USING THE HOP
(DØ Hand Operated Processor)

PREFACE

The DØ Hand Operated Processor (otherwise known as the HOP) is a unique piece of hardware that provides the user with direct, bit-by-bit control over the entire ABLE computer system. When the HALT light is lit on the HOP, the D1 sequencer is disabled. The switches on the HOP may then be used to control the sixteen-bit Bidirectional Data Bus and the 8-bit Address Bus, as well as the READ, WRITE, SYNC and HALT control lines. The user may talk to any device in the system, including registers, accumulators with arithmetic operations, memory, and all I/O devices. The switches may also be used to enter machine language programs into memory for testing.

The HOP is a useful testing tool for new hardware and an excellent debugging tool (if the user is well acquainted with machine language programming). It is also a remarkable teaching tool because it makes visible the internal architecture of the Able computer.

Contents page

CONNECTING THE HOP

WARNING: To prevent shock hazard and to protect internal circuitry, always unplug the Digital Synthesizer before removing connector panel.

1. Disconnect AC power from the Digital Synthesizer.

2. Remove the connector panel. Be careful not to disturb the cables behind the panel.

3. Remove the plastic rods which keep the boards from moving during shipping.

4. Press the DØ interface board into an even-numbered slot in the computer bin. If you have a 21-slot or 32-slot bin, you may plug the interface into any available slot. If you have a 15-slot bin, you may plug the interface into any available slot besides slots 1-4 (the "memory only" slots). Because slots 8-10 are generally used for the register module, sequencer, and MFC cards, that leaves slots 5-7 or 11-15 free.

5. Plug the 20-conductor flat cable coming from the HOP into the connector on the D0 board. When plugging in the cable, be sure the colored (red or blue) conductor faces down.

   You may place the HOP box anywhere within reach in the work area.

6. Reconnect AC power to the Digital Synthesizer.

THE HOP PANEL FUNCTIONS

Take a moment to examine the front panel on the HOP.

At the top of the panel there are three rows of LED's which provide a visual display of the current contents of both the Address Bus and the Data Bus. One bits are represented by lit LED's; zero bits are represented by unlit LED's.

The top two rows represent the contents of the sixteen-bit Data Bus. This data may come from any other device in the system or it may be entered directly from the HOP. The upper half of a sixteen-bit word of data is represented in the top row (labeled MS) and the lower half is represented in the middle row (labeled LS).

The third row represents the contents of the eight-bit Address Bus. Any of the I/O devices in the system, as well as the sixteen registers, may be directly accessed through this bus. In addition, indirect addressing through the registers permits access to any memory address.

Below the LED's is a row of switches used to specify eight-bit binary numbers when the HOP is activated. You flip a switch up (or ON) to indicate a one bit; you flip a switch down (or OFF) to indicate a zero bit.

After you set all eight switches, you press the DEPOSIT button beside one of the three rows of LED's. This will deposit the eight-bit binary number into the Data or Address Bus. (When you are entering a sixteen-bit number, you must set up and deposit one half of the sixteen-bit word at a time.) As soon as you press the DEPOSIT button, the LED's, and the particular bus, will represent the number set in the switches.

There are various other buttons and switches on the HOP that are used in the following way:

The HALT switch is flipped up to halt the D1 sequencer and to transfer control of the system to the HOP box. Sometimes the SYNC button must be pressed to complete the halt and turn on the HALT LED. When the HALT switch is flipped down, control is returned to the sequencer.

The following two buttons will function only when the HALT switch is down, that is, when the processor is <u>not</u> halted.

The LOAD button is used to load the operating system from the floppy diskette in the left-hand disk drive.

The RESTART button is used to reset the program counter to zero.

The next three buttons will function only when the processor is halted. The LED above each button, however, will reflect the status of the related control line at all times.

The WRITE button is used to transmit the data in the Data Bus to the device at the address in the Address Bus.

The READ button is used to receive data from the device at the address in the Address Bus into the Data Bus.

The SYNC button is used to complete the HALT operation.

In addition, a program can read from or write to device 0, in which case execution will be temporarily halted until the user presses the SYNC button to return a SYNC signal to the processor.

Now turn on the computer and load from an XPL or SCRIPT system diskette. The Data Bus LED's should all be off, because there is no data in the Data Bus yet. In the Address Bus row, LED 3 and LED 5 should be lit representing Device "50", the terminal address in octal notation. The READ button should also be lit. The system is waiting for input from the terminal. If you press a key on the terminal, the ASCII code for the appropriate character will be represented in the Data Bus LED's.

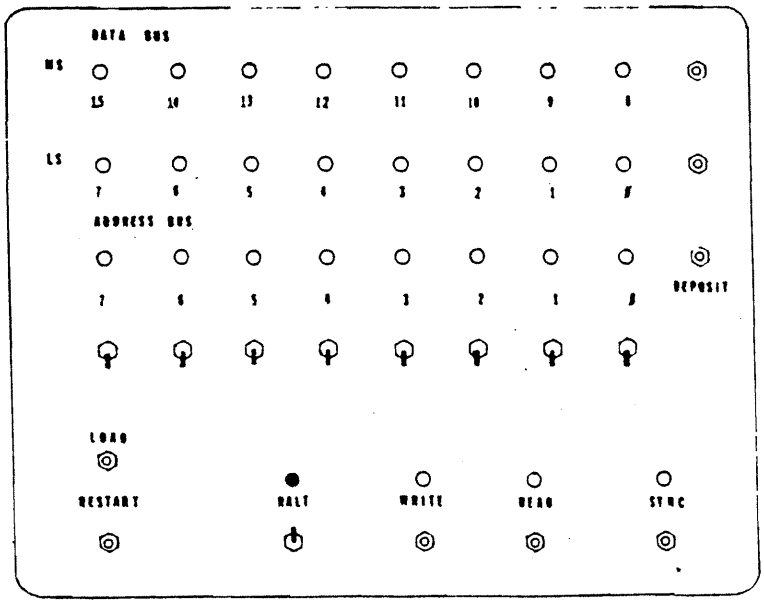
When a program is executed under the control of the D1 sequencer, all of the LED's for the bus lines, as well as those for the control lines, appear to be lit continuously because of the speed of the processing. When the HOP is activated, and you are controlling the processing manually, you can observe each device address and each word of data and instruction.

ACTIVATING THE HOP

To activate the HOP, flip the HALT switch up (or ON) and press the SYNC button if necessary until the HALT light is lit. At this point, the D1 sequencer is disabled. By manipulating the switches and buttons on the HOP, you may now act as the processor.

When you wish to return to the XPL monitor, lower the HALT switch and press the LOAD button.

Before proceeding with instructions on operating the HOP, we have provided in the next chapter a short review of octal and hexadecimal numbering.



NUMBER SYSTEMS REVIEW

When you use the HOP, you must frequently convert octal or hexadecimal numbers into binary numbers. All device addresses in the Able computer system are specified in octal, while data and instruction words may be specified in either octal, "split" octal, or hexadecimal. While the choice of notation depends on the taste of the programmer, we use split octal notation in the exercises in this manual because it is easy to use with the two rows of the Data Bus. This chapter provides a quick review of how to convert octal or hexadecimal into binary.

In this manual, as in Scientific XPL, octal numbers are enclosed in quotation marks ("177777"). In split octal, the notation for the most significant eight bits appears first followed by the notation for the least significant eight bits. The two numbers are each enclosed in quotation marks and separated by commas ("377", "157"). Hexadecimal numbers are also enclosed in quotation marks and the letter H appears immediately after the left-hand quotation mark ("H0001").

OCTAL

Octal is merely a shorthand form of binary. In octal, the digits one through seven are used to represent three binary digits.

binary		octal
0 0 0	=	"0"
0 0 1	=	"1"
0 1 0	=	"2"
0 1 1	=	"3"
1 0 0	=	"4"
1 0 1	=	"5"
1 1 0	=	"6"
1 1 1	=	"7"

To convert an octal number into a binary number, start from the right and change each octal digit into its three-digit binary equivalent. Run the resulting binary groups into a single eight- or sixteen-bit binary number.

Examine the following eight-bit numbers.

"50" = 0 0 1 0 1 0 0 0

0 5 0

"101" = 0 1 0 0 0 0 0 1

1 0 1

On the HOP, sixteen-bit numbers must be entered in two 8-bit halves. In octal these numbers may be specified in two different ways.

If a single octal number (e.g., "177557") is used for the entire sixteen-bit word, the groups of three binary digits fall as below.

"177557" = MS 1 1 1 1 1 1 1 1

1 7 7

LS 0 1 1 0 1 1 1 1

5 5 7

In split octal, the sixteen-bit word is "split" into two octal numbers ("377", "157").

"377" = MS 1 1 1 1 1 1 1 1

3 7 7

"157" = LS 0 1 1 0 1 1 1 1

1 5 7

As you can see, this form of notation corresponds more easily to the two rows of LED'S for the Data Bus.

HEXADECIMAL

In hexadecimal, another shorthand form of binary, the digits one through nine plus the letters A through F are used to represent four binary digits.

