DBM-1
manual

pragmatic designs INC.
950 Benicia Ave.
Sunnyvale, CA 94086
Phone: (408) 736-8670

Disclaimer:

Pragmatic Designs, Inc. makes no representations or warranties with respect to the contents hereof and specifically disclaims any implied warranties or merchantability or fitness for any particular purpose. Further, Pragmatic Designs, Inc. reserves the right to revise this publication and to make changes from time to time in the content hereof without obligation of Pragmatic Designs, Inc. to notify any person of such revision or changes.

DBM-1 MANUAL

TABLE OF CONTENTS

DBM-1 MANUAL

TABLE OF CONTENTS

ILLUSTRATIONS

1.0 INTRODUCTION

The DBM-1 "Debug Memory" is a random-access memory card which simulates read only memories to assist users in developing microcomputer programs for use in other systems.

The Debug Memory simulates read only memory devices to the user's target microcomputer system while remaining under the control of the S-100 development system. The card may be configured to simulate many types of EPROM, PROM, and ROM devices including:

27Ø8, 27Ø8L, 27Ø8-1	1Ø24 x 8 UV Erasable PROM
2758	1Ø24 x 8 UV Erasable PROM, single +5V supply
23Ø8, TMS 47Ø0	1Ø24 x 8 MOS ROM
26Ø8	1Ø24 x 8 Factory Programmable PROM
TMS 2716	2Ø48 x 8 UV Erasable PROM
2716, 2716-1, 2716-2	2Ø48 x 8 UV Erasable PROM, single +5V supply
TMS 2516	2Ø48 x 8 UV Erasable PROM, single +5V supply
2316E	2Ø48 x 8 MOS ROM
2616	2Ø48 x 8 Factory Programmable PROM

Under software control, data from the S-1ØØ system is written into the Debug Memory. The user's target microcomputer system can then read data out of the Debug Memory by inputting its own memory addresses and chip selects, just as if the Debug Memory were a ROM chip in the user system. At any point, the entire contents of the Debug Memory can be displayed and/or modified by the operator through the S-100 development system.

Each Debug Memory can simulate a ~~2048~~ 1ØØ byte block of memory. Therefore, two ~~1K~~ 1ØØ type, or one 2K type ROM may be simulated by one DBM-1 card. Additional Debug Memory cards may be added to simulate additional memory.

The Debug Memory connects to the user's target microcomputer system through one or two 24 pin DIP jumper cables. Depending on the capacitive loading and timing margins in the user's system, cable lengths of up to three feet or longer may be used reliably, even with unbuffered system

busses. The DBM-1 includes a passive terminator to reduce crosstalk and ringing on the interface signals. Generally, however, the cable length should be kept as short as possible.

The Debug Memory contains a hardware address trap to assist in program development. The target trap address is loaded by writing it into the highest two memory locations of the Debug Memory. When the target trap address is accessed by the user's system, a flip-flop is set, lighting an LED on the board. A 16 pin connector makes several trap signals available for interfacing to the user's system. These signals may be used to drive an external indicator LED, control the target processor's memory ready logic, or to reset the target processor. When two DBM-1 cards are used to simulate a ~~4K~~ 4000 block of ROM, the trap logic may be daisy chained, so that the trap address for the 4K block may be written into the highest two locations of the 4K block, rather than at the top of each 2K block.

When Debug Memory is not being used for development of the user's target microcomputer system, it may be used as normal RAM in the S-100 development system. By the same token, when Debug Memory is simulating a ROM in the user's target microcomputer, the S-100 system may be used for other purposes, such as assembling, listing or executing other applications programs. The ROM simulation will not be disturbed unless the S-100 system accesses Debug Memory.

2.0 FUNCTIONAL DESCRIPTION

The DBM-1 block diagram is shown in Figure 1. The five major blocks of the card are:

Card Select

Memory

Memory Address Multiplexer

Control

Address Trap

Each of these blocks is discussed in the following sections.

2.1 Card Select Block

The DBM-1 is mapped into the S-100 development system as ~~a 2K~~ an 100 block of memory, whose base address is selectable by jumpers on the card. The high order S-100 address lines are compared to the selected address by the Card Select Block, and when they match, the card is selected for access by the S-100 bus.

