI,R3 H'17'      BLANK
0579 1A23 CF17D8         STRA,R3 DISBUF+7       CLEAR DISPLAY
0580 1A26 1B05           BCTR,UN ALTER2  SET UP DISPLAY
0581               *
0582               *NO ADDRESS CONTINUE FROM LAST ADDRESS
0583               *
0584 1A28 0C17DE  ALTER4 LODA,R0 MEM+1   GET ADDRESS
```

```
LINE ADDR  OBJECT  E SOURCE

0585 1A2B 0AF2           LODR,R2 *AL1+1 MEM
0586                *
0587                *UPDATE THE DISPLAY
0588                *
0589 1A2D 3B4B      ALTER2 BSTR,UN BRKPT6  SET UP ADDRESS DISPLAY
0590 1A2F 02             LODZ   R2      GET MSD
0591 1A30 3F19F1         BSTA,UN BRKPT7  SET UP DISPLAY
0592                *
0593                *
0594 1A33 0C17EF    ALTER1 LODA,R0 ALTF    CHECK ALTER FLAG
0595 1A36 E401           COMI,R0 1       PATCH COMMAND
0596 1A38 1807           BCTR,EQ ALTER8  NOT PATCH
0597 1A3A 0417           LODI,R0 H'17'   BLANK CHAR
0598 1A3C CC17D7         STRA,R0 DISBUF+6
0599 1A3F 1B05           BCTR,UN ALTER9  PATCH COMMAND
0600                *
0601 1A41 0C97DD    ALTER8 LODA,R0 *MEM    GET THE DATA
0602 1A44 BBEA           ZBSR  *BRKPT4  SET UP DATA VALUE DISPLAY
0603 1A46 08EC      ALTER9 LODR,R0 *ALTER1+1    SET FLAG TO SINGLE BYTE DATA
0604 1A48 BBFC           ZBSR *GNPA    DISPLAY BUFFER AND WAIT FOR NEW ENTRY
0605 1A4A 5B0C           BRNR,R3 ALTER3   NO DATA
0606 1A4C CC97DD         STRA,R0 *MEM    CHANGE DATA IN LOCATION
0607 1A4F EC97DD         COMA,R0 *MEM    CHECK DATA STORED OK
0608 1A52 1804           BCTR,EQ ALTER3  DATA STORED OK
0609 1A54 0703           LODI,R3 3       ALTER OR PATCH WRITE ERROR
0610 1A56 9BE8           ZBRR    *ERR    GOTO ERROR
0611                *
0612                *EXIT FROM COMMAND
0613                *
0614 1A58 08DA      ALTER3 LODR,R0 *ALTER1+1    EXIT FROM ALTER OR PATCH
0615 1A5A E401           COMI,R0 1       IS IT PATCH
0616 1A5C 9807           BCFR,EQ ALTER6  IF YES TAKE THIS BRANCH
0617 1A5E E687           COMI,R2 H'87'   IS IT ALTER NEXT KEY FUNCTION?
0618 1A60 9C187D    AL2    BCFA,EQ MON2    GO TO MONITOR NEW COMMAND
0619 1A63 1B0B           BCTR,UN ALTER7   GO UPDATE THE DISPLAY
0620                *
0621                *EXIT FROM PATCH
0622                *
0623 1A65 E60F      ALTER6 COMI,R2 H'0F'   WAS LAST KEY FUNCTION KEY
0624 1A67 19F8           BCTR,GT *AL2+1 MON2    FUNCTION KEY WAS LAST GO TO MONITOR
0625 1A69 0405           LODI,R0 5       RETURN ON SECOND DIGIT FLAG
0626 1A6B C8C7           STRR,R0 *ALTER1+1       SAVE IN ALTF
0627 1A6D CE17D8         STRA,R2 DISBUF+7        SET DISPLAY
0628                *
0629                *INCREMENT INDIRECT ADDRESS
0630                *
0631 1A70 3F1C55    ALTER7 BSTA,UN INK     INCREMENT THE ADDRESS
0632 1A73 1F1A2D         BCTA,UN ALTER2
0633                *
0634                *PREPARE BIN DATA FOR DISPLAY
0635                *
0636 1A76 C1        DISLSI STRZ    R1      SAVE NUMBER IN R0
0637 1A77 44F0           ANDI,R0 H'F0'   MASK FOR MSD
0638 1A79 450F           ANDI,R1 H'0F'   MASK FOR LSD
0639 1A7B BBF6           ZBSR    *ROT    ROTATE A NIBBLE
0640 1A7D 17             RETC,UN
```

```
LINE ADDR  OBJECT  E SOURCE

0642               ************************************************************
0643               *
0644               *
0645               *DISPLAY AND ALTER REGISTERS COMMAND
0646               *
0647               *THE DISPLAY AND ALTER REGISTERS COMMAND ALLOWS
0648               *THE USER TO EXAMINE AND ALTER R0,R1,R2,R3,R1',R2',R3',PSU,PSL,PC
0649               *
0650               *THIS COMMAND ALSO PROVIDES ENTRY POINT TO ALTERNATE FUNCTIONS
0651               *REG 9 NOT DEFINED
0652               *REG A ADJUST CASSETTE COMMAND
0653               *REG B NOT DEFINED
0654               *REG D NOT DEFINED
0655               *REG E NOT DEFINED
0656               *REG F ENTER THE FAST PATCH MODE
0657               *
0658               *REGISTERS USED
0659               *
0660               *R0 SCRATCH
0661               *R1 SCRATCH
0662               *R2 SCRATCH
0663               *R3 SCRATCH
0664               *
0665               *SUBROUTINES CALLED
0666               *
0667               *MOV   MOVE DATA TO DISBUF
0668               *GNP   GET NUMERIC PARAMETERS
0669               *ROT   ROTATE A NIBBLE
0670               *GNPA  DISPLAY AND GET NUMERIC PARAMETERS
0671               *BRKPT4        SET DISPLAY 6&7
0672               *SCBP2 SET DISPLAY 4&5
0673               *
0674               *RAM MEMORY USED
0675               *
0676               *DISBUF        DISPLAY BUFFER
0677               *UREG  USER REGISTERS
0678               *LADR  LAST ADDRESS REGISTER PC COUNTER
0679               *T2  TEMP REGISTER
0680               *
0681               ************************************************************
0682 1A7E 051F     REG    LODI,R1 <REQ-1  GET ADDRESS OF R= DISPLAY
0683 1A80 06A4            LODI,R2 >REQ-1
0684 1A82 BBFE            ZBSR *MOV     MOVE DATA TO DISBUF
0685 1A84 20              EORZ    R0      SET FLAG TO RETURN AFTER KEY PRESSED
0686 1A85 BBEC            ZBSR    *DISPLY
0687               *
0688 1A87 F480            TMI,R0  H'80'   SEE IF FUNCTION
0689 1A89 1882            BCTR,EQ *REG14+1 MON2    GOTO MONITOR
0690 1A8B E409            COMI,R0 9       CHECK THE COMMAND
0691 1A8D 1E1AC2          BCTA,LT REG2    DISPLAY AND ALTER REGISTERS R0 THRU PSL
0692 1A90 E40A            COMI,R0 H'0A'   IS IT ADJUST CASSETTE COMMAND
0693 1A92 1C1F32          BCTA,EQ TCAS    TEST CASSETTE
0694 1A95 E40C            COMI,R0 H'0C'   IS IT DISPLAY AND ALTER PC
0695 1A97 1807            BCTR,EQ REG3    DISPLAY AND ALTER PC
0696 1A99 E40F            COMI,R0 H'0F'   IS IT THE PATCH COMMAND
```

```
LINE ADDR  OBJECT  E SOURCE

0697 1A9B 1C1A08          BCTA,EQ PTCH    DO THE PATCH COMMAND
0698 1A9E 1B5E            BCTR,UN REG     NOT DEFINED TRY AGAIN
0699                 *
0700                 *DISPLAY AND ALTER PROGRAM COUNTER
0701                 *
0702 1AA0 051F       REG3   LODI,R1 <PCEQ-1 GET ADDRESS OF PC EQUALS DISPLAY
0703 1AA2 06AC           LODI,R2 >PCEQ-1
0704 1AA4 BBFE           ZBSR    *MOV    MOVE DATA TO DISBUF
0705 1AA6 088E           LODR,R0 *REG4+1  GET CURRENT PC ADDRESS
0706 1AA8 BBEA           ZBSR  *BRKPT4  SET UP DISPLAY
0707 1AAA 0C17E8      REG11  LODA,R0 LADR    GET MSB OF CURRENT PC
0708 1AAD 3F1D8A          BSTA,UN SCBP2   SET UP DISPLAY
0709 1AB0 20              EORZ    R0      SET FLAG TO DOUBLE BYTE
0710 1AB1 BBFC            ZBSR *GNPA    DISPLAY ADDRESS AND WAIT FOR ENTRY
0711 1AB3 5B05            BRNR,R3 REG5    DON'T CHANGE DATA
0712 1AB5 CC17E9      REG4   STRA,R0 LADR+1  UP DATE THE PC SAVE LSB
0713 1AB8 C9F1            STRR,R1 *REG11+1 SAVE MSB OF PC
0714 1ABA E687        REG5   COMI,R2 H'87'   ENTR/NXT TERMINATION
0715 1ABC 9C187D      REG14  BCFA,EQ MON2    IF NOT NEW FUNCTION EXIT
0716 1ABF 1F1A7E          BCTA,UN REG     GO ASK FOR NEW REGISTER
0717                 *
0718                 *DISPLAY AND ALTER REGISTERS
0719                 *
0720 1AC2 CC17E6      REG2   STRA,R0 T2      SAVE IT
0721 1AC5 C3              STRZ    R3      SAVE R0 TO USE AS INDEX
0722 1AC6 0510            LODI,R1 H'10'   P CHAR
0723 1AC8 E707            COMI,R3 7       IS IT PSU
0724 1ACA 1A0A            BCTR,LT REG8    NOT PSU PSL
0725 1ACC 1904            BCTR,GT REG10   NOT PSU
0726 1ACE 0412            LODI,R0 H'12'   CHAR U
0727 1AD0 1B06            BCTR,UN REG12    GO DISPLAY
0728                 *
0729 1AD2 0411        REG10  LODI,R0 H'11'   CHAR L
0730 1AD4 1B02            BCTR,UN REG12    GO DISPLAY
0731                 *
0732 1AD6 0513        REG8   LODI,R1 H'13'      CHAR R
0733 1AD8 CD17D3      REG12  STRA,R1 DISBUF+2       SET DISPLAY RN=
0734 1ADB CC17D4      REG9   STRA,R0 DISBUF+3       SET UP DISPLAY
0735 1ADE 0F77F2          LODA,R0 UREG,R3 GET REGISTER CONTENT
0736 1AE1 BBEA            ZBSR  *BRKPT4  SET UP DISPLAY
0737 1AE3 0401            LODI,R0 1       SET FLAG TO SINGLE BYTE
0738 1AE5 BBFC            ZBSR *GNPA    DISPLAY REG CONTENT AND WAIT FOR ENTRY
0739 1AE7 5B0C            BRNR,R3 REG7    NO DATA TERMINATE
0740 1AE9 0BD8        REG6   LODR,R3 *REG2+1       GET THE INDEX VALUE
0741 1AEB CF77F2          STRA,R0 UREG,R3 PUT NEW VALUE IN REGISTER
0742 1AEE E708            COMI,R3 8       IS IT PSL?
0743 1AF0 9803            BCFR,EQ REG7    NO CHECK TERMINATION
0744 1AF2 CC17FC          STRA,R0 UREG+10 SAVE FOR RESTORE OF PSL
0745 1AF5 E687        REG7   COMI,R2 H'87'   CHECK TERMINATION
0746 1AF7 98C4            BCFR,EQ *REG14+1 MON2    NEW FUNCTION
0747 1AF9 03              LODZ    R3      INCREMENT INDEX VALUE
0748 1AFA D800            BIRR,R0 $+2     INCREMENT REGISTER COUNT
0749 1AFC E408            COMI,R0 8       ROLL OVER?
0750 1AFE 9D1AC2      REG13  BCFA,GT REG2    NO UP DATE DISPLAY
0751 1B01 20              EORZ    R0      GET A 0 GO TO R0
0752 1B02 1BFB            BCTR,UN *REG13+1    UPDATE DISPLAY
```