0 0 0 0	=	"H0"
0 0 0 1	=	"H1"
0 0 1 0	=	"H2"
0 0 1 1	=	"H3"
0 1 0 0	=	"H4"
0 1 0 1	=	"H5"
0 1 1 0	=	"H6"
0 1 1 1	=	"H7"
1 0 0 0	=	"H8"
1 0 0 1	=	"H9"
1 0 1 0	=	"HA"
1 0 1 1	=	"HB"
1 1 0 0	=	"HC"
1 1 0 1	=	"HD"
1 1 1 0	=	"HE"
1 1 1 1	=	"HF"

To convert a hexadecimal number into a binary number, start from the right and change each hexadecimal digit into its four-bit binary equivalent. Run the four-bit groups together into a single eight or sixteen-bit word.

Examine the following:

"H01" = 0 0 0 0 0 0 0 1
(0 0 0 0 = 0; 0 0 0 1 = 1)

"HFFFF" = MS 1 1 1 1 1 1 1 1
(1 1 1 1 = F; 1 1 1 1 = F)

LS 1 1 1 1 1 1 1 1
(1 1 1 1 = F; 1 1 1 1 = F)

Be sure you clearly understand these examples before proceeding. Remember, on the HOP, a binary 1 is indicated by a switch in the UP or ON position and a lit LED.

WRITING AND READING

WRITING TO THE TERMINAL

To get a feeling for HOP operation, you will first perform a simple write from the HOP to the terminal. You will use the switches and the DEPOSIT button to input the device address of the terminal into the Address Bus. Next you will input the ASCII code for the letter A into the Data Bus. Finally you will press the WRITE button to perform the data transmission.

The terminal is connected to the computer through the D50 terminal interface. The device address of the terminal is "50".

1. First make sure the HALT LED is lit. If it isn't, push the HALT switch up and press the SYNC button.

2. Raise switch 3 and switch 5 to represent octal "50". All other switches should be down.

3. Press the DEPOSIT button at the end of the Address Bus row of LED's.

   The appropriate LED's should come on immediately. The Address Bus is now set for device "50".

4. Now set the switches to represent the character upper case A. Since A in the ASCII character code is "101", raise switch 0 and switch 6.

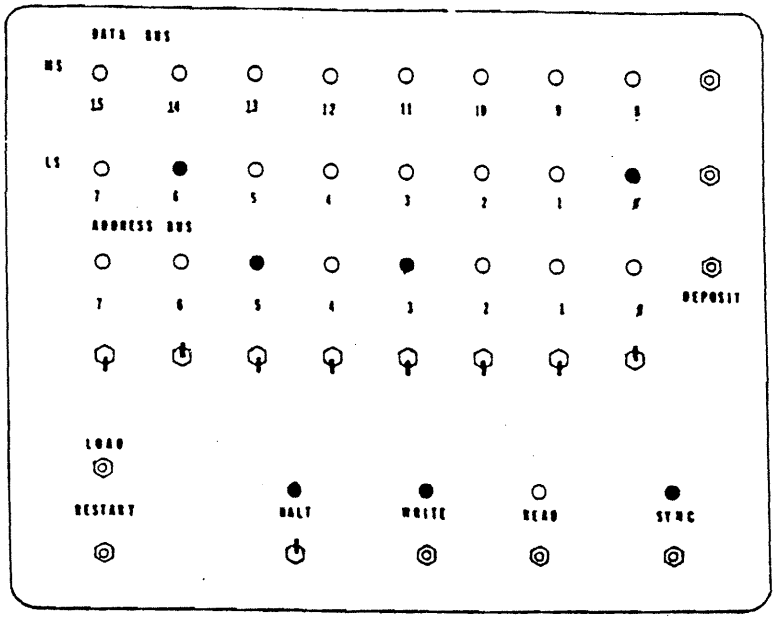
5. Press the DEPOSIT button at the end of the LS half of the Data Bus.

   (The upper half of the Data Bus is ignored when ASCII characters are transmitted to and from D50. Therefore, in this particular case, it doesn't matter how the LED's in the top row are set up.)

6. Once the Address Bus and the Data Bus have been correctly set up, press the WRITE button.

   Data will be transferred from the Data Bus into the D50 interface. The interface will respond by lighting the SYNC light. The SYNC light will remain lit as long as the WRITE button is depressed. When you release the WRITE button, the character will appear on the terminal.

Writing to the Terminal



READING FROM THE TERMINAL

To read from a device in the system, you simply place the address of the device into the Address Bus and press the READ button.

Thus, to receive a character from the terminal keyboard you follow this procedure:

1. Leave address "50" in the Address Bus.

2. Clear the Data Bus by pushing all switches down or OFF and pressing the DEPOSIT buttons for the top two rows of LED's. You will frequently repeat this procedure in the exercises that follow.

3. Press the Z key (for lower case Z) on the terminal.

4. Now press the READ button on the HOP. As the button is pressed, the READ and SYNC LED's will be lit and the LS Data Bus will display the ASCII code for lower case Z. The LED's representing bits 1, 3, 4, 5, and 6 in the Data Bus will be lit.

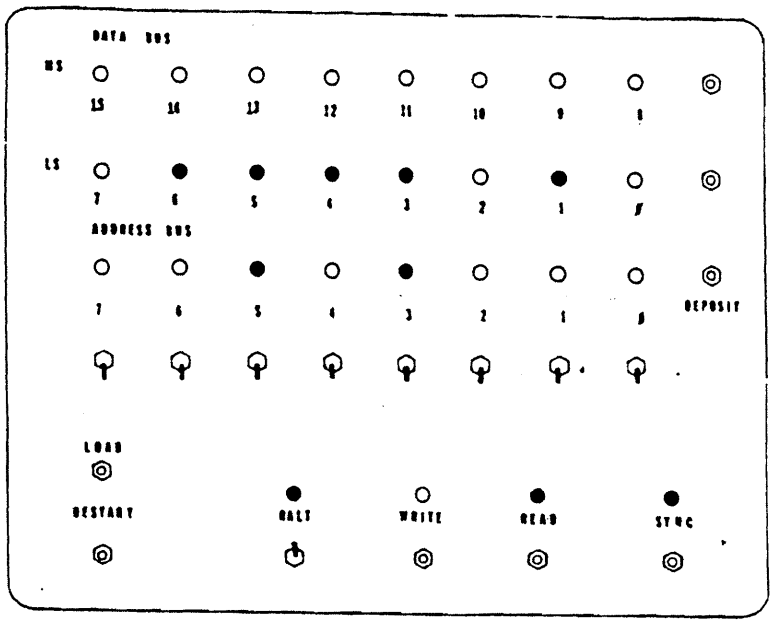
   When you release the READ button, the LED's in the the

READ and SYNC LED's as well as the LED's in the Data Bus will go out.

You can hold down the READ button and continue to press keys on the terminal. The LED's in the Data Bus will represent each new letter.

Practice sending to and receiving from the terminal until you are comfortable with the switches. A table of the ASCII character code may be found in Appendix I.

Reading from the Terminal



ADDRESSING MEMORY

In the Able computer, a location in memory cannot be addressed directly. Instead it is addressed indirectly through any one of sixteen registers. A sixteen-bit memory address is stored in a register where it is used as a pointer to the memory location.

Before you proceed with the exercise, note that the eight bits that make up an Able register address have the following meaning:

Bits 0 - 3 indicate the number of the register from "0" to "17".

Bit 4 is the increment bit. When this bit is set at one, the contents of the register will be automatically post-incremented by one each time a write or a read is performed to that address.

Bit 5 is the indirect bit. When this bit is set at one, the contents of the register are used as a sixteen-bit pointer to a location in memory. When this bit is set at zero, the register itself is addressed.

Bits 6 and 7 are set as ones to identify the address as a register address.

The following exercise will show you how to to access the first location in memory through register "17".

Remember, if you wish to return to the XPL monitor at any time, lower the HALT switch and press the LOAD button.

1. Set up the Address Bus with register address "317".

   a. Set the switches as follows:

7	6	5	4	3	2	1	0
on	on	off	off	on	on	on	on

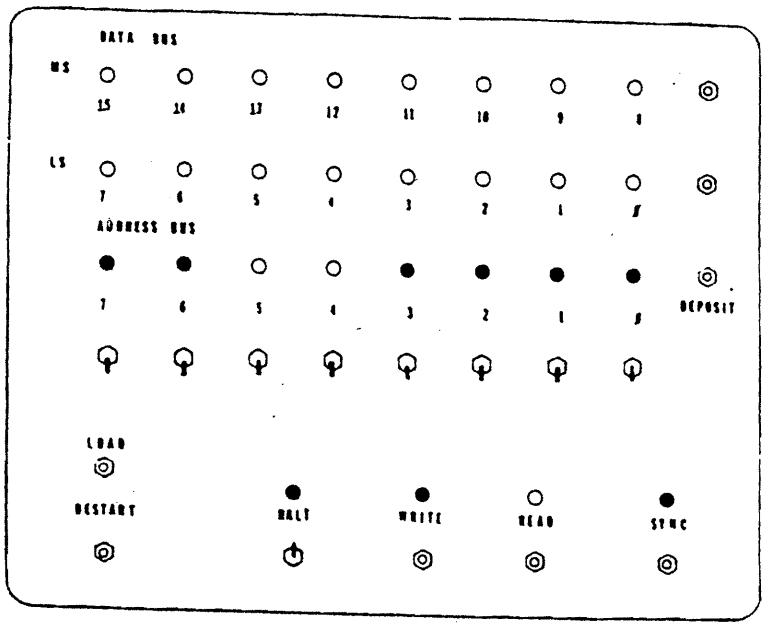
   b. Press the Address Bus DEPOSIT button.