2.2 Memory Block

The Memory Block consists of ~~16K~~ bits of static RAM, organized as ~~2048~~ 100, 8 bit words. The data out is enabled onto the user's target computer system by its chip enable/read control lines. The Control Block enables bidirectional data flow to the S-~~100~~ bus.

2.3 Memory Address Multiplexer Block

The Address Multiplexer Block selects the user target microcomputer address lines or the S-100 address lines for driving the memory address inputs. Normally, the user target microcomputer addresses the memory. Only when an access is requested by the S-100 bus do the multiplexers switch. When an S-100 bus access occurs, signals on the 16 pin interface

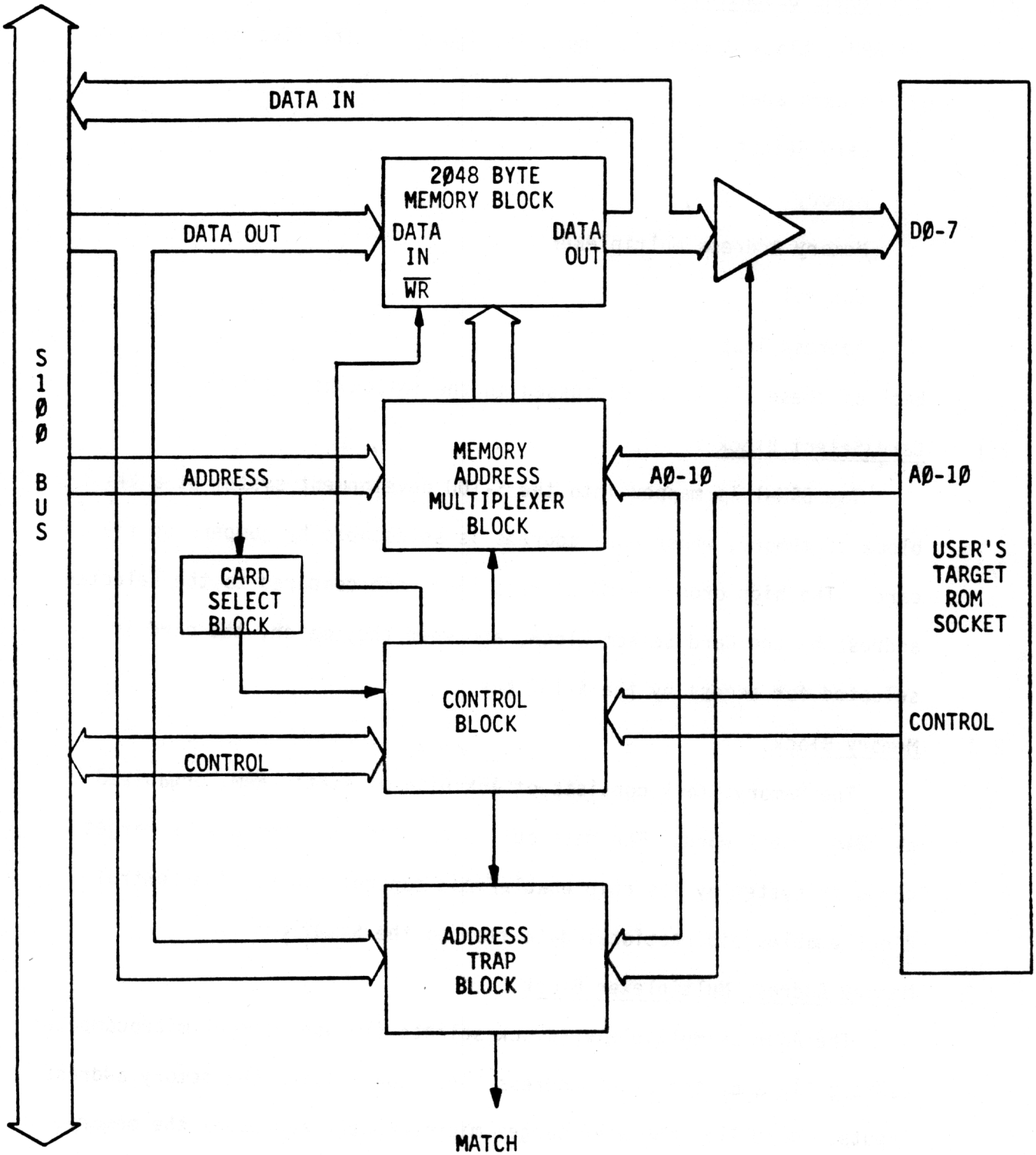


Figure 1

BLOCK DIAGRAM

cable and on connector P-2 indicate that data returned to the target system is invalid and that the target system is not addressing the memory at that time. This signal may be used to reset the target processor, if desired.

2.4 Control Block

The Control Block controls the flow of data from the S-1ØØ bus to the Memory Block. It provides the select signal to the Address Multiplexer Block, generates the memory read/write signal, and requests wait states of the S-1ØØ bus.

2.5 Address Trap Block

The Address Trap Block decodes the two high memory locations of the selected 2K memory block. Data written to these locations is latched so that it may be compared to the addresses input by the user's target microcomputer system. When a match occurs, a TRAP flip-flop is set, the trap LED is lit, and signals on the 16 pin interface cable and on the connector P-2 indicate that the trap address was reached. The TRAP flip-flop is cleared by the next S-1ØØ access to the Debug Memory.

If two DBM-1 cards are used, the address trap logic may be daisy chained, so that the trap address is written into the high two locations of the 4K byte memory block.

The trap address match is sampled on either the leading or trailing edge (switch selectable) of the user system's chip select. If the ROM being simulated is part of an array on a memory board, an additional qualifier signal input (i.e. BOARD SEL/) may be provided to qualify the actual system address match for the trap.

3.0 MEMORY MAPPING