```
LINE ADDR  OBJECT  E SOURCE

0754               ********************************************************
0755               *
0756               *
0757               *GET NUMERIC PARAMETERS
0758               *
0759               *
0760               *THIS ROUTINE GETS EITHER 2 OR 4 DIGIT NUMERIC PARAMETERS
0761               *
0762               *INPUT PARAMETERS
0763               *
0764               *R0 CONTAINS INPUT PARAMETER
0765               *
0766               *BIT0 = 0 DOUBLE BYTE
0767               *BIT0 = 1 SINGLE BYTE DATA TO BE RETURNED
0768               *BIT1 = 0 REQUIRES FUNCTION KEY DEPRESSION TO EXIT
0769               *BIT1 = 1 WHEN SET WITH BIT0 EXIT IS AFTER ENTRY OF THIRD DIGIT
0770               *         OF SINGLE BYTE DATA
0771               *BIT2 = 1 WHEN SET WITH BIT0 EXIT IS AFTER SECOND DIGIT
0772               *         OF SINGLE BYTE DATA
0773               *
0774               *SINGLE BYTE DATA USES DISPLAY BUFFER 5 THRU 7
0775               *DOUBLE BYTE DATA USES DISPLAY BUFFER 4 THRU 7
0776               *OTHER DIGITS OF DISBUF MUST BE INITIALIZED ON ENTRY
0777               *
0778               *RETURNS WHEN FUNCTION KEY DEPRESSED
0779               *
0780               *OUTPUT PARAMETERS
0781               *
0782               *R0 = LSB OF DOUBLE BYTE DATA OR SINGLE BYTE DATA
0783               *R1 = MSB OF DOUBLE BYTE DATA OR 0 FOR SINGLE BYTE DATA
0784               *R2 = FUNCTION KEY PRESSED CODE
0785               *R3 = 0 DATA RETURNED IN R0(LSB), R1(MSB)
0786               *R3 = NOT 0 NO DATA RETURNED R0,R1 = 0
0787               *
0788               *REGISTERS USED
0789               *
0790               *R0 SCRATCH
0791               *R1 SCRATCH
0792               *R2 SCRATCH
0793               *R3 SCRATCH
0794               *
0795               *SUBROUTINES CALLED
0796               *
0797               *DISPLY        DISPLAY AND READ KEY BOARD
0798               *CLR BLANK DIGIT DISPLAY
0799               *
0800               *RAM MEMORY USED
0801               *
0802               *T1 SAVE ENTRY FLAG
0803               *DISPLY 4 THRU 7
0804               *
0805               ********************************************************
0806               *
0807               *
0808               *DISPLAY AD= AND GET DATA
```

```
LINE ADDR  OBJECT  E SOURCE

0809                    *
0810 1B04 051F          GAD    LODI,R1 <ADR-1  GET ADDRESS OF AD= DISPALY
0811 1B06 068C                 LODI,R2 >ADR-1
0812 1B08 BBFE                 ZBSR *MOV     MOVE DATA TO DISBUF
0813 1B0A 20                   EORZ    R0      SET FLAG TO DOUBLE BYTE DATA
0814 1B0B 1B2E                 BCTR,UN GNPI     GET THE ADDRESS DATA
0815                    *
0816                    *THIS ROUTINE CLEARS DIGIT DISPLAY
0817                    *
0818 1B0D 0517          CLR    LODI,R1 H'17'   BLANK SYMBOL
0819 1B0F F401                 TMI,R0  1       SINGLE BYTE?
0820 1B11 1803                 BCTR,EQ CLR1    ONE BYTE DATA
0821 1B13 CD17D5               STRA,R1 DISBUF+4        INITIALIZE DISPLAY TO BLANK
0822 1B16 CD17D6        CLR1   STRA,R1 DISBUF+5        GET HERE FOR ONE BYTE DATA
0823 1B19 CD17D7               STRA,R1 DISBUF+6
0824 1B1C CD17D8               STRA,R1 DISBUF+7
0825 1B1F 17                   RETC,UN
0826                    *
0827                    *THIS ENTRY POINT ALLOWS DISPLAY OF DATA IN DISPLAY
0828                    *BUFFER 4 THRU 7
0829                    *
0830 1B20 C8A6          GNPAI  STRR,R0 *GNP12+1        SAVE INPUT FLAG IN T1
0831 1B22 0480                 LODI,R0 H'80'   TURN ON DECIMAL POINT FOR ENTRY
0832 1B24 BBEC                 ZBSR    *DISPLY DISPLAY MESSAGE AND READ KEY BOARD
0833 1B26 08A0                 LODR,R0 *GNP12+1      GET INPUT PARAMETER
0834 1B28 F680                 TMI,R2  H'80'   FUNCTION KEY?
0835 1B2A 9809                 BCFR,EQ GNP13   FIRST CHAR IS COMMAND TERMINATE
0836 1B2C E687                 COMI,R2 H'87'   ENTR/NXT?
0837 1B2E 1802                 BCTR,EQ $+4
0838 1B30 3B5B                 BSTR,UN CLR     CLEAR DISPLAY
0839 1B32 1F1B74               BCTA,UN GNP4
0840 1B35 F404          GNP13  TMI,R0  4       PATCH COMMAND RETURN SECOND DIGIT?
0841 1B37 3A54                 BSTR,NG CLR     CLEAR DISPLAY
0842 1B39 1B0C                 BCTR,UN GNP12
0843                    *
0844                    *THIS ENTRY POINT CLEARS DISPLAY AND WAITS FOR ENTRY
0845                    *
0846 1B3B C88B          GNPI   STRR,R0 *GNP12+1        SAVE INPUT FLAG IN T1
0847 1B3D 3B4E          GNP11  BSTR,UN CLR     CLEAR DISPLAY
0848 1B3F 0480          GNP2   LODI,R0 H'80'   TURN ON DECIMAL POINT FOR ENTRY
0849 1B41 BBEC                 ZBSR *DISPLY  DISPLAY MESSAGE AND READ KEY BOARD
0850 1B43 F680          GNP5   TMI,R2  H'80'   FUNCTION KEY PRESSED?
0851 1B45 182D                 BCTR,EQ GNP4    GO TERMINATE
0852                    *
0853                    *MOVE DISPLAY 1 DIGIT LEFT
0854                    *
0855 1B47 0C17E5        GNP12  LODA,R0 T1      GET INPUT PARAMETER
0856 1B4A F483                 TMI,R0  H'83'   THIRD DIGIT *EXIT ON THIRD ENTRY*SINGLE BYTE
0857 1B4C 1826                 BCTR,EQ GNP4    GO TERMINATE
0858 1B4E F425                 TMI,R0  H'25'   2ND DIGIT*EXIT ON 2ND DIGIT*SINGLE BYTE
0859 1B50 1822                 BCTR,EQ GNP4    GO TERMINATE
0860 1B52 F401                 TMI,R0  1       SINGLE BYTE DATA?
0861 1B54 1808                 BCTR,EQ GNP3    ONLY TWO DIGITS
0862 1B56 0D17D6        GN1    LODA,R1  DISBUF+5       GET DIGIT
0863 1B59 CD17D5        GN2    STRA,R1 DISBUF+4        SHIFT IT
0864 1B5C 0D17D7        GN3    LODA,R1 DISBUF+6        GET DIGIT
```

```
LINE ADDR  OBJECT  E SOURCE

0865 1B5F C9F6           STRR,R1 *GN1+1       SHIFT IT
0866 1B61 0D17D8   GNP3  LODA,R1 DISBUF+7       GET DIGIT
0867 1B64 C9F7           STRR,R1 *GN3+1       SHIFT IT
0868 1B66 CAFA           STRR,R2 *GNP3+1       ENTER NEW DIGIT
0869 1B68 08DE           LODR,R0 *GNP12+1     GET INPUT PARAMETER
0870 1B6A 7508           CPSL    WC      CLEAR WITH CARRY
0871 1B6C 8440           ADDI,R0 H'40'   SET BEEN HERE ONCE FLAG
0872 1B6E 6420           IORI,R0 H'20'   SET SECOND DIGIT FLAG
0873 1B70 C8D6     GN4   STRR,R0 *GNP12+1     RESTORE THE FLAG
0874 1B72 1B4B           BCTR,UN GNP2    GET NEXT ENTRY
0875               *
0876               *SET UP DATA TO BE RETURNED
0877               *
0878 1B74 20       GNP4  EORZ    R0      GET A 0
0879 1B75 C1             STRZ    R1      CLEAR R1 DATA
0880 1B76 C3             STRZ    R3      CLEAR R3
0881 1B77 08CF           LODR,R0 *GNP12+1     GET INPUT PARAMETER
0882 1B79 F401           TMI,R0  1       CHECK FOR SINGLE BYTE
0883 1B7B 1812           BCTR,EQ GNP7    IF EQ ONLY 1 DIGIT
0884 1B7D 0C9B5A         LODA,R0 *GN2+1 DISBUF+4       GET MSD OF MSB
0885 1B80 E410           COMI,R0 H'10'   SEE IF HEX DIGIT
0886 1B82 9A03           BCFR,LT GNP6    IF NOT SKIP TO NEXT DIGIT
0887 1B84 3B1F           BSTR,UN ROTI     ROTATE NIBBLE
0888 1B86 C1             STRZ    R1      SAVE IN R1
0889 1B87 08CE     GNP6  LODR,R0 *GN1+1 DISBUF+5       GET LSD OF MSB
0890 1B89 E410           COMI,R0 H'10'   SEE IF HEX DIGIT
0891 1B8B 9A02           BCFR,LT GNP7    IF NOT SKIP TO NEXT DIGIT
0892 1B8D 61             IORZ    R1      INCLUSIVE OR MSD AND LSD OF MSB
0893 1B8E C1             STRZ    R1      SAVE IN R1
0894 1B8F 08CC     GNP7  LODR,R0 *GN3+1 DISBUF+6       GET MSD OF LSB
0895 1B91 E410           COMI,R0 H'10'   SEE IF HEX DIGIT
0896 1B93 9A03           BCFR,LT GNP8    IF NOT SKIP TO NEXT DIGIT
0897 1B95 3B0E           BSTR,UN ROTI     ROTATE THE NIBBLE
0898 1B97 C3             STRZ    R3      SAVE IN R3
0899 1B98 08C8     GNP8  LODR,R0 *GNP3+1 DISBUF+7       GET LSD OF LSB
0900 1B9A E410           COMI,R0 H'10'   SEE IF HEX DIGIT
0901 1B9C 9A04           BCFR,LT GNP9    IF NOT RETURN
0902 1B9E 63             IORZ    R3      INCLUSIVE OR MSD WITH LSD OF LSB
0903 1B9F 0700           LODI,R3 0     SET DATA IN R0,R1 FLAG
0904 1BA1 17             RETC,UN
0905 1BA2 077F     GNP9  LODI,R3 127     NO DATA
0906 1BA4 17             RETC,UN
0907               *
0908               *THIS ROUTINE ROTATES A NIBBLE 4 BITS LEFT
0909               *
0910 1BA5 7508     ROTI   CPSL    WC      CLEAR WITH CARRY
0911 1BA7 D0             RRL,R0
0912 1BA8 D0             RRL,R0
0913 1BA9 D0             RRL,R0
0914 1BAA D0             RRL,R0
0915 1BAB 17             RETC,UN
```