2. Set up the Data Bus to represent memory address "000", "000", the first location in memory. Simply clear the Data Bus by setting all the switches in the OFF position and pressing the LS and MS DEPOSIT buttons as described in the previous exercise.

3. Press WRITE to place the number "000", "000" into

register "17".

Storing "000","000" in R "17"



4. Press READ to verify the contents of register "17".

   The LED's should remain unlit. The SYNC and READ lights should remain lit as long as you press READ.

5. Now change the Address Bus so that the indirect bit is on.

   a. Reset the switches for the address of register "17" as above only this time raise the switch for the indirect bit (bit 5). Thus, the switches should be set as follows:

   7 6 5 4 3 2 1 0
   on on on off on on on on

   b. Press the DEPOSIT button for the Address Bus.

6. Press READ.

   You are now looking at the contents of memory address "000", "000" rather than at the contents of register "17".

   If you have loaded the XPL4 monitor before using the HOP, the contents of this memory address wll be zero and and the lights in the Data Bus should remain unlit. If you have loaded any other XPL monitor, the contents of this memory address will be "200", "377". The LED'S for bits 0-7 and 15 should be lit.

   In all cases, the READ and SYNC buttons should be lit. If any of these things are not so, either you have set up the wrong address or the device isn't working properly.

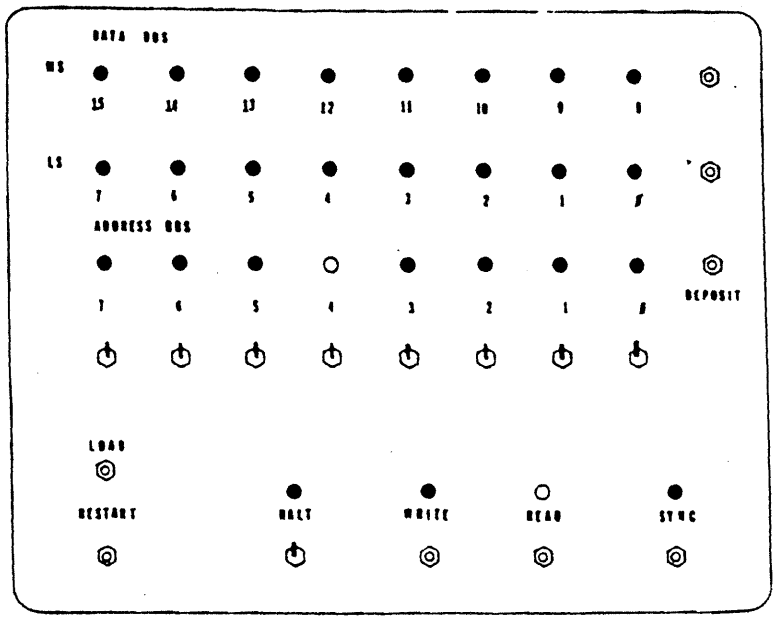
7. Place the number "377", "377" in the Data Bus.

   a. Set all the switches in the ON position.

   b. Press the DEPOSIT buttons by the LS and MS rows of the DATA BUS.

8. Press WRITE to send this data to memory address "000", "000".

Writing to Memory Address Stored in R "17"



9. Again, clear the Data Bus by setting the switches to the OFF position and pressing the LS and MS DEPOSIT buttons.

10. Press READ.

    The sixteen LED's should all light up. The number "377", "377" is stored in memory address "000", "000".

Increment Bit

The increment bit in the register address makes it possible to read from or write to successive memory addresses without having to change the contents of the register each time. Thus, when this bit is set to one, every time you press WRITE or READ the contents of the register, that is, the memory address, will be incremented by 1. You can write the same data to multiple locations or you can change the contents of the Data Bus before you press WRITE. Or you can read from consecutive locations in memory.

1. Now change the Address Bus so that the increment bit is on.

    a. Reset the switches for the address of register "17" as in step 5 above, only this time raise the switch for the increment bit (bit 4). All the switches should be set in the ON position.

    b. Press the DEPOSIT button for the Address Bus.

2. Leave the Data Bus as in the previous exercise with all LED's lit.

3. Press the WRITE button.

    You are writing "377", "377" to address "000", "000" and incrementing the memory address in the register to "000", "001".

4. Press the WRITE button again.

    You are writing "377", "377" to address "000", "001" and incrementing the memory address in the register to "000", "002".

5. Press the WRITE button a third time.

   You are writing "377", "377" to address "000", "002" and incrementing the memory address in the register to "000", "003".

Now, let's see if the memory address stored in register "17" has in fact been incremented.

1. Set the increment and indirect bits in the Address Bus to zero so that you can read the contents of the register itself.

   a. Set the switches as follows:

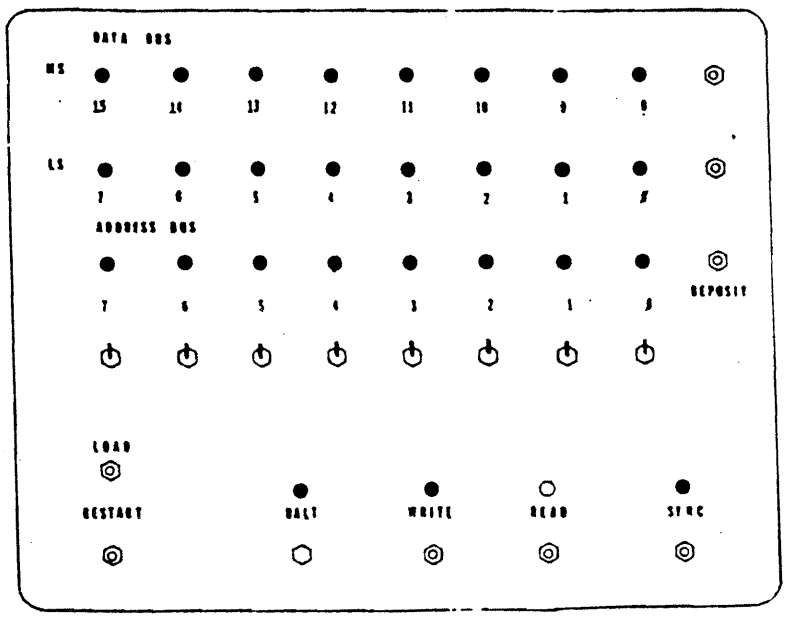
7	6	5	4	3	2	1	0
on	on	off	off	on	on	on	on

   b. Press DEPOSIT on the Address Bus.

2. Now press READ.

   The LED's for bit 0 and bit 1 in the middle row will light up indicating the number "000", "003". The contents of the register have been post-incremented three times.

Writing to Memory and Post-Incrementing



ENTERING A MACHINE LANGUAGE PROGRAM

With the techniques you have just mastered, you can enter a machine language program. In this chapter you will enter a very short program that will cause the computer to write continuously to the terminal and fill the terminal screen with upper case A's.

The program will simply transfer the contents of a register to the terminal interface. There will be two lines of code, or two sixteen-bit instruction words. Each sixteen-bit instruction word in memory consists of two parts. The least signicant half consists of an address for a data source and the most significant half consists of a data destination.*

The first instruction word asks the computer to transmit the contents of register "0" (source) to device "50" (destination). The instruction word is "50", "300".

```
MS   0 0 1 0 1 0 0 0  Device "50"
        5     0
LS   1 1 0 0 0 0 0 0  Register "0"
      3    0     0
```

The data in register "0" will be the ASCII code for the letter A.

The second instruction takes an immediate source** with a value of zero and transmits this value to register "17" (destination). This instruction word is "317", "200".

```
MS  1 1 0 0 1 1 1 1  Register "17"
     3    1     7
LS  1 0 0 0 0 0 0 0  Immediate source = 0
     2    0     0
```

Since register "17" is the program counter, this instruction causes the program to jump back to 0 in an infinite loop.

* See the ABLE Series Hardware Reference Manual for more informatin on the formatting of instruction words.

** An immediate source provides a six-bit, two's complement number in bits 0-5 of the instruction word itself.

First fill register "0" with the ASCII code for upper case A.

1. Place "300" in the Address Bus

2. Place "101" into the Data Bus.

3. Press WRITE.

Next enter the two lines of code into memory.

1. Place "317" in the Address Bus.

2. Clear the Data Bus.

3. Press WRITE.

   You have just stored memory address "000", "000" in register "17".

4. Place "377" in the Address Bus.

5. Place "50", "300" in the Data Bus.

6. Press WRITE.

   You have just stored the first instruction word in the memory location at "000", "000" and post incremented the address in register "17".

7. Place "317", "200" in the Data Bus.

8. Press WRITE.

   You have just stored the second instruction word in the memory location at "000", "001".