The ROM being simulated by Debug Memory may be located anywhere within the user's target microcomputer system. Similarly, the DBM-1 itself may be located on any ~~2K~~ 100 byte boundary in the S-100 development system memory map. The user must, therefore, mentally "map" the target microcomputer address to the S-100 system address, simply by adding an offset to each memory access address.

For example, if the ROM being simulated is at the user's target microcomputer location 0-7FFH, and the card is jumpered for address block 8000-87FFH in the S-100 system memory map, a constant of 8000H is added to the desired target address to determine the S-100 address. Target location 0 becomes S-100 location 8000H, etc. The trap address is written into locations 87FEH (low byte) and 87FFH (high byte).

3.1 Memory Interface

The DBM-1 is interfaced to the target computer by one or two 24 pin DIP jumper cables. The length of these cables should be kept as short as possible. Typical ribbon cable has a capacitance of approximately 22pf per foot. The DBM-1 cable option supplies three foot cables of this type, which should be adequate for most applications.

If system capacitive loading and timing margins are small, or if use of a longer cable is required, refer to Appendix E for the schematic of a bi-polar memory buffer board. This board can be placed close to the target system ROM sockets to eliminate the capacitive loading of the ribbon cable.

4.0 JUMPER OPTIONS/SOCKET SELECT

DBM-1 requires users to select memory address blocks, ROM type, polarity of certain trap signals, and several trap options. These options are selected by installing jumper wires or by setting DIP switches. The following sections discuss these options.

4.1 Address Select

The DBM-1 location in the S-100 development computer's memory map is selectable by switch SW1, as defined in figure 2. The card may be placed on any ~~2K~~ 100 boundary.

The jumpers and the address bits they represent are as follows:

Jumper	Address Bit
A	A10
B	A12
C	A13
D	A14
E	A15

For example, to locate the card from location 8000H-8FFFH in the S-100 computer memory map, jumper as shown in figure 3.

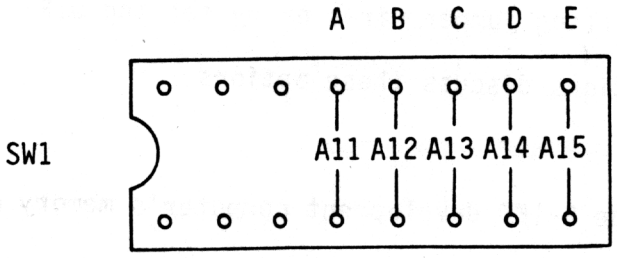
Note that the absence of a jumper indicates a logic 1. An installed jumper represents a logic 0.