```
LINE ADDR  OBJECT  E SOURCE

0917               ************************************************************
0918               *
0919               *
0920               *READ CASSETTE COMMAND
0921               *
0922               *
0923               *
0924               *THIS IS THE HEX OBJECT LOADER
0925               *
0926               *THIS ROUTINE REQUESTS A FILE ID AND THEN LOADS 2650 HEX OBJECT MODULES
0927               *INTO MEMORY
0928               *
0929               *REGISTERS USED
0930               *
0931               *ALL
0932               *
0933               *SUBROUTINES CALLED
0934               *
0935               *IN CASSETTE INPUT ROUTINE
0936               *MOV   MOVE DATA TO DISPLAY BUFFER
0937               *GNP   GET NUMERIC PARAMETERS
0938               *
0939               ************************************************************************
0940               *
0941               *
0942 1BAC 051F    RCAS   LODI,R1 <FEQ-1  GET ADDRESS OF F= DISPLAY
0943 1BAE 06B4           LODI,R2 >FEQ-1
0944 1BB0 BBFE           ZBSR *MOV     MOVE DATA TO DISBUF
0945 1BB2 0401           LODI,R0 1       SET FLAG FOR SINGLE BYTE
0946 1BB4 BBFA           ZBSR    *GNP    GET THE FILE ID
0947 1BB6 180A           BCTR,EQ RCAS1   FILE ID SPECIFIED
0948 1BB8 E687           COMI,R2 H'87'   ENTR/NXT KEY?
0949 1BBA 988C           BCFR,EQ *RCAS4+1        GODO NEW FUNCTION
0950 1BBC 047F           LODI,R0 127     SET FILE ID FLAG TO FILE ID FOUND
0951 1BBE C8A4           STRR,R0 *RCAS5+1        STORE IN FILE ID FLAG
0952 1BC0 1B24           BCTR,UN LOAD
0953               *
0954               *
0955               *FILE ID SPECIFIED
0956               *
0957 1BC2 CC17E0  RCAS1  STRA,R0 FID+1   SAVE FILE ID
0958 1BC5 E687           COMI,R2 H'87'   ENTR/NXT KEY?
0959 1BC7 9C187D  RCAS4  BCFA,EQ MON2    GO DO NEW FUNCTION
0960 1BCA 20             EORZ    R0      SET FILE ID TO ID NOT FOUND
0961 1BCB C897           STRR,R0 *RCAS5+1        STORE IN FILE ID FLAG
0962 1BCD 75FD           CPSL    H'FD'     CLEAR PSL
0963 1BCF BBEE    RCAS2  ZBSR    *IN     LOOK FOR BEGINNING OF FILE
0964 1BD1 E416           COMI,R0 H'16'   BEGINNING OF FILE CHAR?
0965 1BD3 987A           BCFR,EQ RCAS2   LOOP TILL FIND BEGIN OF FILE
0966 1BD5 3F1C28         BSTA,UN BIN     GET THE FILE ID
0967 1BD8 E9E9           COMR,R1 *RCAS1+1   CHECK FILE ID FOR MATCH
0968 1BDA 1805           BCTR,EQ RCAS3   FOUND A MATCH
0969 1BDC 20             EORZ    R0      GET A 0
0970 1BDD C885           STRR,R0 *RCAS5+1 NO MATCH SAVE IN FID FLAG
0971 1BDF 1B05           BCTR,UN LOAD
```

```
LINE ADDR  OBJECT  E SOURCE

0972 1BE1 047F      RCAS3  LODI,R0 127    SET FLAG TO FILE IS MATCH
0973 1BE3 CC17DF    RCAS5  STRA,R0 FID    FILE ID FOUND
0974 1BE6 75FD      LOAD   CPSL   H'FD'   CLEAR PSL
0975 1BE8 BBEE             ZBSR   *IN    GET A CHAR
0976 1BEA E43A             COMI,R0 A':'  START OF LINE CHAR?
0977 1BEC 9878             BCFR,EQ LOAD  LOOP TILL FIND START FO RECORD
0978 1BEE 20               EORZ   R0     GET A 0
0979 1BEF CC17E1           STRA,R0 BCC   PRESET BCC
0980 1BF2 3B34             BSTR,UN BIN   INPUT A BYTE OF DATA
0981 1BF4 CD17DD           STRA,R1 MEM HI ADDR
0982 1BF7 3B2F             BSTR,UN BIN INPUT A BYTE OF DATA
0983 1BF9 CD17DE           STRA,R1 MEM+1 LO ADDR
0984 1BFC 3B2A             BSTR,UN BIN INPUT A BYTE OF DATA
0985 1BFE 01               LODZ   R1
0986 1BFF 1C1C42           BCTA,EQ LOAD1  GO TO START OF PROGRAM IF BYTE COUNT 0
0987 1C02 C3               STRZ   R3     SAVE BYTE COUNT
0988 1C03 08DF             LODR,R0 *LOAD-2 GET FILE ID FLAG
0989 1C05 185F             BCTR,EQ LOAD  FILE ID NOT FOUND SKIP TO END OF FILE
0990 1C07 3B1F             BSTR,UN BIN INPUT A BYTE OF DATA
0991 1C09 1804             BCTR,EQ BLOA  BCC OK READ THE RECORD
0992 1C0B 0704      BLOA1  LODI,R3 4      BCC ERROR
0993 1C0D 9BE8             ZBRR   *ERR   GOTO ERROR
0994 1C0F 3B17      BLOA   BSTR,UN BIN INPUT A BYTE OF DATA
0995 1C11 CD97DD           STRA,R1 *MEM STORE DATA IN MEMORY
0996 1C14 ED97DD           COMA,R1 *MEM  DO THE ERROR CHECK
0997 1C17 1804             BCTR,EQ BLOA2  DATA STORED OK
0998 1C19 0705             LODI,R3 5      READ CASSETTE MEMORY WRITE ERROR
0999 1C1B 9BE8             ZBRR   *ERR   GOTO ERROR
1000 1C1D 3B36      BLOA2  BSTR,UN INK INCREMENT POINTER MEM
1001 1C1F FB6E             BDRR,R3 BLOA LOOP TILL DONE
1002 1C21 3B05             BSTR,UN BIN INPUT A BYTE OF DATA
1003 1C23 9866             BCFR,EQ BLOA1 BCC ERROR
1004 1C25 1F1BE6           BCTA,UN LOAD
1005                *
1006                * INPUT A PAIR OF HEX ASCII CHAR
1007                *CONVERT TO BINARY
1008                *OUTPUT IS IN R1
1009                *CALCULATE BCC ON DATA
1010                *
1011 1C28 BBEE      BIN    ZBSR   *IN INPUT A CHAR
1012 1C2A 7509             CPSL   C+WC   CLEAR CARRY AND WITH CARRY
1013 1C2C 3B36      BIN1   BSTR,UN AHO3   LOOK UP VALUE
1014 1C2E 02               LODZ   R2     PUT VALUE IN R0
1015 1C2F BBF6             ZBSR   *ROT   ROTATE VALUE
1016 1C31 C1               STRZ   R1     SAVE VALUE IN R1
1017 1C32 BBEE             ZBSR   *IN    GET A CHAR
1018 1C34 7509             CPSL   C+WC   CLEAR CARRY AND WITH CARRY
1019 1C36 3B2C             BSTR,UN AHO3   LOOK UP VALUE
1020 1C38 01               LODZ   R1     GET SAVED VALUE
1021 1C39 62               IORZ   R2     MAKE THE BINARY BYTE
1022                *
1023                *CALCULATE BCC
1024                *
1025 1C3A C1        CBCC   STRZ   R1     SAVE VALUE
1026 1C3B 2C17E1           EORA,R0 BCC   XOR WITH CURRENT BCC
1027 1C3E D0               RRL,R0        ROTATE LEFT
```