9. Place "317" in the Address Bus.

10. Clear the Data Bus.

11. Press WRITE.

    You have just set the program counter to zero. Another way to do this is to press the RESTART switch.

12. Clear the Address Bus.

Now run the program by lowering the HALT switch. The

terminal should fill with A's.

The WRITE and SYNC lights should be on and the HALT light should be off.

WHEN THE MACHINE CRASHES.....

A "crash" is a general term for breakdown - the system has become inoperative, lost in a loop, or failed in some way. The crash can be due to a failure in the computer or in some peripheral device. We call a breakdown due to malfunction in the computer hardware or software a "machine crash".

Frequently a machine crash is caused when a non-existant device or register is addressed in either the source or destination half of an instruction word. At New England Digital, we use the HOP to track down problems of this nature. If you are well acquainted with machine language programming, you may wish to use the HOP to do some of your own trouble shooting. You can use the following approach.

You can usually tell a machine crash by looking at the SYNC light. If the SYNC light is lit, the computer is definitely running although the code it is executing may not be the program you expect. If the SYNC light is not lit, the computer may have crashed. However, in some cases (i.e., during continuous writes to the Device "003" or certain accesses to the Winchester disk) the SYNC light appears to be unlit and the computer is running, or legitimately waiting for a device to return SYNC.

Let us assume that you have determined that the machine has crashed. First raise the HALT switch and examine the contents of register "17", the program counter. If that memory address is clearly incorrect, for example if it were greater than the amount of memory in your system, you have already discovered a symptom of the problem.

Usually, however, the address in the program counter would have been post-incremented after a write or read to an incorrect address. You will have to look at an earlier instruction in the program to find the possibly faulty code. You also have to determine the sequence of instructions before you can know what is wrong. Since instruction words and data words may follow one another in memory, you must look through several words in memory, both before and after the memory address at the time of the crash, to determine the sequence.

To do this, you subtract one from the memory address in the program counter, set the earlier memory address in the program counter, and examine its contents using the indirect bit. If that memory location contains an instruction word, write down the instruction, subtract again from the program counter, and look at the next earlier instruction. Continue

backward a few words.

If the contents of the first location are data, you subtract again, and look at the location before it. This location should contain an instruction word. Write it down, subtract two from the counter (to skip over the next data word) and look at the earlier instruction word. Continue backward a few instruction words.

You can then use the increment bit to look forward at the contents of the memory locations after the crash until you can determine the sequence of instructions and, hopefully, find the error. Note that the process may be further complicated by the fact that there may be jumps from some other location in memory that cannot be determined.

These comments merely get you started and do not begin to cover the complex problem-solving process. However, if you can learn to use the HOP to determine machine conditions or states at the time of a crash, you will be able to provide valuable clues to an experienced New England Digital technician and thereby aid in fixing the problem.

APPENDIX I. ASCII CODE CONVERSION TABLE IN OCTAL

000	NUL	020	DLE	040	SP	060	0	100		120	P	140		160	p
001	SOH	021	DC1	041	!	061	1	101	A	121	Q	141	a	161	q
002	STX	022	DC2	042	"	062	2	102	B	122	R	142	b	162	r
003	ETX	023	DC3	043	#	063	3	103	C	123	S	143	c	163	s
004	EOT	024	DC4	044	$	064	4	104	D	124	T	144	d	164	t
005	ENQ	025	NAK	045	%	065	5	105	E	125	U	145	e	165	u
006	ACK	026	SYN	046	&	066	6	106	F	126	V	146	f	166	v
007	BEL	027	ETB	047		067	7	107	G	127	W	147	g	167	w
010	BS	030	CAN	050	(	070	8	110	H	130	X	150	h	170	x
011	HT	031	EM	051	)	071	9	111	I	131	Y	151	i	171	y
012	LF	032	SUB	052	*	072	:	112	J	132	Z	152	j	172	z
013	VT	033	ESC	053	+	073	;	113	K	133	[	153	k	173	{
014	FF	034	FS	054	,	074	<	114	L	134		154	l	174	|
015	CR	035	GS	055	-	075	=	115	M	135	]	155	m	175	}
016	SO	036	RS	056	.	076	>	116	N	136	^	156	n	176	~
017	SI	037	US	057	/	077	?	117	O	137		157	o	177	DEL

APPENDIX III. LOOKING AT THE SYNCLAVIER (R) II KEYBOARD UNIT

The HOP can be used to manually check the functioning of the buttons and keys on the Synclavier (R) II keyboard unit as well as the segmented digital displays and small LED's. You communicate with the keyboard unit through two registers on the SK1 or SK2 board, the data register ("130") and the control register ("131").

THE BUTTONS

The buttons are the easiest to check.

The buttons are divided into eight panels of eight buttons each. The eight panels are numbered as follows:

8	10	12	14
9	11	13	15

To select one of these panels, you write a control word to address "131". The bits of the control word have the following meaning: Bits 0 - 3 are used to specify the panel number from 8-15. Bit 4 is a display enable bit. It is set at one in such software as the Synclavier (R) II run-time package to enable automatic scanning through all keyboard I/O addresses. In HOP communication with the keyboard unit this bit should be set at 0. Bit 5 is set at one to enable writes to the buttons. Bit 6 is set at one to enable reads from the buttons.

Once you have selected a panel by writing to address "131", you write to and read from address "130" to communicate with individual buttons. The sixteen buttons of each panel are represented by the sixteen bits of the data word. The buttons are numbered from 0-15 from upper left to lower right in each panel. Thus, LED 0 in the Data Bus on the HOP represents the upper leftmost button in each panel and LED 15 represents the lower rightmost button in the panel.

Writing to Panel #8 (ENVELOPE GENERATORS)

1. Deposit "131" in the Address Bus.

   This is the address of the control register on the SK1 or SK2 card.

2. Deposit "000", "150" in the Data Bus.

   This enables reads and writes to panel #8.

3. Press WRITE.

4. Deposit "130" in the Address Bus.

   This is the address of the data register on the SK1 or SK2 card.

5. Deposit "377", "377" in the Data Bus.

6. Press WRITE.

   All the buttons in the ENVELOPE GENERATORS panel should light up. Note that this is a static write. The LED's will appear much brighter than they normally do.

7. Clear the Data Bus.

8. Press WRITE.

   All the buttons in the panel should become unlit.

9. Now practice lighting individual buttons by setting the appropriate bits to ones. You can light any combination at a time. Remember, the buttons and the HOP LED's are numbered in reverse order.

   End your experimentation with all the buttons unlit.

Reading From Panel #8

1. Now hold down the READ button on the HOP and press each of the buttons in the panel.

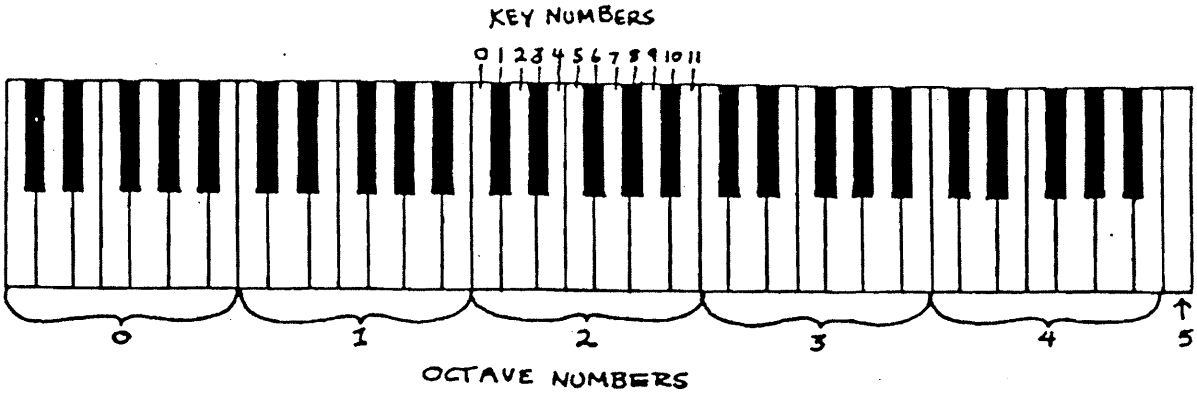
   Each button should light (brightly) for as long as you hold it down and the corresponding bit on the HOP should do the same.

   Pushing more than two or three buttons at one time causes an unstable condition to occur on the Data Bus. The buttons on the control panel may light but the LED's on the HOP may not. This is a NORMAL condition.

   Pushing any buttons in the panel below, or in any other panel, should have no effect. Any buttons which light the buttons in other rows are probably shorted.

READING FROM THE KEYBOARD

You can read input from the keyboard in much the same way. The keyboard is divided into five 12-note octaves plus a sixth "octave" with only one note - the highest C.



To select an octave, you write a control word to address "131". The bits of this word have the same meanings as those in the control word for the button panels, except bits 0-3 are used to indicate the octave number from 0-5. Since you can only read from the keyboard, bit 5 should be set at 0. Bit 6 should be set at 1.