4.2 ROM Type

The ROM type to be simulated is determined by which sockets on the card are used for interfacing to the user's target microcomputer ROM sockets, and by a switch selection*. Three different pinout interface sockets are provided, as defined in figure 4. (Only one type of ROM may be simulated at any one time.

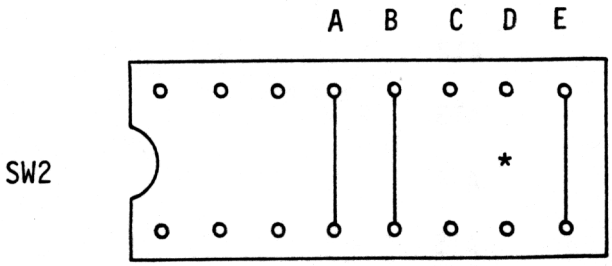
* See Appendix D for a description of software control select of ROM type.

on the connector P-2 indicate that the trap address was received



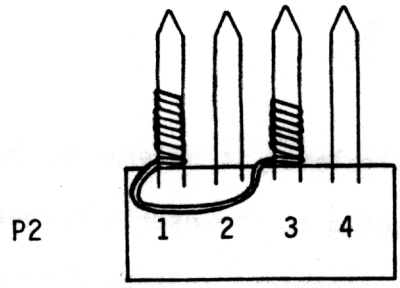
Jumper → Address Bit = Ø
No Jumper → Address Bit = 1

Switch 1



* Open = 2K ROM
Closed = 1K ROMs (highest location in memory block should have Bit 7 = Ø)

Switch 2
(Normal Settings. See text for options.)



Jumper for normal trap operation (not daisy-chained)

Figure 2
Jumper/Switch Select Summary

A11 = Ø, jumper
A12 = Ø, jumper
A13 = Ø, jumper
A14 = Ø, jumper
A15 = 1, no jumper

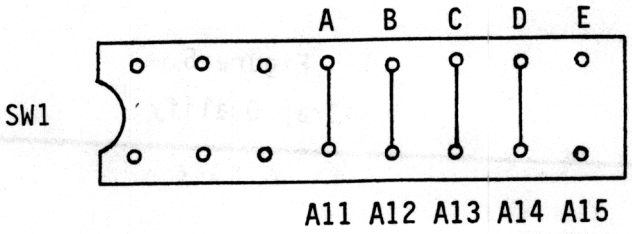


Figure 3
Jumpering Example (8ØØØH)

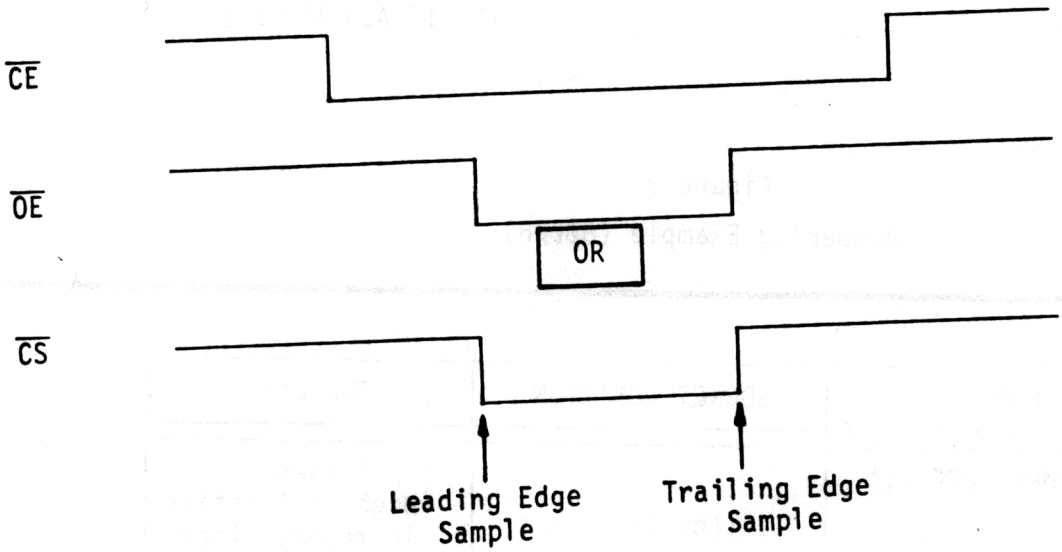
ROM	SOCKET POSITION	SW2-D**
27Ø8, 27Ø8L, 27Ø8-1* 2758 23Ø8, TMS 47ØØ* 26Ø8	1 (Low 1K) 2 (High 1K)***	Closed (Highest location in memory block should have Bit 7 = Ø)
I2716, 2716-1, 2716-2 TMS 2516 2316E 2616	3	Open
TMS 2716	4	Open