```
LINE ADDR  OBJECT  E SOURCE

1028 1C3F C8FB          STRR,R0 *CBCC+2 UPDATE THE BCC
1029 1C41 17            RETC,UN
1030              *
1031              *
1032              *FINISHED READING FILE
1033              *
1034 1C42 0C17DF  LOAD1 LODA,R0 FID     CHECK FILE ID FLAG FOR FILE ID FOUND
1035 1C45 1C1BCF        BCTA,EQ RCAS2   NO LOOK FOR START OF NEXT FILE
1036 1C48 088F          LODR,R0 *INK2+1 GET VALUE FROM MEM     PLACE START ADDRESS IN PC
1037 1C4A CC17E8        STRA,R0 LADR
1038 1C4D 0887          LODR,R0 *INK+1  GET VALUE FROM MEM+1
1039 1C4F CC17E9        STRA,R0 LADR+1
1040 1C52 1F1874        BCTA,UN MON3     GO TO THE MONITOR
1041              *
1042              *INCREMENT ADDRESS MEM
1043              *
1044 1C55 0C17DE  INK   LODA,R0 MEM+1   GET ADDRESS
1045 1C58 0E17DD  INK2  LODA,R2 MEM
1046 1C5B D802          BIRR,R0 INK1    INCREMENT IT
1047 1C5D DA00          BIRR,R2 INK1
1048 1C5F C8F5    INK1  STRR,R0 *INK+1   SAVE  IN MEM+1
1049 1C61 CAF6          STRR,R2 *INK2+1 SAVE IN MEM
1050 1C63 17            RETC,UN
1051              *
1052              *LOOK UP ASCII HEX TO CONVERT TO BINARY
1053              *
1054 1C64 06FF    AHO3  LODI,R2 255     PRESET INDEX
1055 1C66 EE3FC5        COMA,R0 ASCII,R2,+      CHECK THE VALUE
1056 1C69 14            RETC,EQ RETURN IF EQUAL
1057 1C6A E610          COMI,R2 H'10'   CHECK FOR MAX COUNT
1058 1C6C 9878          BCFR,EQ AHO3+2  LOOP
1059 1C6E 0706          LODI,R3 6       CHAR NOT ASCII HEX
1060 1C70 9BE8          ZBRR    *ERR    GOTO ERROR
1061              *
1062              *CARRAGE RETURN AND LINE FEED
1063              *
1064 1C72 040D    CRLFF LODI,R0 13      CARRAGE RETURN
1065 1C74 BBF0          ZBSR *OUT       PRINT
1066 1C76 040A          LODI,R0 10      LINE FEED
1067 1C78 BBF0          ZBSR *OUT       PRINT
1068 1C7A 17            RETC,UN
1069              *
1070              *CONVERT BINARY TO ASCII HEX AND PRINT
1071              *
1072 1C7B 7508    HOUTT  CPSL    WC
1073 1C7D BBF4          ZBSR    *DISLSD CONVERT BIN TO NIBBLE
1074 1C7F C2            STRZ    R2      SAVE IN R2
1075 1C80 0E7FC5        LODA,R0 ASCII,R2        TENS DIGIT
1076 1C83 BBF0          ZBSR    *OUT    PRINT TENS DIGIT
1077 1C85 0D7FC5        LODA,R0 ASCII,R1        GET UNITS DIGIT
1078 1C88 BBF0          ZBSR *OUT       PRINT UNITS DIGIT
1079 1C8A 17            RETC,UN
```

```
LINE ADDR  OBJECT  E SOURCE

1081               ************************************************************
1082               *
1083               *
1084               *WRITE CASSETTE COMMAND
1085               *
1086               *THIS ROUTINE WRITES 2650 HEX FORMAT TO CASSETTE TAPE
1087               *
1088               *REGISTERS USED
1089               *
1090               *R0 SCRATCH
1091               *R1 SCRATCH
1092               *R2 SCRATCH
1093               *R3 SCRATCH
1094               *
1095               *SUBROUTINES CALLED
1096               *
1097               *OUT WRITE CHAR TO TAPE
1098               *HOUT CONVERT BINARY TO ASCII HEX AND WRITE TO TAPE
1099               *INK INCREMENT POINTER MEM
1100               *
1101               *RAM USED
1102               *
1103               *BCC   BLOCK CHECK CHAR
1104               *MEM   POINTER
1105               *BAD   PROGRAM START ADDRESS
1106               *SAD   DUMP STOP ADDRESS
1107               *FID   FILE ID FLAG AND STORAGE
1108               *
1109               *THIS ROUTINE PUNCHES A HEX FORMAT TAPE
1110               *
1111               *
1112               *   LEADER16ID:ADDRCTBCAADDCCRR.....BC
1113               *
1114               ************************************************************
1115               *
1116               *
1117 1C8B 20      WCAS4  EORZ    R0      GET A 0
1118 1C8C BBFA           ZBSR    *GNP    GET NUMBER
1119 1C8E E687           COMI,R2 H'87'   ENTR/NXT KEY
1120 1C90 17             RETC,UN
1121               *
1122               *
1123 1C91 051F    WCAS   LODI,R1 <LADEQ-1        GET ADDRESS OF LAD= DISPLAY
1124 1C93 06BC           LODI,R2 >LADEQ-1
1125 1C95 B8FE           ZBSR    *MOV    MOVE TO DISPLAY BUFFER
1126 1C97 3B72           BSTR,UN WCAS4   GET ADDRESS DATA
1127 1C99 988E           BCFR,EQ *WCAS6+1 MON2    IF NOT EXIT
1128 1C9B CD17DD         STRA,R1 MEM     SAVE START ADDRESS
1129 1C9E CC17DE         STRA,R0 MEM+1
1130 1CA1 0412           LODI,R0 H'12'   CHANGE DISPLAY
1131 1CA3 CC17D1         STRA,R0 DISBUF  DISPLAY 'UAD=    '
1132 1CA6 3B63           BSTR,UN WCAS4   GET ADDRESS DATA
1133 1CA8 9C187D  WCAS6  BCFA,EQ MON2    NOT ENTR/NXT MUST BE NEW COMMAND
1134               *
1135               *CHECK FOR START ADDRESS GT THAN STOP
```

```
LINE ADDR  OBJECT  E SOURCE

1136                *
1137 1CAB ED17DD           COMA,R1 MEM     CHECK HI BYTE
1138 1CAE 1806             BCTR,EQ WCAS7
1139 1CB0 1909             BCTR,GT WCAS9
1140 1CB2 0707      WCAS8  LODI,R3 7       SET THE ERROR NUMBER
1141 1CB4 9BE8             ZBRR    *ERR    GOTO ERROR
1142 1CB6 EC17DE    WCAS7  COMA,R0 MEM+1   CHECK LO BYTE
1143 1CB9 1A77             BCTR,LT WCAS8
1144                *
1145 1CBB D802      WCAS9  BIRR,R0 WCASA   INCREMENT STOP ADDRESS
1146 1CBD D900             BIRR,R1 WCASA   SO DUMP IS INCLUSIVE
1147 1CBF CD17C8    WCASA  STRA,R1 EAD     SAVE END ADDRESS
1148 1CC2 CC17C9           STRA,R0 EAD+1
1149 1CC5 0405             LODI,R0 H'05'   CHANGE DISPLAY
1150 1CC7 CC17D1           STRA,R0 DISBUF  DISPLAY 'SAD=    '
1151 1CCA 3F1C8B           BSTA,UN WCAS4   GET PROGRAM START ADDRESS
1152 1CCD 9C9CA9    WCAS3  BCFA,EQ *WCAS6+1 MON2   GOTO MONITOR NEW FUNCTION
1153 1CD0 CD17CA           STRA,R1 BAD     SAVE START ADDRESS
1154 1CD3 CC17CB           STRA,R0 BAD+1
1155 1CD6 051F             LODI,R1 <FEQ-1  GET ADDRESS OF F= DISPLAY
1156 1CD8 06B4             LODI,R2 >FEQ-1
1157 1CDA BBFE             ZBSR *MOV     MOVE DATA TO DISBUF
1158 1CDC 0401             LODI,R0 1       SET FLAG TO SINGLE BYTE
1159 1CDE BBFA             ZBSR *GNP     GET THE FILE ID
1160 1CE0 E687             COMI,R2 H'87'   ENTR/NXT KEY
1161 1CE2 98C5             BCFR,EQ *WCAS6+1 MON2    EXIT NEW COMMAND
1162 1CE4 C894             STRR,R0 *WCAS5+1   SAVE FILE ID
1163 1CE6 060A             LODI,R2 10      SET THE DELAY
1164 1CE8 0719      PUN10  LODI,R3 25
1165 1CEA 20                EORZ    R0      GET A 0
1166 1CEB BBF0             ZBSR    *OUT    OUTPUT A LEADER
1167 1CED FB7B             BDRR,R3 PUN10+2
1168 1CEF 12               SPSU        GET FLAG
1169 1CF0 2440             EORI,R0 H'40'   COMPLEMENT IT
1170 1CF2 92               LPSU        RESTORE IT
1171 1CF3 FA73             BDRR,R2 PUN10    DECREASE THE COUNT
1172 1CF5 0416             LODI,R0 H'16' START OF FILE CHAR
1173 1CF7 BBF0             ZBSR *OUT       PRINT
1174 1CF9 0C17E0    WCAS5  LODA,R0 FID+1   GET FILE ID
1175 1CFC BBF2             ZBSR *HOUT      CONVERT TO ASCII HEX AND PRINT
1176 1CFE BBF8      PUN2   ZBSR    *CRLF   OUTPUT CARRAGE RETURN AND LINE FEED
1177 1D00 043A             LODI,R0 A':'    START OF BLOCK CHAR
1178 1D02 BBF0             ZBSR    *OUT    PRINT
1179 1D04 20               EORZ    R0      GET A 0
1180 1D05 C8A3             STRR,R0 *PUN3+1     PRESET BCC
1181 1D07 0C17C8           LODA,R0 EAD CALCULATE NO OF BYTES TO OUTPUT
1182 1D0A 7709             PPSL    WC+C    SET CARRY AND WITH CARRY
1183 1D0C 0F17C9           LODA,R3 EAD+1   GET END ADDRESS
1184 1D0F ABAC             SUBR,R3 *BDUM1+1 MEM+1  SUBTRACT START ADDRESS FROM STOP ADDRESS
1185 1D11 A8A5             SUBR,R0 *BDUM+1 MEM
1186 1D13 7508             CPSL    WC      CLEAR WITH CARRY
1187 1D15 1E1CB2           BCTA,NG WCAS8   START > STOP
1188                *
1189                *
1190 1D18 581B      PUN4   BRNR,R0 ADUM    START ADDRESS GT THAN 256 AWAY FROM STOP
1191 1D1A 5B15             BRNR,R3 GDUM    START ADDRESS LT 256 AWAY FROM STOP
```