Once you have selected an octave by writing to address "131", you read from address "130" to receive input from the keyboard. The keys are numbered from 0 to 11 from left to right. The LED's for the Data Bus represent the keys of the same number. As soon as you press READ, all LEDs from 0-11 should become lit since the keys are always grounded. When you press a key, it is ungrounded and the LED corresponding to that key should become unlit.

Reading from the Lowest Octave

1. Deposit "131" in the Address Bus.

2. Deposit "000", "100" in the Data Bus.

   This enables reads from the lowest octave on the keyboard.

3. Press WRITE.

4. Clear the Data Bus.

5. Press READ.

6. LED'S 0 - 11 should light up.

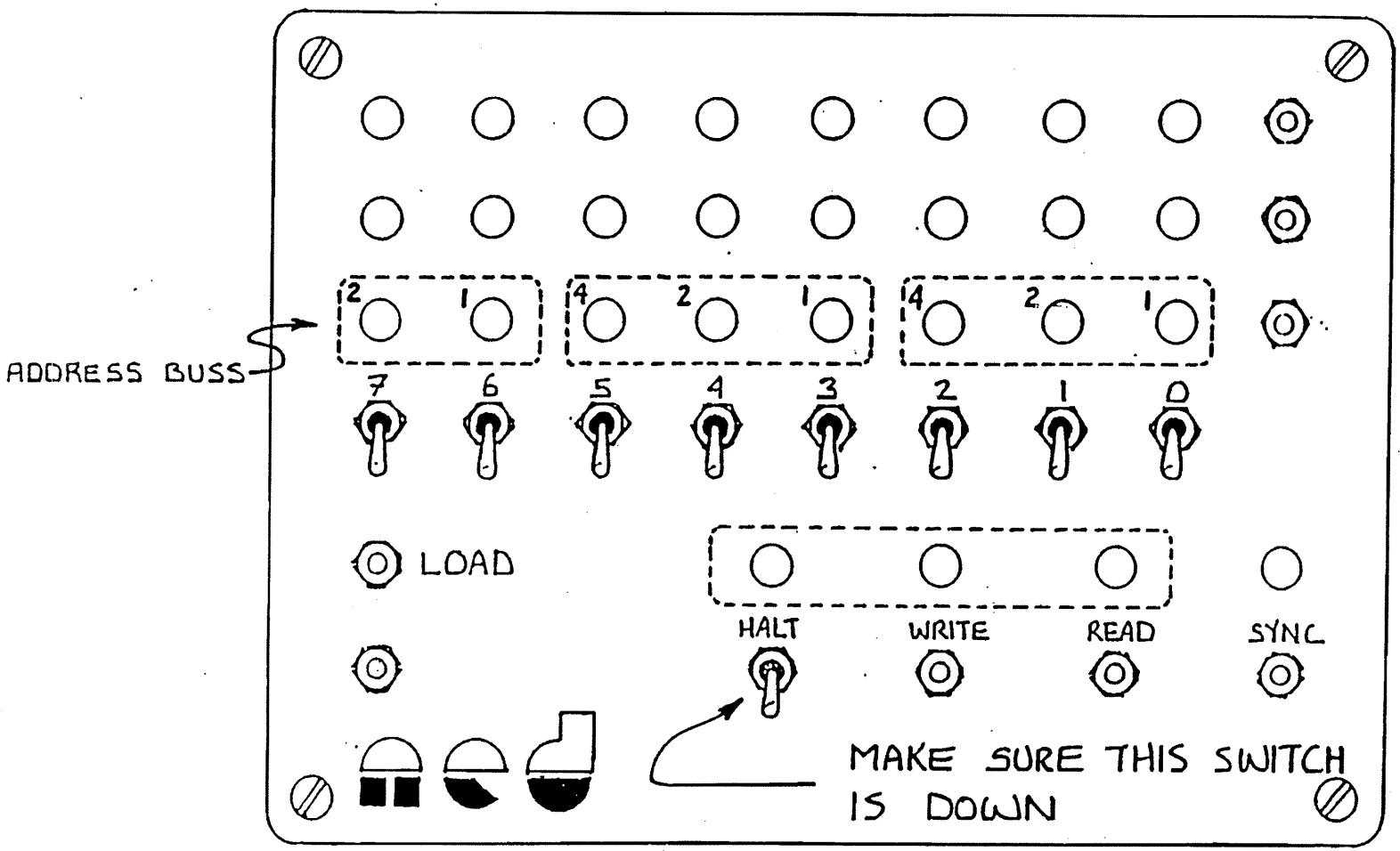
7. Now press each key in the lowest octave and observe the LED's.

8. Experiment by reading from different octaves on the keyboard.

DATE / /

TIME _____

THE HOP



INSTALLING THE HOP

- PLACE THE HOP CARD (D01) IN ANY EMPTY SLOT OF THE COMPUTER BIN.
- IF A SLOT ISN'T AVAILABLE PULL A NON-ESSENTIAL BOARD (MI70, D-40 SMPTE, D134, D30TD)
- NEXT BOOT SYSTEM AND USE

READING THE HOP~WHEN SYS'M IS CRASHING

- PUT AN "X" IN EACH CIRCLE ABOVE (W/ DASHED ENCLOSURE) IF LIGHT IS ON.
- BOOT SYSTEM & GET IT TO CRASH AGAIN. «« KEEP TRACK OF ALL CRASH PATTERNS »»
- CALL N.E.D. SERVICE TECH W/ RESULTS

System Device Configuration Map - ABEL Computer System
Not All PCB/Device addresses are shown

DEVICE "1"

Bit 0 Z Flag
Bit 1 C Flag
Bit 2 M Flag
Bit 3 I ON Bit
Bit 4 D43 Real Time clock / D137 Clock Calendar
Bit 5 D40 Modem Controller / D136 clock
Bit 6 D50 Terminal Interface
Bit 7 D16 Scientific Timer
Bit 8 D03 Real Time Clock
Bit 10 Interrupt REQ -> Read "54" for more info
Bit 9,12,13 Undefined (?)
Bit 11 DSP70 Interrupt REQ -> Read "55" for more
Bit 14 User Defined
Bit 15 User Defined

DEVICE "3"
Bit 0 = 0 Flag not Set
= 1 Flag is Set
"H8000" -> Write("3") = Reset IEN
"HC000" -> Write("3") = Set IEN

DEVICE "4 "= LOAD
DEVICE "5" = MULTIPLY
DEVICE "6" = DIVIDE
DEVICE "7 "= RESULT
DEVICE "10"RPC (Repeat Counter)
DEVICE "11"(Model D) Memory page register

DEVICE "51"
"H0100" D70 Exists
"H0200" External Memory Exists
"H0400" Poly Exists
H0800" D32X Exists
"H1000" D100A Exists
"H2000" D110A Exists
"H4000" D24 Exists
"H8000" D57 Exists ->Read D57 for more Bits

DEVICE "24"
READ("24") Bits8-11 = D24 ID# selected/interrupting
Bits13-15 = D24 Revision #
Bit 12 = Unused = 0
Bit 0-7 = SCSI Data Bits D0-D7
WRITE("24") Bits 8-11 = Select D24 ID #
Bit 12 = Select Enable
Bits 0-7 = SCSI Data bits D0-D7

DEVICE "25"
Interrupt Enable/Arbitrate Config. are Bits9-11
SCSI Signal / Control lines are Bits 0 - 8
Bits 12-15 are unused

DEVICE"26"
READ/WRITE 1 Data Byte with REQ/ACK Hndshke
MSB always 0 or ignored

DEVICE "27"
READ/WRITE 1 Data Word with REQ/ACK Hndshke
MSB sent/recv'd first, then LSB to/from SCSI Bus

DEVICE "54"
"H0001" Interrupt on D40
"H0002" Interrupt on D42
"H0004" Interrupt on D43
"H0008" Interrupt on D3
"H0080" Interrupt on D24

DEVICE "55"
"H0010" Interrupt on DSP70

DEVICE "70"
Write("70") = "H0008" SMPTE Select
Write("70") = "H0010" MIDI Select
Write("70") = "H0020" DSP Select
Bits 0,1 are for Board/subsystem Select/ID #0-3
DSP70 Interrupts READ("70") ->"H0021" thru "H0023"

DEVICE "57"
"H0001" Model D processor Exists
"H0002" D40Q Exists
"H0004" M64K Exists -> Implies D16 Timer, D136. D137
"H0010" D115D Comm. Processor Exists

DEVICE "60" = MEMORY ADDRESS M.S.
DEVICE "61" = MEMORY ADDRESS L.S.
DEVICE "62" = MEMORY DATA
DEVICE "63" = MEMORY DATA W/INCREMENT

POLY DEVICE ADDRESSES:
DEVICE "154" Extended Data
DEVICE "155"PS Channel Number
DEVICE "156" PS Function Code
DEVICE "157" Data