* Pin 18 should be grounded or active low in target microcomputer
** See Appendix D for a description of software control of select
*** Both 1K ROMs must have common AØ-A9 and DØ-D7 signals

Figure 4
DIP Socket Options

No External Qualify Signal	SW2-E Closed
Applied External Qualify Signal	SW2-E Open

Figure 5
Trap Qualify



Leading Edge Sample	SW2-B Closed	SW2-C Open
Trailing Edge Sample	SW2-B Open	SW2-C Closed

Figure 6
Trap Address Sample

Two S-100 Wait States	SW-2A Closed
One S-100 Wait State	SW2-A Open

Figure 7
Wait State Select

4.3 Trap Qualify

The ROM being simulated may be part of an array on a memory board within the user's system. Individual ROMs may be "selected" to drive the board's internal data bus, but the board itself may not be selected to drive the microcomputer bus. Since such selects would "fool" the trap logic into thinking a memory location had been accessed by the microcomputer, a BOARD SELECT/ qualifier input is provided for the trap. If not used, SW2-E should be CLOSED. (See figure 5.)

NOTE: Don't apply an external signal to the Qualify input with SW2-E closed. Open the switch before applying the signal.

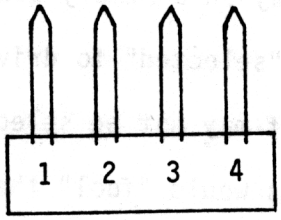
4.4 Trap Address Sample

The trap address match is sampled on either the leading or trailing edge of the ROM's chip select control signal. Normally, the user's microcomputer address lines are valid at the leading edge of the enable so SW2-C should be OPEN, and SW2-B should be CLOSED. (See figure 6.)

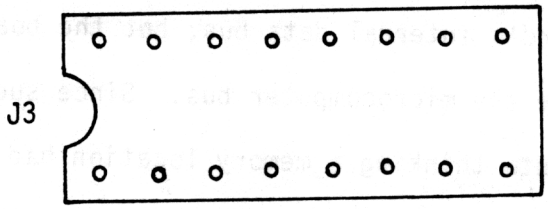
NOTE: If changing switch settings for B and C with the card powered up, don't close both SW2-B and SW2-C simultaneously. Open one switch before closing the other.

4.5 S-100 Wait States

Wait states are required for S-1ØØ access to the DBM-1 (don't forget that the user's target microcomputer accesses at full speed). Normally, two states should be generated. In some S-1ØØ microcomputers, one wait state may be sufficient. SW2-A selects one or two wait states for S-1ØØ accesses. (See figure 7.)



P2



PIN	FUNCTION
P2-1	$\overline{\text{AUX OUT}}$ (daisy-chain)
P2-2	$\overline{\text{QUALIFY}}$
P2-3	$\overline{\text{AUX IN}}$ (daisy-chain)
P2-4	GROUND
J3-1	N.O.
J3-2	$\overline{\text{QUALIFY}}$
J3-3	TARGET RESET
J3-4	$\overline{\text{TARGET RESET}}$ (Open Collector)
J3-5	$\overline{\text{MATCH}}$ 1
J3-6	MATCH PULSE
J3-7	$\overline{\text{MATCH}}$ 2
J3-8	GROUND

Figure 8
User Interface Signals

5.0 USER INTERFACE SIGNALS

Several signals relating to the user's target microcomputer are available to the user on a 16 pin connector P2. These include daisy chain, qualify input, "match", and reset signals. The pinout of the interface connectors is shown in figure 8, and each of these signals is discussed in the following sections.