```
LINE ADDR  OBJECT  E SOURCE

1192 1D1C 0C17CA          LODA,R0 BAD THIS IS END OF FILE BLOCK
1193 1D1F 3B39            BSTR,UN EDUM SO OUTPUT START ADDRESS OF PROGRAM
1194 1D21 0C17CB          LODA,R0 BAD+1
1195 1D24 3B34            BSTR,UN EDUM    OUTPUT A BYTE AS 2 ASCII HEX CHARS
1196 1D26 20              EORZ    R0      END OF FILE BLOCK
1197 1D27 3B31            BSTR,UN EDUM OUTPUT BYTE COUNT
1198 1D29 0C17E1    PUN3  LODA,R0 BCC     GET BCC
1199 1D2C 3B2C            BSTR,UN EDUM OUTPUT BCC
1200 1D2E 1F1874          BCTA,UN       MON3      GOTO MONITOR
1201                *
1202                *
1203 1D31 E71E      GDUM  COMI,R3 H'1E'   IS START LT 30 AWAY FROM STOP
1204 1D33 1A02            BCTR,LT BDUM    OUTPUT LAST BYTES
1205 1D35 071E      ADUM  LODI,R3 H'1E'   NO OF BYTES THIS RECORD IS 30
1206 1D37 0C17DD    BDUM  LODA,R0 MEM     OUT ADDR HI
1207 1D3A 3B1E            BSTR,UN EDUM    OUTPUT BYTE AS 2 ASCII HEX CHARS
1208 1D3C 0C17DE    BDUM1 LODA,R0 MEM+1   OUT ADDR LO
1209 1D3F 3B19            BSTR,UN EDUM    OUTPUT BYTE AS 2 ASCII HEX CHARS
1210 1D41 03              LODZ    R3      OUT BYTE COUNT
1211 1D42 3B16            BSTR,UN EDUM    OUTPUT BYTE AS 2 ASCII HEX CHARS
1212 1D44 08E4            LODR,R0 *PUN3+1     OUT BCC FOR ADDR AND BYTE COUNT
1213 1D46 3B12            BSTR,UN EDUM    OUTPUT BYTE AS 2 ASCII HEX CHARS
1214 1D48 0C97DD    DDUM  LODA,R0 *MEM    OUTPUT DATA FROM MEM
1215 1D4B 3B0D            BSTR,UN EDUM    OUTPUT BYTE AS 2 ASCII HEX CHARS
1216 1D4D 3F1C55          BSTA,UN INK     INCREMEMT POINTER MEM
1217 1D50 FB76            BDRR,R3 DDUM    LOOP TILL DONE
1218 1D52 0C17E1          LODA,R0 BCC     GET BCC
1219 1D55 3B03            BSTR,UN EDUM    OUTPUT BCC FOR DATA
1220 1D57 1F1CFE          BCTA,UN PUN2
1221                *
1222 1D5A 3F1C3A    EDUM  BSTA,UN CBCC    CALCULATE BCC
1223 1D5D 01              LODZ    R1      GET VALUE TO OUTPUT
1224 1D5E BBF2            ZBSR    *HOUT   PRINT AS 2 ASCII HEX CHARS
1225 1D60 17              RETC,UN
```

```
LINE ADDR  OBJECT  E SOURCE

1227               ********************************************************
1228               *
1229               *
1230               *SET OR CLEAR BREAK POINT
1231               *
1232               *
1233               *TO SET BREAK POINT ENTR ADDRESS AND DEPRESS FUNCTION KEY
1234               *TO CLEAR BREAK POINT DEPRESS FUNCTION KEY
1235               *
1236               *SUBROUTINES CALLED
1237               *
1238               *MOV MOVE DATA TO DISBUF
1239               *GNPA DISPLAY AND GET ADDRESS DATA
1240               *ROT ROTATE A NIBBLE
1241               *SCBP2 SET DISBUF 4&5
1242               *BRKPT4 SET DISBUF 6&7
1243               *DSLSD CONVERT TO BINARY FOR DISPLAY
1244               *
1245               *RAM MEMORY USED
1246               *
1247               *BPF BREAK POINT FLAG
1248               *BPL LOCATION OF BREAK POINT
1249               *BPD DATA TO BE RESTORED IN BREAK POINT LOCATION
1250               *
1251               *
1252               *
1253               ********************************************************************
1254               *
1255 1D61 051F     SCBP   LODI,R1 <BPEQ-1  GET ADDRESS OF BP= DISPALY
1256 1D63 069C            LODI,R2 >BPEQ-1
1257 1D65 BBFE            ZBSR *MOV     MOVE DATA TO DISBUF
1258 1D67 0C17CF          LODA,R0 BPF     BREAK POINT SET?
1259 1D6A 180A            BCTR,EQ SCBP1   NOT SET GET ADDRESS
1260               *
1261               *BREAK POINT SET SET UP ADDRESS DISPLAY
1262               *
1263 1D6C 0C17CE          LODA,R0 BPL+1   PREPARE THE ADDRESS
1264 1D6F BBEA            ZBSR  *BRKPT4  SET UP DISPLAY
1265 1D71 0C17CD          LODA,R0 BPL     GET MSB
1266 1D74 3B14            BSTR,UN SCBP2   SETUP DISPLAY
1267 1D76 20       SCBP1  EORZ    R0      SET UP GET NUMBER PARAMETER TO 4 DIGIT
1268 1D77 BBFC            ZBSR *GNPA    GET THE ADDESS IF ANY
1269 1D79 1818            BCTR,EQ SCBP4   SET THE BREAK POINT
1270               *
1271               *THIS SECTION CLEARS THE BREAK POINT
1272               *
1273 1D7B 0B88            LODR,R3 *SCBP6+1     CHECK BREAK POINT FLAG
1274 1D7D 1889            BCTR,EQ *SCBP5+1    BREAK POINT NOT SET GO TO MONITOR
1275 1D7F E681            COMI,R2 H'81'   IS TERMINATION BKP?
1276 1D81 9885            BCFR,EQ *SCBP5+1    NO LEAVE BREAK POINT SET GO TO MONITOR
1277 1D83 20              EORZ    R0      GET A 0
1278 1D84 CC17CF   SCBP6  STRA,R0 BPF     CLEAR BREAK POINT FLAG
1279 1D87 1F187D   SCBP5  BCTA,UN MON2    GO TO MONITOR
1280               *
1281 1D8A BBF4     SCBP2  ZBSR    *DISLSD CONVERT TO BIN FOR DISPLAY
```

```
LINE ADDR  OBJECT  E SOURCE

1282 1D8C CD17D6           STRA,R1 DISBUF+5
1283 1D8F CD17D5           STRA,R0 DISBUF+4
1284 1D92 17               RETC,UN
1285                *
1286                *THIS SECTION SETS THE BREAK POINT
1287                *
1288 1D93 CD17CE    SCBP4  STRA,R0 BPL+1   SET BREAK POINT ADDRESS
1289 1D96 CD17CD           STRA,R1 BPL
1290 1D99 20               EORZ    R0      CLEAR BREAK POINT FLAG
1291 1D9A C8E9             STRR,R0 *SCBP6+1        CHECK THE BREAK POINT CAN BE SET
1292 1D9C 0C97CD           LODA,R0 *BPL    GET DATA FROM BREAK POINT LOCATION
1293 1D9F 05B0             LODI,R1 H'B0'   BREAK POINT INSTRUCTION...WRTC,R0
1294 1DA1 CD97CD           STRA,R1 *BPL    TRY TO SET BREAK POINT
1295 1DA4 ED97CD           COMA,R1 *BPL    DID IT SET OK?
1296 1DA7 1804             BCTR,EQ SCBP7   BREAK POINT CAN BE SET
1297 1DA9 0701             LODI,R3 1       CANT SET BREAK POINT ERROR
1298 1DAB 9BE8             ZBRR    *ERR    GOTO ERROR
1299 1DAD CC97CD    SCBP7  STRA,R0 *BPL    RESTORE USER DATA
1300 1DB0 047F             LODI,R0 127     SET THE BREAK POINT FLAG
1301 1DB2 C8D1             STRR,R0 *SCBP6+1        SET IT
1302 1DB4 1BD2             BCTR,UN *SCBP5+1    GOTO MONITOR
```

```
LINE ADDR  OBJECT  E SOURCE

1304                ******************************************************
1305                *
1306                *
1307                *MOVE 8 BYTES OF DATA POINTED TO IN R1 AND R2 TO DISBUF
1308                *
1309                *
1310                *REGISTERS USED
1311                *
1312                *R0 SCRATCH
1313                *R1 HI ADDRESS BYTE OF DATA ADDRESS-1
1314                *R2 LO ADDRESS BYTE OF DATA ADDRESS-1
1315                *R3 NOT USED
1316                *
1317                *SUBROUTINES CALLED
1318                *
1319                *NONE
1320                *
1321                *RAM MEMORY USED
1322                *
1323                *T TEMP INDIRECT ADDRESS
1324                *
1325                ************************************************************
1326                *
1327 1DB6 CD17E3    MOVI   STRA,R1 T       SET INDIRECT ADDRESS
1328 1DB9 CE17E4           STRA,R2 T+1
1329 1DBC 0608             LODI,R2 8       SET INDEX TO MOVE 8 BYTES
1330 1DBE 0EF7E3    MOV1   LODA,R0 *T,R2   GET A BYTE
1331 1DC1 CE77D0           STRA,R0 DISBUF-1,R2       MOVE TO BUFFER
1332 1DC4 FA78             BDRR,R2 MOV1
1333 1DC6 17               RETC,UN
```

```
LINE ADDR  OBJECT  E SOURCE

1335                ************************************************************
1336                *
1337                *
1338                *KEY BOARD SCAN AND DISPLAY ROUTINE
1339                *
1340                *THIS ROUTINE WRITTEN BY ALEX GOLDBURGER
1341                *
1342                *
1343                *TO USE THIS ROUTINE PLACE DATA TO BE DISPLAYED
1344                *IN DISBUF (SEE CODES AT BEGINNING OF PROGRAM)
1345                *
1346                *ON ENTRY R0 CONTAINS A FLAG
1347                *
1348                *R0 = 0 NORMAL OPERATION
1349                *       ON EXIT R0 = KEY PRESSED CODE
1350                *R0 = 1-127 GO THRU SCAN ONCE AND EXIT
1351                *          ON EXIT R0 = KEY PRESSED CODE
1352                *R0 = H'80' TURN ON DECIMAL POINT FOR ENTRY MODE
1353                *          ON EXIT R0 = KEY PRESSED CODE
1354                *
1355                *SEE KEY PRESSED CODES AT BEGINNING OF PROGRAM
1356                *
1357                *REGISTERS USED IN BANK ON ENTRY
1358                *
1359                *R0    SCRATCH
1360                *R1    KEYBOARD FLAGS
1361                *R2    DIGIT SELECT
1362                *R3    DIGIT POINTER
1363                *
1364                *SUBROUTINES CALLED
1365                *
1366                *NONE
1367                *
1368                *RAM MEMORY USED
1369                *
1370                *DISBUF        DISPLAY BUFFER
1371                *KFLG  KEY BOARD FLAG
1372                *
1373                ************************************************************
1374                *
1375 1DC7 0406        DLOOP  LODI,R0 6       DELAY TO MAKE LOOPS EQUAL
1376 1DC9 F87E               BDRR,R0 $
1377 1DCB 52        DLOOP1 RRR,R2  ROTATE DIGIT SELECT
1378 1DCC 0F77D0             LODA,R0 DISBUF-1,R3       GET DATA TO BE DISPLAYED
1379 1DCF C1                 STRZ    R1      SAVE DISPLAY CODE
1380 1DD0 4580               ANDI,R1 H'80'   MASK FOR DECIMAL POINT
1381 1DD2 447F               ANDI,R0 H'7F'   MASK OFF DECIMAL POINT
1382 1DD4 0C7F68             LODA,R0 SEGTBL,R0       CONVERT TO SEGMENT DATA
1383 1DD7 61                 IORZ    R1      SET THE DECIMAL POINT IF NEEDED
1384 1DD8 F601               TMI,R2  H'01'   COL 7?
1385 1DDA 1A08               BCTR,NG DLOOP3  DONT PUT DECIMAL POINT HERE
1386 1DDC 0D17ED             LODA,R1 KFLG+1  GET FLAG
1387 1DDF 9A03               BCFR,NG DLOOP3  IF FLAG NOT NEG NO DECIMAL POINT
1388 1DE1 4580               ANDI,R1 H'80'   MASK DECIMAL POINT
1389 1DE3 61                 IORZ    R1      SET DECIMAL POINT
```