Device Address Assignments

00	HOP
01	Processor status/control word
02	Priority encoder (Model C)
03	D3 200 Hz real-time clock
04-07	D4567 multiply/divide unit
10	Source: last PC, Destination: repeat counter (Model C)
11	Page register (Model D)
12-13	12-bit ADC
14-15	Second 12-bit ADC
16-17	D16 scientific timer
20-27	Reserved for Dartmouth (sort of)
24-27	D24 SCSI host adapter
30-37	Digital I/O modules
	30 Kennedy tape drive interface
	31-32 Multichannel
40-47	Async serial or parallel printer ports
	40-41 Printer
	42-43 Modem
	44-45 Mouse
50-51	D50 terminal interface; system ID bits in upper half of 51
52-53	*** Unassigned ***
54-56	Additional interrupt bits
57	Additional system ID bits
60-63	M128K/M512K external memory
64-65	256 12-bit DACs
66-67	D66 buffered ADC/DAC
70-73	MPU interface
	70 Interface select (bit 3: SMPTE, bit 4: MIDI)
74-77	*** Unassigned ***
100-104	Floppy disk interface (F0/F1)
105-107	IMI Winchester disk interface
110-114	Remote floppy disk interface (R0/R1)
115-117	*** Unassigned ***
120-127	Reserved for user devices (sort of)
126-127	Z80 board
130-131	SK2 clavier interface
132-133	Device address expansion interface (tentative)
	132 Unit select (0-7: computer bus switch)
134-135	Digital guitar interface
136	Second 200 Hz real-time clock (sync'd to poly's 50 MHz)/D16 int contr
137	Clock/calendar/D16 rate control
140-143	Communications processor
144-147	*** Unassigned ***
150-153	X.25 interface
154-157	Polyphonic sampling interface
160-164	FM synthesizer interface
165-167	*** Unassigned ***
170-177	Reserved for test modules

bit 5: 56000 D

```
      here are some possible guesses for trouble shooting: */
54
55 /* command = 0,  status = -4          scsi bus contains nonzero
56                                       bits.  Reset would not
57                                       clear them.  Probable
58                                       scsi cable in backwards,
59                                       short, termination resistor
60                                       problem, or bad hardware */
61
62 /* command =18,  status = -2          target did not respond.
63                                       probably means loose cable,
64                                       bad power cable, bad
65                                       power supply, bad drive */
66
67 /* command =18,  status = -11         LUN did not respond.  means
68                                       bad configuration, loose
69                                       cable to one LUN, or defunct
70                                       drive */
71
72
73 /* command = 0,  status = -13         no voice card exists for that
74                                       track.  this indicates a problem
75                                       in the poly synth.  voice card
76                                       or amplitude computer may be
77                                       missing, unplugged, or defective */
78
79
80 /* command = 0,  status = -14         no memory exists for that track.
81                                       this indicates an incorrect
82                                       lod system configuration, or
83                                       problems in the poly unit.
84                                       512 sectors of poly memory
85                                       are required for each track */
86
87 /* command = 0,  status = -15         could not write a directory to
88                                       the disk.  Wrote the data out, got
89                                       a good status, but read different
90                                       data back.  Most likely a bad d24,
91                                       disk controller, or disk drive */
92
93 /* command = 0,  status = -17         the number of sectors available for
94                                       recording on this track is not
95                                       equal to the number of sectors available
96                                       for recording on another track.
97                                       this most likely means one of
98                                       the disk drives for this track
99                                       did not respond. */
100
101
```

Title : Diagnostic Diskette Documentation
Project Engineer: Bill Leathers
Date : 08/01/90
New England Digital Corporation

This document describes the diagnostics on the Release 2.4 diagnostic diskette supplied to the customer. For each diagnostic there is a brief description and operating instructions.

The following is a list of the programs available on the Main Menu:

REL. 2.4 DIAGNOSTIC DISKETTE MAIN MENU

A. CONFIGUR	B. DTDMENU
C. AUDIOIO	D. DSP70TST
E. EXTMEM	F. FLOPTEST
G. GRAPHTST	H. KEYBOARD
I. LOGGER	J. MAINMEN
K. MEMBURN	L. MIDITEST
M. MULTICHN	N. PMCUTEST
O. PRINTEST	P. PSXMBTE
Q. SMPTETST	R. STMADJ
S. TAPETEST	T. THRUPUT
U. WINBOOT	

CONFIGUR - Use this program to configure the diagnostic diskette to match your system. The diskette is originally configured for a MG600 (Pericom) terminal. If you do not have a MG600, the terminal screen may be garbled when booted. CONFIGUR is selected by default when the disk is first booted, so just press <RETURN> to re-configure the diskette.

DTDMENU - This program provides a menu of diagnostics that can be down-loaded to the Direct-To-Disk. When DTDMENU is executed, the following DTD diagnostic menu is displayed.

DIRECT-TO-DISK DIAGNOSTIC MENU

A. AUDIOIO	B. EXTMEM
C. LOGGER	D. MAINMEM
E. MEMBURN	F. METERTST
G. MIDITST	H. MITTEST
I. PSXMBT	J. STARTUP
K. STMADJ	L. THRUPUT
M. UDIO2TRK	

DTD Terminal Type: MG600

Specify the DTD diagnostic terminal type using <SPACE BAR>, select a program by letter or arrow keys, and press <RETURN> when ready.

<SPACE BAR> - Change terminal type <BREAK> - Exit

These diagnostics require that a terminal be connected to the DTD diagnostic terminal port. If only one terminal is available, it can be switched from the main port to the diagnostic port in order to run the tests. Use the <SPACE BAR> to select the proper terminal type. Then select a test using the arrow keys or corresponding command letters. Once the terminal type and test have been selected, press <RETURN> to enable the down-load feature. Once enabled, the prompt at the bottom of the screen changes to:

To run another program, select one here, then reset the DTD or exit the current program using <BREAK> on the DTD diagnostic terminal.
<SPACE BAR> - Change terminal type <BREAK> - Exit

In order for the DTD computer bin to receive the diagnostic, it must be executing code in the loader firmware which enables it to receive SCSI information. The DTD computer should be ready to receive information from SCSI under any of the following circumstances: The DTD was just turned on, the RESET button was just pressed, or the last program executed on the DTD specifically returned the DTD to the proper point in the loader firmware when it finished.

As soon as the DTD begins receiving the diagnostic, the message "Starting up Multi-Track Computer..." will appear at the bottom of the screen. This message disappears when the transfer is completed. If only one terminal is available, it should be switched to the diagnostic terminal port as soon as the transfer begins. Hopefully the switch can be done before the program on the DTD begins to execute. If not, reset the DTD computer (return it to the loader), and the same diagnostic will be sent again, even though the terminal is connected to the DTD diagnostic port.

At this point, the selected diagnostic should begin execution. Prior to Release N final, all diagnostics executing on the DTD would 'hang' the DTD computer if the <BREAK> key was pressed. This is because the DTD loader firmware was being entered at a point which asked the computer to boot from a nonexistent floppy drive. With later releases, most programs have an insert in the source code which determines if the

program is executing on a DTD. This is determined by the existence of a PSBMC WITH THE D24 CONNECTION INSTALLED. If a program has this requirement satisfied, it will enter the loader at a point which will request information from SCSI. Note that for this function to work, any program added to the DTDMENU (inserted in .DTDXCAT) must have the DTDCHK insert prior to compilation.

AUDIOIO - This program is used to test basic functionality of the STM or Mono-sampling systems. There are also a few waveforms which can be generated from the poly system. The program takes a burst of samples from the selected A/D conversion system and displays either an FFT or time domain sweep on the screen. Because the results are displayed graphically, a terminal with graphics capability is required. When the program is executed, the following screen is displayed:

A. Sweep
B. Setup

A message "Creating test waves" appears on the bottom of the screen for several seconds. During this time, sine, random, and band-limited pulse waveforms are being computed and stored in poly memory.

The Sweep command causes the program to acquire samples and display results according to the setup defined in the Setup page. In the setup page, parameters are changed by positioning the cursor (arrow keys) on a parameter and pressing the Space Bar. A prompt for a new parameter value appears at the bottom of the screen. The FFT computation code in this program is equivalent to that in SFM.

DSP70TST - This program tests the basic DSP functions and the communications between the DSP70 and the Able computer. When the test is executed, a extensive menu of individual tests is displayed. To run each test once, select command #10 from the menu. To run all of the tests iteratively, select command #11 from the menu. If an error is encountered, the test stops.

EXTMEM - This program tests external computer memory (M128K & M512K Dynamic RAM). When the test executes it first displays the number of sectors (512 sectors per M128K & 2048 sectors per M512K). The test then gives options concerning how errors are reported and which test types will be performed. Pressing <RETURN> in response to all prompts will enable all test types and give full error reports. The external memory test in the diagnostic LOGGER is functionally equivalent to this test.

FLOPTEST - This program tests the ability of a floppy drive to store information. A faulty drive OR a faulty diskette can cause errors in this test. When the test first executes, it prompts for the drive to test (F0, F1, R0, R1. - Type 'F0' to test Floppy drive 0) and then asks for a FORMATTED diskette. The diagnostic diskette should be removed and replaced with a formatted diskette. All information on this diskette will be destroyed. FLOPTEST displays a warning to this effect, prompts for a formatted diskette, and waits for a <RETURN> key before beginning the test.

GRAPHTST - This program is used to test graphics printing and consists of three parts. The first prints 50 lines of the character set, the second part sends a pattern to the terminal and the printer, and the third sends a pattern to the terminal, reads the pattern back from the terminal graphics RAM, and then sends it to the printer. Any differences between what is printed on the screen and what is printed on paper indicates a problem with graphics printing capability. A system which fails only the third part probably has a bad terminal, while one which fails parts 2 & 3 usually has a problem with the printer.