5.1 Daisy-Chain

The daisy-chain logic allows the trap address to be written at the top of the 4K block of memory when two DBM-1's are used. Using box type contact jumper wire, connect the two boards as shown in figure 9.

5.2 Qualify Signal

The qualify signal may be obtained from the user system's board enable logic. If two DBM-1 cards are used, the qualify signal is applied to both cards.

5.3 Match Signals

Buffered outputs from the Address Trap logic are provided to the user. These signals and their polarities and timing are defined in figure 10.

5.4 Reset Signals

When the S-100 development computer accesses the DBM-1 memory block, output signals are provided which indicate to the user's target microcomputer that the data returned by the Debug Memory is invalid. The signals are stretched to approximately 100ms by a one-shot. One signal is active high, the other is active low, open-collector, with a current limiting resistor. It may be connected directly to the user's target microcomputer power-on-reset node if desired.

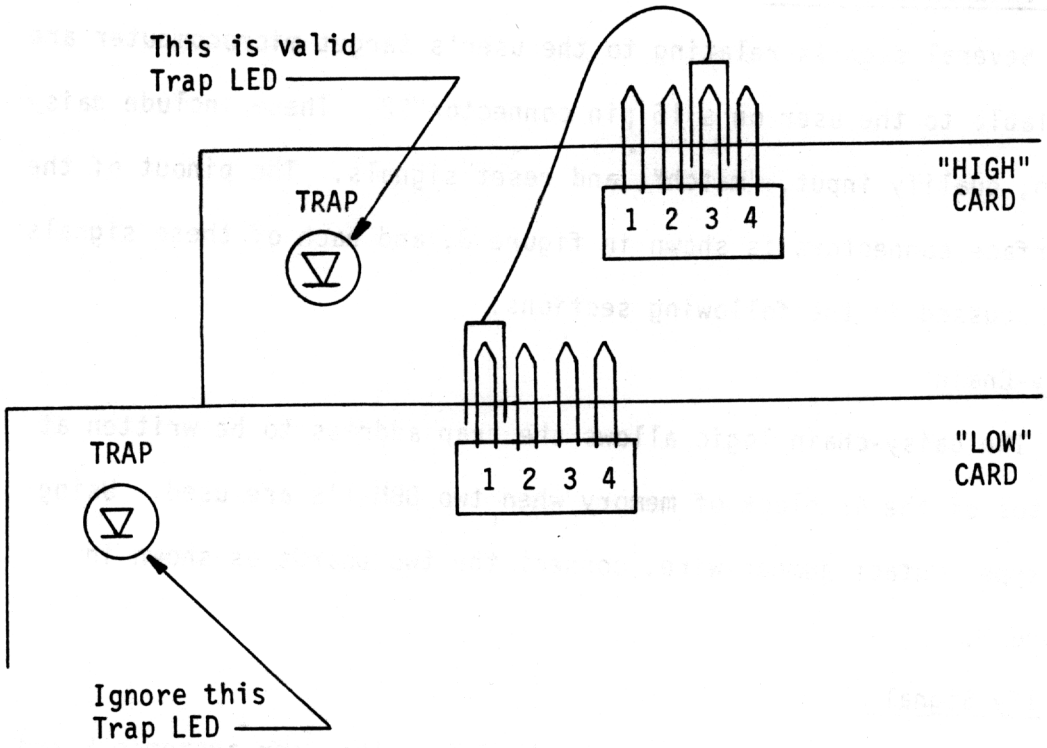
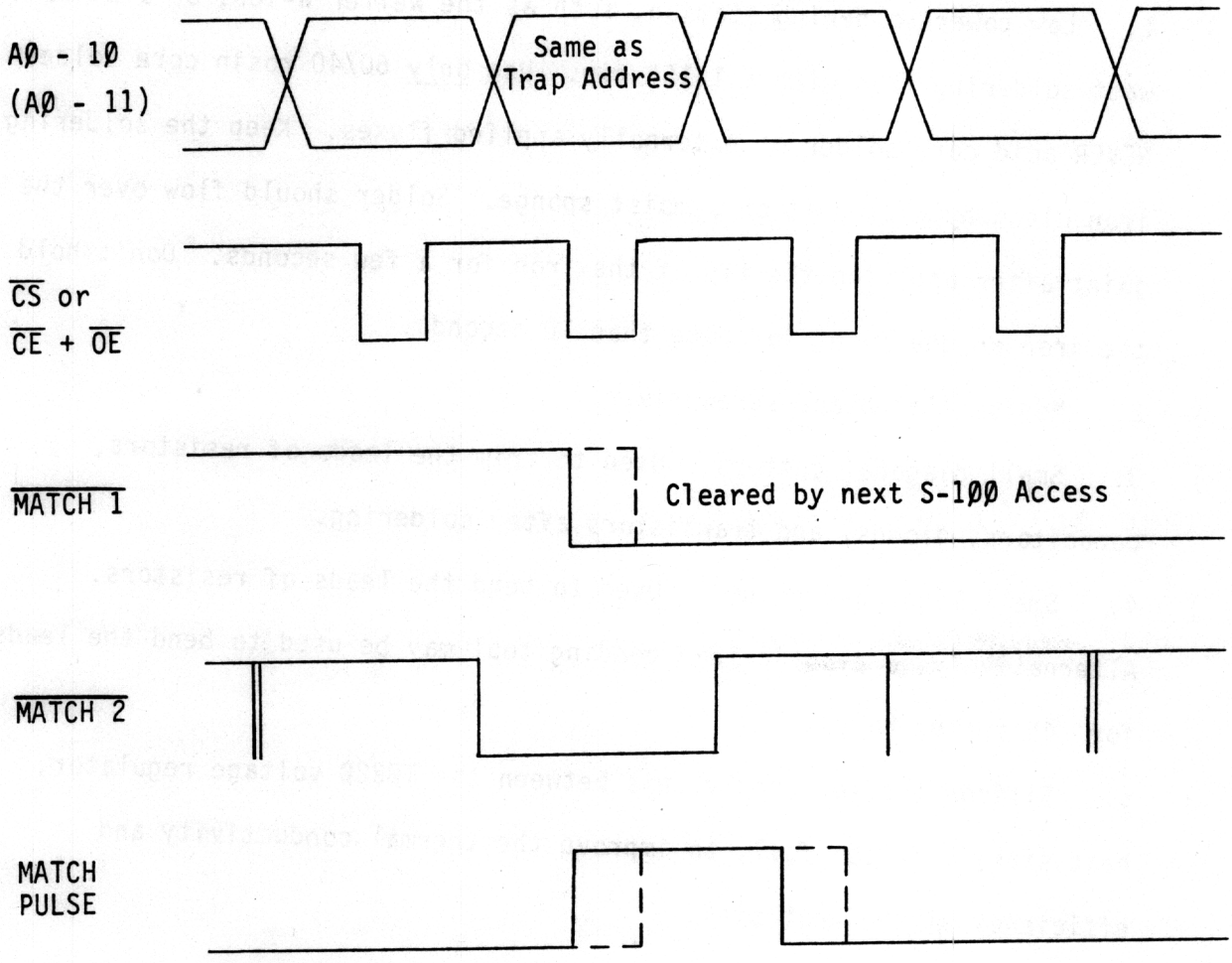


Figure 9

Trap Daisy-Chain



Signals available from "HIGH" card if two DBM-1s are daisy-chained, except MATCH2 indicates MATCH of AØ - 1Ø only.

Figure 10
Match Signals

6.0 DBM-1 ASSEMBLY PROCEDURE (KITS ONLY)

Users with DBM-1 kits should follow this assembly procedure. The following tools are suggested for this assembly procedure.