```
LINE ADDR  OBJECT  E SOURCE

1390 1DE4 0500      DLOOP3 LODI,R1 0       GET A 0
1391 1DE6 D5F9             WRTE,R1 SEG     TURN OFF SEGMENTS
1392 1DE8 D6FA             WRTE,R2 DIGIT   ENABLE NEXT DIGIT
1393 1DEA D4F9             WRTE,R0 SEG     AND DISPLAY IT
1394 1DEC 0C17EC           LODA,R0 KFLG    SEE IF KEY IS DOWN?
1395 1DEF 980B             BCFR,EQ DLOOP4  KEY UP DEBOUNCE
1396 1DF1 1B1A             BCTR,UN DLOOP5  IS KEY DOWN?
1397                *
1398 1DF3 FB52      DLOOP2 BDRR,R3 DLOOP   DECREMENT DIGIT PTR
1399                *                      TEST IF ONE SCAN IS DONE.
1400                *                      IF ONE SCAN DONE INITIALIZE SCAN
1401                *                      PARAMETERS AND KEY FLAGS
1402 1DF5 0C17ED           LODA,R0  KFLG+1 CHECK FOR ONE PASS THEN EXIT MODE
1403 1DF8 1933             BCTR,GT DISP3   IF ONE PASS EXIT
1404 1DFA 1B23             BCTR,UN DISP4   RESET THE FLAGS
1405                *
1406                *
1407 1DFC 3B28      DLOOP4 BSTR,UN GETKEY  GET A KEY
1408 1DFE 9806             BCFR,EQ DLP0    KEY IS DOWN RESET DEBOUNCE
1409 1E00 0887             LODR,R0 *DLP1+1 KFLG+2  GET COUNTER VALUE
1410 1E02 F804             BDRR,R0 DLP1
1411 1E04 1B14             BCTR,UN DISP1   SET FLAG TO ACCEPT KEY
1412 1E06 0460      DLP0   LODI,R0 H'60'   SET THE DELAY COUNT
1413 1E08 CC17EE    DLP1   STRA,R0 KFLG+2  SAVE DELAY COUNT
1414 1E0B 1B66             BCTR,UN DLOOP2  DO THE NEXT SCAN
1415                *
1416                *
1417 1E0D 3B17      DLOOP5 BSTR,UN GETKEY  IS A KEY DOWN?
1418 1E0F 1862             BCTR,EQ DLOOP2    NO
1419 1E11 1B24             BCTR,UN CODE
1420                *
1421                *ENTRY TO DISPLAY ROUTINE HERE
1422                *
1423 1E13 CC17ED    DISPLI STRA,R0 KFLG+1  SAVE INPUT PARAMETER
1424 1E16 0460      DISP2  LODI,R0 H'60'   KEY WAS DOWN - SET KFLG
1425                *                      NOT TO ACCEPT KEY NEXT SCAN
1426 1E18 C8EF             STRR,R0 *DLP1+1 KFLG+2  SET KEY DEBOUNCE DELAY
1427 1E1A CC17EC    DISP1  STRA,R0 KFLG    SAVE KFLG
1428 1E1D 7509             CPSL    C+WC    CLEAR CARRY AND WITH CARRY
1429 1E1F 0708      DISP4  LODI,R3 H'08'   INITIALIZE DIGIT POINTER
1430 1E21 0601             LODI,R2 H'01'   AND DIGIT SELECT
1431 1E23 1F1DCB           BCTA,UN DLOOP1  GO DISPLAY
1432                *
1433                *GET KEY CODE
1434                *
1435 1E26 55FE      GETKEY REDE,R1 KBDIN   READ KEYBOARD
1436 1E28 450F             ANDI,R1 H'0F'   MASK OFF UNUSED BITS
1437 1E2A 250F             EORI,R1 H'0F'   INVERT THE INPUT
1438 1E2C 17               RETC,UN
1439                *
1440                *SINGLE PASS EXIT
1441                *
1442 1E2D 040A      DISP3  LODI,R0 10
1443 1E2F F87E             BDRR,R0 $       DELAY
1444 1E31 D4F9             WRTE,R0 SEG     TURN OFF SEGMENTS
1445 1E33 0488             LODI,R0 H'88'   NO KEY PRESSED CODE
```

```
LINE ADDR  OBJECT  E SOURCE

1446 1E35 C2              STRZ    R2      SAVE IN R2
1447 1E36 17              RETC,UN
1448                *
1449                *CONVERT KEY LINE DATA TO KEY CODE
1450                *
1451 1E37 20        CODE  EORZ    R0      GET A 0
1452 1E38 D4F9            WRTE,R0 SEG     TURN OFF SEGMENTS
1453 1E3A A701            SUBI,R3 1       DECREMENT COLUMN COUNTER
1454 1E3C D4FA            WRTE,R0 DIGIT   TURN OFF COLUMNS
1455 1E3E 0604      CODE1 LODI,R2 4       LOOP COUNT
1456 1E40 51        CODE4 RRR,R1          GET WEIGHT OF KEY LINE
1457 1E41 E580            COMI,R1 H'80'   CHECK FOR 1 KEY DOWN
1458 1E43 1808            BCTR,EQ CODE2   R0 = 0,4,8, OR H'C'
1459 1E45 8404            ADDI,R0 H'04'
1460 1E47 FA77            BDRR,R2 CODE4   CHECK FOR ONLY 1 KEY
1461 1E49 0708            LODI,R3 8       MORE THAN 1 KEY DOWN OR NO KEY DOWN
1462 1E4B 9BE8            ZBRR    *ERR    GOTO ERROR
1463 1E4D E704      CODE2  COMI,R3 H'04'   NUMBER OR FUNCTION KEY?
1464 1E4F 1A05            BCTR,LT CODE3   # KEY
1465 1E51 50              RRR,R0          DIVIDE KEYLINE WEIGHT BY 2
1466 1E52 6480            IORI,R0 H'80'   FUNCTION KEY DESIGNATOR
1467 1E54 4701            ANDI,R3 H'01'   RETAIN LSB ONLY
1468 1E56 83        CODE3 ADDZ    R3      TO GET WHOLE KEYCODE
1469 1E57 C2              STRZ    R2      SAVE KEY CODE IN R2
1470 1E58 17              RETC,UN
```

```
LINE ADDR  OBJECT  E SOURCE

1472                ************************************************************
1473                *
1474                *
1475                *GOTO ROUTINE
1476                *
1477                *
1478                *REGISTERS USED
1479                *
1480                *R0 SCRATCH
1481                *R1 SCRATCH
1482                *R2 SCRATCH
1483                *R3 SCRATCH
1484                *R1' RESTORED
1485                *R2' RESTORED
1486                *R3' RESTORED
1487                *PSU RESTORED
1488                *PSL RESTORED
1489                *
1490                *SUBROUTINES USED
1491                *
1492                *NONE
1493                *
1494                *RAM MEMORY USED
1495                *
1496                *SSF   SINGLE STEP FLAG
1497                *BPF   BREAK POINT FLAG
1498                *BPL   BREAK POINT LOCATION
1499                *BPD   BREAK POINT DATA
1500                *LADR INDIRECT ADDRESS TO JUMP THRU
1501                *
1502                ************************************************************
1503                *
1504 1E59 0C17D0    GO     LODA,R0 SSF     GET SINGLE STEP FLAG
1505 1E5C 9819             BCFR,EQ G01     NO SINGLE STEP GOTO USER
1506 1E5E 0C17CF           LODA,R0 BPF     GET BREAK POINT FLAG
1507 1E61 1814             BCTR,EQ G01     BREAK POINT GO TO USER NO BREAK POINT
1508 1E63 0C97CD           LODA,R0 *BPL    GET USER DATA
1509 1E66 CC17CC           STRA,R0 BPD     SAVE USER DATA
1510 1E69 04B0             LODI,R0 H'B0'   WRTC,R0 BREAK POINT INSTRUCTION
1511 1E6B CC97CD           STRA,R0 *BPL    SET THE BREAK POINT
1512 1E6E EC97CD           COMA,R0 *BPL    CHECK BREAK POINT SET OK
1513 1E71 1804             BCTR,EQ G01     GOTO USER
1514 1E73 0701             LODI,R3 1       ERROR BREAK POINT NOT SET OK
1515 1E75 9BE8             ZBRR    *ERR    GOTO ERROR
1516 1E77           G01    EQU     $
```

```
LINE ADDR  OBJECT  E SOURCE

1518               ************************************************************
1519               *
1520               *
1521               *RESTORE REGISTERS BEFORE GOING TO USER PROGRAM
1522               *
1523               *
1524               *
1525               *
1526               *REGISTERS USED
1527               *
1528               *R0 THRU R3' PSU PSL
1529               *
1530               *SUBROUTINES CALLED
1531               *
1532               *UREG+9       RESTORE PSL
1533               *
1534               *RAM MEMORY USED
1535               *
1536               *UREG         = R0
1537               *UREG+1       = R1
1538               *UREG+2       = R2
1539               *UREG+3       = R3
1540               *UREG+4       = R1'
1541               *UREG+5       = R2'
1542               *UREG+6       = R3'
1543               *UREG+7       = PSU
1544               *UREG+8       = PSL
1545               *UREG+9       = PPSL INSTRUCTION OPCODE
1546               *UREG+10      = PSL
1547               *UREG+11      = RETC,UN       INSTRUCTION OPCODE
1548               *
1549               ************************************************************
1550 1E77 0577    RESTRG LODI,R1 H'77'   PPSL INSTRUCTION OPCODE
1551 1E79 CD17FB         STRA,R1 UREG+9  CREATE A SUBROUTINE TO RESTORE PSL
1552 1E7C 0517         LODI,R1 H'17'   RETC,UN INSTRUCTION OPCODE
1553 1E7E CD17FD         STRA,R1 UREG+11
1554 1E81 7510         CPSL    RS      CLEAR REGISTER SWITCH
1555 1E83 0D17F3         LODA,R1 UREG+1  RESTORE R1
1556 1E86 0E17F4         LODA,R2 UREG+2  RESTORE R2
1557 1E89 0F17F5         LODA,R3 UREG+3  RESTORE R3
1558 1E8C 7710         PPSL    RS      SET THE REGISTER SWITCH
1559 1E8E 0D17F6         LODA,R1 UREG+4  RESTORE R1'
1560 1E91 0E17F7         LODA,R2 UREG+5  RESTORE R2'
1561 1E94 0F17F8         LODA,R3 UREG+6  RESTORE R3'
1562 1E97 0C17F9    RESTR1 LODA,R0 UREG+7  GET PSU DATA
1563 1E9A 6C17F1         IORA,R0 IFLG    SET INTERUPT INHIBIT IF REQUIRED
1564 1E9D 92           LPSU            RESTORE PSU
1565 1E9E 0C17F2         LODA,R0 UREG    RESTORE R0
1566 1EA1 75FF         CPSL    255     CLEAR PSL
1567 1EA3 3F17FB         BSTA,UN UREG+9  RESTORE PSL
1568 1EA6 1F97E8         BCTA,UN *LADR   GOTO USER
1569               *
```