KEYBOARD - This program tests the functions of the Velocity/Pressure keyboard. When the KEYBOARD test is selected from the diagnostic menu, the following command screen is displayed:

Keyboard Test - 01/01/88

==> A. Display test
B. Lights test
C. Keys/buttons test
D. Foot-switch test
E. Pedal 1 test
F. Pedal 2 test
G. Mod/Pitch wheels test
H. Ribbon controller test
I. Keyboard knob test
J. Breath controller test
K. Instructions

Press <BREAK> to exit

Select test using arrow keys and press <RETURN> to begin execution.

Instructions for the keyboard test are displayed when the command "K. Instructions" is executed.

LOGGER - The LOGGER diagnostic tests many system functions and maintains a log of any errors recorded. When LOGGER executes, the first screen displayed shows the configuration of the system. This screen has the following format. (The title line should say either Synclavier or Direct-To-Disk reflecting the system type on which it is executing.)

System configuration - Synclavier

Processor type : Model D

External memory : 1536K

D16 Timer :	Yes
SMPTE :	Yes
Poly System	Yes
D32X :	Yes
D100A :	Yes
D110A:	No
D40Q	Yes
M64K	Yes
D34 GPI	No
D24 SCSI IDs :	#0,1
MIDI Systems :	#0
D115D IDs :	#0
DSP 70 IDs :	None

Poly Bins:	Bin 1	Bin 2	Bin 3
Megabytes :	32	32	NOT
Poly Voices:	32	32	FOUND
PSAC #1:	Yes	Yes	
PSAC # 2:	Yes	Yes	
PSADC :	Yes	Yes	
DDDAC :	Yes	Yes	
STMBoxes:	1		
STM Inputs:	2		
PSAD Type:	Anal.		
PRM:	Yes		

Press <RETURN> to continue ...

When LOGGER leaves the configuration screen, The test select screen is displayed. Note that the system configuration screen can be accessed again from the select screen.

Error log diagnostic - 08/01/89

	Bin 1	Bin 2	Bin 3
Serial chain	Off	Off	Off
Poly memory addressing	Off	Off	Off
Poly Memory	Off	Off	Off
Phase accumulator #1	Off	Off	Off
Phase accumulator #2	Off	Off	Off
Sample rate generator	Off	Off	Off
A/D controller	Off	Off	Off
Amplitude computer #1	Off	Off	Off
Amplitude computer #2	Off	Off	Off
Wave, Env bus/ Osc. swap	Off	Off	Off

D16 timer load	Off
D16 vs. D3 timer	On
D4567 Multiply/Divide	On
External memory	On
M64K clocks/calendar	Off

Select desired tests and press <RETURN> to start testing.

<S> - Select all tests <D> - De-select all tests
<SPACE BAR> - Select/De-select <C> - System Configuration

Pressing the <S> key will select all tests that are valid to run on the system. (For example, if a system has no external memory, it would be invalid to run the external memory test.) If it is not valid to run a given test, that test will remain "Off". Pressing <D> will turn all tests off. Individual tests can be turned on and off by moving through the tests using the arrow keys and then selecting/de-selecting the test using the <SPACE BAR>.

When the desired tests are selected, press <RETURN> to start testing, and the following screen is displayed. If a poly bin is not present, its corresponding error column is not displayed.

Error log diagnostic - 08/01/89 Iteration # 1

Test	Bin #1 Errors	Bin #2 Errors	Test	Errors
Serial chain	------	------	D16 timer	------
Poly memory addressing	------	------	D16 timer / D03 clock	------
Poly Memory	------	------	D4567 Multiply/Divide	------
Phase accumulator #1	------	------	External memory	------
Phase accumulator #2	------	------	M64K Clocks	------
Sample rate generator	------	------		
A/D controller	------	------		
Amplitude computer #1	------	------		
Amplitude computer #2	------	------		
Wave bus	------	------		
Envelope bus	------	------		
Oscillator swap	------	------		

<S> - Select tests <BREAK> - Exit

While any individual test is executing, pressing <SPACE BAR> causes the remainder of that test to be skipped, and LOGGER will move to the next selected test.

MAINMEM - This is a main memory test which tests 60K or 63K of main computer memory. An M64K together with a D40Q will give an additional 3K of memory (63K). This test first fills memory with numbers, then rotates the numbers in small blocks until they are back in their original order. This rotation is repeated for all blocks of memory at a high rate until all blocks are rotated. Then the entire memory is checked against the original values written. This test has been known to fail memory cards that show real-time errors but pass MEMBURN.

MEMBURN - This program also tests main computer memory. When MEMBURN executes, it first prompts for the amount of memory in the system. At this point, all or part of memory can be selected for test. Once the amount of memory is selected, the test begins, and displays the following:

- The running time of the test
- The current status of each section of memory under test
- the number of test iterations completed.

If an error is detected the word "FAULTY" appears under the corresponding section of memory.

METERTST - This diagnostic tests the functions of the DTD meter bridge display, and therefore is only on the DTD diagnostic menu. The program sends display patterns to the meter bridge while showing the proper response on the terminal. Any difference between the two indicates a problem. When the test first executes, it prompts 'Automatic mode (Y/N): '. If automatic mode is not chosen, the patterns must be single-stepped using <SPACE BAR>. The test then asks for the start and end channel to test. After these prompts, the test begins by briefly lighting all LEDs and then sequencing through the patterns.

MIDITEST - When the MIDI test is selected from the diagnostic disk the following screen is displayed:

MIDI DATA LOOP TEST - 08/01/89

Existing subsystems: 1

Subsystem #1

No data loop established

Connect the MIDI cable from each output channel(1-4) to the INPUT channel and verify that a data loop is established.

Hit the space bar to go to the next MIDI subsystem.

The 'Existing subsystems:' line lists the number of subsystems found. For example, a system with a 2 X 8 MIDI unit should show 1,2

The program tests 1 subsystem at a time by dumping a test pattern through all of the 4 outputs. This pattern is always being sent, but a data loop is not established until the MIDI cable is connected from an output to the input.

When a loop is established, the message 'No data loop established' (in reverse video) will disappear from the left side of the screen and 'Data loop established' (also in reverse video) will appear on the right side of the screen. If the MIDI system receives a different byte pattern than it sent, a message to that effect is displayed.

Pressing the space bar will move the test to the next sub-system. If there is only a 1 X 4 then the test will show the same screen again. When the current subsystem is one on the right of the MIDI module, The prompt reads 'Connect the MIDI cable from each output channel (5-8) to the AUX channel and verify that a data loop is established.'

MITTEST - This program tests basic functionality of the PRODIGI digital transfer unit (There are only 15 of these in existence). The PRODIGI is controlled by a D24 in the DTD computer bin, so this program is available only from the DTDMENU. When the test is executed, a extensive menu of individual tests is displayed. To run each test once, select command #10 from the menu. To run all of the tests iteratively, select command #11 from the menu. If an error is encountered, the test stops.

MULTICHN - This test program first checks all voices in the system to verify that each is present at exactly 1 MT input. This "routing" procedure constructs a numerical map describing which multi-channel input corresponds to each voice. If FM voices are present, they are routed first, followed by any poly voices. The voice routing screen is shown below:

MT1 Inputs			MT1 inputs		
Chan	Left	Right	Chan	Left	Right
0	nnn	nnn	1	nnn	nnn
2	nnn	nnn	3	nnn	nnn
.	.	.	.	.	.
30	nnn	nnn	31	nnn	nnn

FM (or PS) voices route properly
Press <RETURN> to continue or any key to map FM voices again...

The 'nnn' numbers are the MT input numbers which can range from 0 to 255. If an MT input number for a channel is missing, then the voice was not present at an MT input. If 'XXX' appears in place of an MT input number, then that voice routed to more than one MT input. In the case of mono poly voices, the two 'Left' columns will have no MT input numbers.

NOTE: with some of the newer systems, the routing process (in this diagnostic or in real-time) may result in shorts the first time after the machine has been powered up. Re-routing usually executes successfully. The real-time software stores the routing configuration to a system file (.SINF-7) and does not re-route every time the Synclavier is started. It will re-route only if the number of voices has been changed. If .SINF-7 can't be located in the system file, then the routing is performed every time. It is important to realize that if an incorrect routing is stored in this file, real time errors can occur even if there is no hardware problem. When trouble-shooting multi-channel, it is usually a good idea to rename the .SINF file until the problem is solved.

If an error occurs during the mapping process then the prompt 'routing error occurred, press <RETURN> for more information...' appears at the bottom of the screen. An

example system with both types of routing errors might show this screen:

The following voices did not route:
0L,3R

The following voices routed to more than one MT input:
4L,6L,18L,23R

The following MT inputs were involved:
MT1 card 15: 5,7 MT1 card 11: 0,3

The MT1 cards are numbered from right to left in the bin, starting from their corresponding MT2. In a system with more than 8 multi-channel outputs, and card 0 is indicated as a possible problem, the right-most MT1 card in the top bin or any of the MT1 cards chained to it might be the source. (The numbers following the MT1 card are the corresponding inputs on that card, numbered from 0-7.)