```
LINE ADDR  OBJECT  E SOURCE

1571               ************************************************************
1572               *
1573               *
1574               *SUBROUTINE TO SAVE R1,R2,R3
1575               *
1576               *REGISTERS USED IN BANK ON ENTRY
1577               *
1578               *R1 SAVED IN SAVREG+1
1579               *R2 SAVED IN SAVREG+2
1580               *R3 SAVED IN SAVREG+3
1581               *
1582               *SUBROUTINES CALLED
1583               *
1584               *NONE
1585               *
1586               *RAM MEMORY USED
1587               *
1588               *SAVREG+1
1589               *SAVREG+2
1590               *SAVREG+3
1591               ************************************************
1592 1EA9 CD17DA  SAVR0  STRA,R1 SAVREG+1
1593 1EAC CE17DB  SAVR01 STRA,R2 SAVREG+2
1594 1EAF CF17DC  SAVR02 STRA,R3 SAVREG+3
1595 1EB2 17            RETC,UN
1596               ************************************************************
1597               *
1598               *
1599               *SUBROUTINE TO RESTORE R1,R2,R3
1600               *
1601               *
1602               *REGISTERS USED IN BANK ON ENTRY
1603               *
1604               *R1 RESTORED TO VALUE IN SAVREG+1
1605               *R2 RESTORED TO VALUE IN SAVREG+2
1606               *R3 RESTORED TO VALUE IN SAVERG+3
1607               *
1608               *SUBROUTINES CALLED
1609               *
1610               *NONE
1611               *
1612               *RAM MEMORY USED
1613               *
1614               *SAVREG+1
1615               *SAVREG+2
1616               *SAVREG+3
1617               ************************************************
1618 1EB3 09F5    RESTR0 LODR,R1 *SAVR0+1
1619 1EB5 0AF6           LODR,R2 *SAVR01+1
1620 1EB7 0BF7           LODR,R3 *SAVR02+1
1621 1EB9 17             RETC,UN
```

```
LINE ADDR  OBJECT  E SOURCE

1623                ********************************************************
1624                *
1625                *
1626                *CASSETTE IO ROUTINES
1627                *PROGRAM WRITTEN BY BBC
1628                *
1629                * 04-27-77
1630                *
1631                * THESE ROUTINES WRITES OR READS ONE BYTE TO OR FROM
1632                * THE CASSETTE IN SIMCA FORMAT.
1633                *
1634                * THE FREQUENCY IS DETERMINED BY FREQ
1635                * (CYCLE TIME IS 3.333 MICRO-SEC.)
1636                *
1637                *
1638                *ROUTINES SAVE AND REESTORE R1,R2,R3 OF CURRENT BANK
1639                *
1640                *IN RETURNS WITH DATA BYTE IN R0
1641                *OUT REQUIRES BYTE TO BE OUTPUT TO BE IN R0
1642                *
1643                *TCAS IS THE CASSETTE READ TEST USED TO SET LEVELS ON PLAY BACK
1644                *
1645                *SEE FRONT OF PROGRAM FOR DISPLAYS AND INSTRUCTIONS
1646                *
1647                *
1648                *REGISTERS USED
1649                *
1650                *R0,R1,R2,R3 ARE SCRATCH
1651                *
1652                *SUBROUTINES CALLED
1653                *
1654                *SAVR0 SAVES R1,R2,R3
1655                *RESTR0 RESTORES R1,R2,R3
1656                *
1657                *RAM MEMORY USED
1658                *
1659                *TEMP  TEMPORARY STORAGE
1660                *
1661                ********************************************************************
1662 0011          FREQ   EQU     17      PULSE TIME ( 0.2 MSEC. )
1663 0088          SPDLY  EQU     8*FREQ  INTER-BIT SPACE
1664 0013          TMDLY  EQU     19      TIME-OUT FOR INTER-BIT DETECTION
1665 0003          PULS1  EQU     3       NUMBER OF PULSES FOR A ONE
1666 0006          PULS0  EQU     2*PULS1 NUMBER OF PULSES FOR A ZERO
1667 0009          THRES  EQU     3*PULS1 TRANSITION THRESHOLD FOR DETECTION
1668 000F          EBIT   EQU     5*PULS1 TRANSITION THRESHOLD FOR END BIT
1669                *
1670                *
1671                * SUBROUTINE OUT
1672                * WRITES ONE BYTE FROM R0 TO CASSETTE
1673                *
1674 1EBA 3B6D      OUTT   BSTR,UN SAVR0   SAVE R1-R3
1675 1EBC D407             WRTE,R0 LEDS    WRITE BYTE TO LEDS FOR DISPLAY
1676 1EBE 0708           LODI,R3 8       BIT COUNT
1677 1EC0 C8A8      OUT1   STRR,R0 *OUT5+1 TEMP    SAVE BYTE IN TEMP
```

```
LINE ADDR  OBJECT  E SOURCE

1678 1EC2 CBAA            STRR,R3 *OUT6+1 TEMP+1  SAVE BIT COUNT IN TEMP+1
1679 1EC4 0506            LODI,R1 PULS0   GET NUMBER OF PULSES FOR A ZERO
1680 1EC6 F401            TMI,R0  H'01'   TEST FOR A ONE
1681 1EC8 9801            BCFR,0  OUT2
1682 1ECA 51              RRR,R1          DIVIDE COUNT IF A ONE
1683 1ECB FB02      OUT2  BDRR,R3 OUT3    CHECK FOR LAST BIT
1684 1ECD 8506            ADDI,R1 PULS0   YES, ADD LAST BIT PULSES
1685 1ECF 0611      OUT3  LODI,R2 FREQ    LENGTH OF PULSE
1686 1ED1 0718            LODI,R3 H'18'   SET ENV AND FREQ
1687 1ED3 D7F8            WRTE,R3 CAS
1688 1ED5 FA7E            BDRR,R2 $       DELAY 10 MICRO-SEC PER ITERATION
1689 1ED7 0611            LODI,R2 FREQ    LENGTH OF PULSE
1690 1ED9 0710            LODI,R3 H'10'   RESET FREQ
1691 1EDB D7F8            WRTE,R3 CAS
1692 1EDD FA7E            BDRR,R2 $       DELAY 10 MICRO-SEC PER ITERATION
1693 1EDF F96E            BDRR,R1 OUT3    DO NEXT PULSE
1694 1EE1 0688            LODI,R2 SPDLY   INTER-BIT SPACE
1695 1EE3 0700            LODI,R3 H'00'   TURN OFF ENV AND FREQ
1696 1EE5 D7F8            WRTE,R3 CAS
1697 1EE7 FA7E            BDRR,R2 $       DELAY 10 MICRO-SEC PER ITERATION
1698                *
1699 1EE9 0C17C6    OUT5  LODA,R0 TEMP    GET CHARACTER BACK
1700 1EEC 50              RRR,R0          ROTATE RIGHT ONE PLACE
1701 1EED 0F17C7    OUT6  LODA,R3 TEMP+1  GET BIT COUNT
1702 1EF0 FB4E            BDRR,R3 OUT1    CONTINUE IF COUNT NON-ZERO
1703 1EF2 3F1EB3    OUT4  BSTA,UN RESTR0  RESTORE R1-R3
1704 1EF5 17              RETC,UN         ELSE, RETURN
1705                *
1706                * SUBROUTINE IN
1707                * READS ONE BYTE FROM CASSETTE TO R0
1708                *
1709 1EF6 3F1EA9    INN   BSTA,UN SAVR0   SAVE R1-R3
1710 1EF9 20              EORZ    R0      SET R0 TO ZERO
1711 1EFA 44FE      IN1   ANDI,R0 H'FE'   MASK OUT LOW BIT
1712 1EFC C8EC            STRR,R0 *OUT5+1 TEMP    SAVE PARTIAL BYTE
1713 1EFE 3B0A            BSTR,UN GBIT    GET NEXT BIT
1714 1F00 88E8            ADDR,R0 *OUT5+1 TEMP    ADD IN PARTIAL BYTE
1715 1F02 50              RRR,R0          MOVE NEW BIT TO HIGH POSITION
1716 1F03 5975            BRNR,R1 IN1     TEST LAST BIT FLAG
1717 1F05 3BEC            BSTR,UN *OUT4+1  YES, RESTORE R1-R3
1718 1F07 D407            WRTE,R0 LEDS    WRITE BYTE TO LEDS FOR DISPLAY
1719 1F09 17              RETC,UN         RETURN
1720                *
1721                * SUBROUTINE TO GET THE NEXT BIT FROM CASSETTE
1722                * BIT IS RETURNED AS LEAST SIGNIFICANT BIT OF R0
1723                *
1724 1F0A 0580      GBIT  LODI,R1 H'80'
1725 1F0C D5F8            WRTE,R1 CAS     SET SENSE TO CASSETTE
1726 1F0E 12              SPSU            GET PSU
1727 1F0F 07FF            LODI,R3 -1      SET TRANSITION COUNT TO -1
1728 1F11 06FF            LODI,R2 H'FF'   SET TIME-OUT TO MAX FOR FIRST TRANSITION
1729 1F13 1B02            BCTR,UN GBT3
1730 1F15 0613      GBT2  LODI,R2 TMDLY   SET END-OF-BIT DETECTION DELAY
1731 1F17 C1        GBT3  STRZ    R1      SAVE LAST COPY OF PSU IN R1
1732 1F18 8701            ADDI,R3 1       INCREMENT TRANSITION COUNTER
1733 1F1A 12        GBT4  SPSU            LOOK FOR TRANSITION
```

```
LINE ADDR  OBJECT  E SOURCE

1734 1F1B E1              COMZ    R1
1735 1F1C 9877            BCFR,EQ GBT2    IF NOT EQUAL NEW TRANSITION
1736 1F1E FA7A            BDRR,R2 GBT4    IF EQUAL, TEST TIME-OUT
1737 1F20 20              EORZ    R0      SET R0 TO ZERO
1738 1F21 D4F8            WRTE,R0 CAS     SET SENSE BACK TO USER
1739 1F23 0501            LODI,R1 1       PRESET END FLAG TO 1
1740 1F25 E70F            COMI,R3 EBIT    ENDBIT THRESHOLD
1741 1F27 9903            BCFR,GT GBT5
1742 1F29 A70C            SUBI,R3 2*PULS0 LAST BIT, SUB ENDBIT PULSES
1743 1F2B C1              STRZ    R1      AND SET END FLAG
1744 1F2C E709      GBT5  COMI,R3 THRES   IS COUNT GREATER THAN THRESHOLD
1745 1F2E 15              RETC,GT         RETURN IF TRUE
1746 1F2F 0401            LODI,R0 1       NO, SET BIT TO ONE
1747 1F31 17              RETC,UN RETURN
1748                *
1749                *
1750                * SUBROUTINE TEST CASSETTE READS
1751                *
1752 1F32 0580      TCAS  LODI,R1 H'80'   SELECT LEAST SIGNIFICANT DIGIT
1753 1F34 D5FA            WRTE,R1 DIGIT
1754 1F36 0740      TCS0  LODI,R3 H'40'   OUTPUT '-' TO DISPLAY
1755 1F38 D407      TCS1  WRTE,R0 LEDS    OUTPUT VALUE TO LED'S
1756 1F3A D7F9            WRTE,R3 DISP    OUTPUT TO DISPLAY
1757 1F3C CF17C7    TCS10 STRA,R3 TEMP+1  SAVE UD CONDITION
1758 1F3F 060A            LODI,R2 10      RETURN AFTER 10 EXACT READS
1759 1F41 CE17C6    TCS2  STRA,R2 TEMP    SAVE R2
1760 1F44 3B44            BSTR,UN GBIT    GET A BIT
1761 1F46 0AFA            LODR,R2 *TCS2+1 TEMP    RESTORE R2
1762 1F48 050C            LODI,R1 2*PULS0 NUMBER OF TRANSITIONS FOR A ZERO
1763 1F4A 60              IORZ    R0      GET CONDITION CODE FOR R0
1764 1F4B 1801            BCTR,EQ TCS3    BRANCH IF A ZERO
1765 1F4D 51              RRR,R1          DIVIDE NOMINAL TRANSITION COUNT BY 2
1766 1F4E 03        TCS3  LODZ    R3      GET COUNT IN R0
1767 1F4F 1867            BCTR,EQ TCS1    BLANK DISPLAY IF 0
1768 1F51 A1              SUBZ    R1      TEST COUNT
1769 1F52 9804            BCFR,EQ TCS4    IF NOT EQUAL, RETURN
1770 1F54 FA6B      TCS35 BDRR,R2 TCS2    IF EQUAL AND COUNT NOT UP, GET NEW BIT
1771 1F56 1B5E            BCTR,UN TCS0
1772 1F58 190A      TCS4  BCTR,GT TCS5    DETERMINE POLARITY
1773 1F5A 08E1            LODR,R0 *TCS10+1        TEMP+1 GET UD CONDITION
1774 1F5C E4DE            COMI,R0 H'DE'   DOWN CONDITION
1775 1F5E 1874            BCTR,EQ TCS35    CANT GO DIRECT FROM DOWN TO UP
1776 1F60 073E            LODI,R3 H'3E'   OUTPUT 'U' TO DISPLAY
1777 1F62 1B54            BCTR,UN TCS1
1778 1F64 07DE      TCS5  LODI,R3 H'DE'   OUTPUT 'D' TO DISPLAY
1779 1F66 1B50            BCTR,UN TCS1
```

```
LINE ADDR  OBJECT  E SOURCE

1781                ************************************************************************
1782                *
1783                *HEXTAB LOOKUP TABLE FOR HEX TO SEVEN SEGMENT
1784                *
1785                *THIS TABLE CONTAINS THE VALUES FOR LIGHTING THE
1786                *SEGMENTS FOR THE DIGITS 0 THRU 9 AND LETTERS A TO F
1787                *
1788 1F68 3F065B4F  SEGTBL DATA    H'3F,06,5B,4F,66,6D,7D,07,7F,67,77,FC,39,DE,79,71'
     1F6C 666D7D07
     1F70 7F6777FC
     1F74 39DE7971
1789                *
1790                *SEGMENT DATA FOR SYMBOLS P L U R H O = BLANK J - . Y N
1791                *
1792 1F78 73383E50         DATA    H'73,38,3E,50,76,5C,48,00,0E,40,80,6E,54'
     1F7C 765C4800
     1F80 0E40806E
     1F84 54
1793                *
1794                *THIS TABLE CONTAINS THE DISPLAY ERROR
1795                *
1796 1F85 170E1313  ERROR  DATA    H'17,0E,13,13,15,13,17,17'
     1F89 15131717
1797                *
1798                *THIS TABLE CONTAINS THE DISPLAY AD=
1799                *
1800 1F8D 170A0D16  ADR    DATA    H'17,0A,0D,16,17,17,17,17'
     1F91 17171717
1801                *
1802                *THIS TABLE CONTAINS THE DISPLAY HELLO
1803                *
1804 1F95 17140E11  HELLO  DATA    H'17,14,0E,11,11,00,17,17'
     1F99 11001717
1805                *
1806                *THIS TABLE CONTAINS THE DISPLAY BP=
1807                *
1808 1F9D 170B1016  BPEQ   DATA    H'17,0B,10,16,17,17,17,17'
     1FA1 17171717
1809                *
1810                *THIS TABLE CONTAINS THE DISPLAY R=
1811                *
1812 1FA5 17171317  REQ    DATA    H'17,17,13,17,16,17,17,17'
     1FA9 16171717
1813                *
1814                *THIS TABLE CONTAINS THE DISPLAY PC=
1815                *
1816 1FAD 17100C16  PCEQ   DATA    H'17,10,0C,16,17,17,17,17'
     1FB1 17171717
1817                *
1818                *THIS TABLE CONTAINS THE DISPLAY F=
1819                *
1820 1FB5 17170F16  FEQ    DATA    H'17,17,0F,16,17,17,17,17'
     1FB9 17171717
1821                *
1822                *THIS TABLE CONTAINS THE DISPLAY LAD=
```

```
LINE ADDR  OBJECT  E SOURCE

1823                *
1824 1FBD 110A0D16  LADEQ  DATA    H'11,0A,0D,16,17,17,17,17'
     1FC1 17171717
1825                *
1826                *THIS TABLE IS THE ASCII LOOK UP TABLE
1827                *
1828 1FC5 30313233  ASCII  DATA    A'0123456789ABCDEF'
     1FC9 34353637
     1FCD 38394142
     1FD1 43444546
1829                *
```

```
LINE ADDR  OBJECT  E SOURCE

1831               ************************************************************
1832               *
1833               *
1834               *USER ENTRY TO DISPLAY ROUTINES
1835               *
1836               *
1837 1FD5 BBFE     USRDSI ZBSR    *MOV    SET UP DISPLAY
1838 1FD7 03             LODZ    R3      GET DISPLAY FLAG
1839 1FD8 BBEC           ZBSR    *DISPLY GO TO DISPLAY ROUTINE
1840 1FDA 17             RETC,UN
```

```
LINE ADDR  OBJECT  E SOURCE

1842               ************************************************************
1843 1FDB                 ORG 8192-26 THE ZBSR OR ZBRR VECTORS ARE HERE
1844               ************************************************************
1845 1FE6 1FD5     USRDSP ACON    USRDSI  USER ENTRY TO DISPLAY ROUTINES
1846 1FE8 1899     ERR    ACON    ERRI    ERROR MESSAGE
1847 1FEA 19E8     BRKPT4  ACON    BRKPTI SET DISBUF6,7 WITH CONTENTS OF R0
1848 1FEC 1E13     DISPLY ACON    DISPLI DISPLAY AND KEYBOARD ROUTINE
1849 1FEE 1EF6     IN     ACON    INN     CASSETTE INPUT ROUTINE
1850 1FF0 1EBA     OUT    ACON    OUTT    CASETTE OUT PUT
1851 1FF2 1C7B     HOUT   ACON    HOUTT   CASSETTE BINARY TO ASCII HEX OUTPUT
1852 1FF4 1A76     DISLSD ACON    DISLSI  CONVERT BYTE TO NIBBLE
1853 1FF6 1BA5     ROT    ACON    ROTI    ROTATE A  NIBBLE
1854 1FF8 1C72     CRLF   ACON    CRLFF   CARRAGE RETURN AND LINE FEED
1855 1FFA 1B3B     GNP    ACON    GNPI    GET NUMBERS
1856 1FFC 1B20     GNPA   ACON    GNPAI   GET NUMBERS AND DISPLAY
1857 1FFE 1DB6     MOV     ACON    MOVI    MOVE DATA TO DISBUF
1858               ************************************************************
1859 1800                 END     SAVRG

 TOTAL ASSEMBLY ERRORS = 0000
```

APPENDIX D — ASCII CONVERSION TABLE

ACSII CHARACTER SET (7-BIT CODE)									
L.S. CHAR \ M.S. CHAR		0 000	1 001	2 010	3 011	4 100	5 101	6 110	7 111
0	0000	NUL	DLE	SP	0	@	P		p
1	0001	SOH	DC1	!	1	A	Q	a	q
2	0010	STX	DC2	"	2	B	R	b	r
3	0011	ETX	DC3	#	3	C	S	c	s
4	0100	EOT	DC4	$	4	D	T	d	t
5	0101	ENQ	NAK	%	5	E	U	e	u
6	0110	ACK	SYN	&	6	F	V	f	v
7	0111	BEL	ETB	'	7	G	W	g	w
8	1000	BS	CAN	(	8	H	X	h	x
9	1001	HT	EM	)	9	I	Y	i	y
A	1010	LF	SUB	*	:	J	Z	j	z
B	1011	VT	ESC	+	;	K	[	k	{
C	1100	FF	FS	,	<	L	\	l	|
D	1101	CR	GS	–	=	M	]	m	}
E	1110	SO	RS	•	>	N	↑	n	~
F	1111	SI	US	/	?	O	← or _	o	DEL

APPENDIX E —
DECIMAL TO HEX CONVERSION TABLE

HEXADECIMAL COLUMNS												
6		5		4		3		2		1		
HEX = DEC		HEX = DEC		HEX = DEC		HEX = DEC		HEX = DEC		HEX = DEC		
0	0	0	0	0	0	0	0	0	0	0	0	
1	1,048,576	1	65,536	1	4,096	1	256	1	16	1	1	
2	2,097,152	2	131,072	2	8,192	2	512	2	32	2	2	
3	3,145,728	3	196,608	3	12,288	3	768	3	48	3	3	
4	4,194,304	4	262,144	4	16,384	4	1,024	4	64	4	4	
5	5,242,880	5	327,680	5	20,480	5	1,280	5	80	5	5	
6	6,291,456	6	393,216	6	24,576	6	1,536	6	96	6	6	
7	7,340,032	7	458,752	7	28,672	7	1,792	7	112	7	7	
8	8,388,608	8	524,288	8	32,768	8	2,048	8	128	8	8	
9	9,437,184	9	589,824	9	36,864	9	2,304	9	144	9	9	
A	10,485,760	A	655,360	A	40,960	A	2,560	A	160	A	10	
B	11,534,336	B	720,896	B	45,056	B	2,816	B	176	B	11	
C	12,582,912	C	786,432	C	49,152	C	3,072	C	192	C	12	
D	13,631,488	D	851,968	D	53,248	D	3,328	D	208	D	13	
E	14,680,064	E	917,504	E	57,344	E	3,584	E	224	E	14	
F	15,728,640	F	983,040	F	61,440	F	3,840	F	240	F	15	