If the routing process executes with no errors, then the following screen is displayed:

Multi-channel Test - 01/01/88

System configuration:
32 stereo FM voices present
32 stereo PS voices present
16 Multi-channel outputs

A. Display FM voice routing map
B. Display PS voice routing map
C. Display Multi-channel output table

Select a command or press <Break> to exit.

Commands A & B perform the routing procedure again with FM and Poly Voices respectively. If the routing was originally completed with no errors, then this is not necessary. The last step (Command C) tests the output section of the multi-channel system. It verifies that every input is capable of switching to each output, and that when connected to a given output, it is not present at any other output (shorted). It should be noted that the multi-channel output test uses information obtained during the voice routings, so the system will not pass the output test if any voices don't route properly.

The C command pages through all voices (FM & PS) 16 at a time. A page of 16 FM voices on a system with 16 multi-channel outputs (and some output problems) might look like this:

("-" = O.K., "X" = shorted, "space" = missing)

```
voice    MT                    1 1 1 1 1 1 1 1 1 1 1 2 2 2 2 2 2 2 ...
card chn card 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 ...

FM 0 0L: 15,1 - - - - - - - - - X X - - - - -
FM 1 0R: 14,1 - - - - - - - - - X X - - - - -
FM 0 1L: 15,2
FM 1 1R: 14,2 - - - - - - - - - X X - - - - -
FM 0 2L: 15,3 - - - - - - - - - X X - - - - -
FM 1 2R: 14,3 - - - - - - - - - X X - - - - -
FM 0 3L: 15,4 - - - - - - - - - X X - - - - -
FM 1 3R: 14,4 - - - - - - - - - X X - - - - -
FM 0 4L: 15,5 - - - - - - - - - X X - - - - -
FM 1 4R: 14,5 - - - - - - - - - X X - - - - -
FM 0 5L: 15,6 - - - - - - - - - X X - - - - -
FM 1 5R: 14,6 - - - - - - - - - X X - - - - -
FM 0 6L: 15,7 - - - - - - - - - X X - - - - -
FM 1 6R: 14,7 - - - - - - - - - X X - - - - -
FM 0 7L: 15,8 - - - - - - - - - X X - - - - -
FM 1 7R: 14,8 - - - - - - - - - X X - - - - -

 <Space bar> - pause  <Return> - return to command screen
```

This output table seems to suggest two separate problems. First, FM voice 1L would not route to any of the multi-channel outputs. Second, every voice that routed to output 10 was also found at output 11 and vice-versa. NOTE: due to the complexity of the multi-channel system, the test results may be misleading. For example, on a particular faulty MT1, all of the voices routed properly, and on the output test, only the voices on one MT1 card distributed properly. As it turned out, the only card that was shown to distribute properly was the faulty card.

PMCUTEST - This diagnostic first looks for a D34GPI and if found, continuously polls the PMCU buttons (RECORD, PLAY, STOP, REW, & FF) and reports button presses to the screen. The program also turns on the light under the button which was pressed, with the exception of the stop button which has no light.

PRINTEST - This test repeatedly sends the word TEST to the printer. It should be run long enough to verify that the XON/XOFF protocol functions properly when the printer buffer is full. (A full page of 'TEST's should be sufficient.) A failure would be indicated by missing letters or uneven spacing. A printer can be simulated using a terminal, and XOFF/XON can be sent by typing Control-S/Control-Q.

PSXMBTE - This program tests poly memory using several methods. The Poly memory test in LOGGER is functionally equivalent to this test. If more than one poly bin is found, PSXMBT will prompt for a bin selection. It then prompts for the following information about the configuration of the poly memory system:

- The number of memory controllers
- The number of 16 Mbyte cards next to each controller
- The number of 4 Mbyte cards next to each controller
- The number of 1 Mbyte cards next to each controller

Once this information is entered, PSXMBTE prompts for the section of memory to test by asking for the starting board and bank and the ending board and bank. Boards are numbered from 0 and there are eight banks per board numbered 0 - 7. To test only the first memory board in the system the correct response would be: 0,0,0,7. To test all boards in a system with n memory boards,the correct response would be 0,0,(n-1),7.

The test then gives the option to select error reporting modes and also to disable the different test types. Pressing <RETURN>for each of these prompts will give full error reporting and enable all test types. The test also prompts for the number of seconds to wait for the refresh test. (30 seconds is usually sufficient.) There is also an override function which is used only for PCB level repair.

This test provides three 'interference' modes during the memory test procedures. This interference is in the form of other transactions to and from the poly bin during the memory test. The three types are:

1) No interference
2) Tracking interference
3) Random interference

The interference type changes every iteration, so at least three iterations are required to use every interference mode.

SMPTETST- This test waits for drop frame SMPTE code and displays the time corresponding to the code received. The values are displayed on the screen in a scrolling fashion.

STARTUP - This diagnostic verifies the storage capability of the SCSI devices used with the Direct-To-Disk, and therefore appears only on the DTD diagnostic menu. The program begins by polling and identifying all SCSI devices. The type and target ID number listed should be checked against the physical configuration of the DTD.

STARTUP then displays the following menu:

0. Change Test Parameters

1. Start Disk Drives
2. Load Tape Drives

3. Disk Read Timing Test
4. Disk Write Timing Test

5. Disk Verification
6. Tape Verification

0. Change Test Parameters - Entering zero (or pressing <RETURN>) enters the test parameters menu. The default parameters 3,4, & 5 depend on the type of tape drives found (parameter 6). The following list shows the parameter defaults for the different tape types:

	EPI	Fujitsu	CDC Patriot
1. Test Pattern	Square wave	Square wave	Square wave
2. Test iterations	1	1	1
3. Tape Test size	500	500	1000
4. Tape Disconnect	enabled	enabled	disabled
5. Tape block mode	variable	variable	fixed
6. Tape type	EPI	FUJITSU	CDC
7. Tape write	enabled	enabled	enabled

The only parameters which need to be modified under normal circumstances are parameters 1,2, & 3. To change a parameter, enter the corresponding number, and the choices are listed.

Parameter #1, Test pattern, allows three choices: Ramp wave, which writes 0,1,...FFFF...; Square Wave which writes 0000, FFFF,..., and 0,-3,... The latter was used to drive specific buffer memory problems, but the ramp and square waves should both be used for normal system test.

Parameter #2, test iterations sets the number of times a test is executed, and #3 sets the tape test size (in blocks). The maximum number of blocks for EPI and Fujitsu is 12800, and for the CDC Patriot, the maximum number of blocks is 25600.

Parameters 4 & 5 depend on the tape type found. If STARTUP finds the wrong tape type, then something else is wrong (IE targeting). These parameters should be as shown in the table above.

Parameter #7 can disable tape write in order to perform constant tape read tests. If tape write is disabled, then the tapes in the drives must have been previously filled with the proper test pattern.

1. Start Disk Drives - This command starts all disk drives found by the buffer memory system. If the Wait Spin jumper is installed on a disk drive, then that drive waits for a start command before spinning its disk up to speed. Production units usually have these jumpers installed, so this command must be executed before any commands which read or write to the disk. When this command is executed, the program starts a timer and waits for the disk drives to acknowledge that they have started. When an acknowledgment is received, the message:

"Disk drive at D24 #3, Port ___, Target __, LUN 0 is ready" is displayed.

2. Load Tape Drives - This command loads the tape into the tape drive and prepares the drive for read/write access. This command must be executed before running tape verification. When this command is executed, the program starts a timer and waits for the tape drives to acknowledge that they have loaded.

3. Disk Read Timing Test - This command reads data from the disks and reports back the average read rate in Kbytes per second.

4. Disk Write Timing Test - WARNING: This command writes data to the disk and will destroy any previously stored information. It reports the average write rate of each drive in Kbytes per second.

5. Disk Verification - WARNING: This command writes data to the disk and will destroy any previously stored information. The selected test pattern is written to the disk drives, read back, and verified. Errors are displayed to the screen.

6. Tape Verification - WARNING: This command writes data to the tapes and will destroy any previously stored information. The selected test pattern is written to the tape drives, read back, and verified. Errors are displayed to the screen.

STMADJ - This program is used to adjust the DC offsets of the STM inputs. The adjustments are made using trim pots on the daughter boards, and the program specifies which pots to adjust while it is executing. Original STMs with NED daughters also require a generator to to a gain adjustment. The program determines the STM type and automatically uses the correct adjustment procedure.

THRUPUT - This program is used to test the basic functionality of the audio paths of the polyphonic sampling system of a Synclavier or DTD. Sine waves can be computed mathematically and generated from the poly system, or an STM input can be armed so that its audio input is stored in poly memory and then passed through to the poly voice D/A converters.

In the current Poly system, there are 33 oscillator channels (0-32). In pre-DTD systems, channel 32 was known as the "ghost" or "refresh" channel, and channels 0-31 were used to drive the voice card D/A channels. In a DTD, channel 32 is also used to synchronize the other channels together. STM inputs also use these oscillator channels, but in real-time software, an STM channel does not share an oscillator channel with a voice card.