( ) 1. Low power soldering station, such as the Weller W-TCP, or a 25 watt watt soldering iron with a 1/16" tip. Use only 60/40 rosin core solder NEVER acid core solder or externally applied fluxes. Keep the soldering iron clean by wiping it on a moist sponge. Solder should flow over the joint after applying the tip of the iron for a few seconds. Don't hold the iron on the joint for more than 10 seconds.

( ) 2. Medium flat blade screwdriver.

( ) 3. Small diagonal cutters. Used to trim the leads of resistors, capacitors, diodes, and transistors after soldering.

( ) 4. Small long nose pliers. Used to bend the leads of resistors. Alternatively, a plastic lead bending tool may be used to bend the leads for .4" spaced holes.

( ) 5. Silicon grease. Apply this between the TØ22Ø voltage regulator, heat sink, and P.C. foil, to improve the thermal conductivity and efficiency of the heat sink.

6.1 Assembly Instructions

Refer to the assembly drawing (Appendix B) and parts list (Appendix C) while following the assembly procedure.

( ) First, carefully check the parts supplied against the parts list using the open spaces on the right edge of the parts list. Also, carefully inspect the PC board for shorts; this is a very important step.

( ) 1. Mount and solder diodes D1, and D3-20. Be sure that the banded end of the diode matches the band indicated on the PC board silk screen.

( ) 2. Mount and solder resistors R1-7. Mount and solder resistor networks RN1 and RN2. Be sure that the dot or diagonal corner on the resistor network matches the diagonal corner on the PC board silk screen.

( ) 3. Mount and solder IC sockets at locations 1, 2, 3, 4, 6, 7, 11, 14, 15, 16, 21, 22, 23, 25, 29, 30, 36 and POSITION J3. If the IC socket option was ordered, mount and solder sockets at the remaining IC locations. Insert the socket so that the Pin 1 indication matches that of the PC board. Solder two corner pins of each IC socket first - then make sure the socket is fully inserted and that all the socket pins enter their PC holes before soldering the remaining pins.

( ) 4. Mount and solder the 100µf electrolytic capacitor at location C4. Be sure the + lead of the capacitor matches the + indication on the PC board silk screen.

( ) 5. Mount and solder all remaining capacitors. Use .033µf disc capacitors at the non-polarized locations, and 2.2µf tantalum capacitors at the polarized locations. Be sure that the + end of tantalum capacitors match the + indication on the PC board. One of two styles of 2.2µf tantalum capacitors will be supplied. Use the appropriate holes

on the PC board for the capacitor style provided.

( ) 6. Coat the top and bottom of the heat sink with a thin layer of silicon grease. Also coat the PC board heat sink areas, and the bottom of Reg-1. Mount Reg-1 and its heat sink using the supplied 4-40 screw and nut.

( ) 7. Install the 5 position DIP switch at location SW2. Insert it into the <u>right most holes</u> of the DIP PC pattern.

( ) 8. Install the right angle post connector at location P2 as shown. The <u>short</u> sections of the pins are inserted in the board.

( ) 9. Mount and solder the red LED at location D2. The flat edge of the LED is the cathode, and should correspond to the bar of the diode symbol on the PC board silk screen, as shown.

( ) 10. Carefully inspect the board for solder bridges, unsoldered connectors, and cold solder joints.

( ) 11. DBM-1 is now ready to power up. Insert the board in the S-100 bus and measure the supply voltage <u>before</u> installing any ICs. There should be a power supply voltage of +5V at the IC sockets. If the power supply indicates normal voltage and there are no signs of component overheating, turn off power and add the ICs to the board per the assembly drawing in Appendix B.

If the power supply does not indicate correct voltage, find and repair this problem before continuing with IC insertion. Once again, when soldering ICs into the PC board be sure that IC Pin 1 matches

the PC board Pin 1. Solder the two corner pins of each IC first, then make sure that all the IC leads have entered their PC board holes before soldering the remaining IC pins. Care should also be exercised when plugging ICs into those positions previously socketed to insure that all leads penetrate the socket and make contact.

(✓) 12. Turn on the power and once again measure the power supply voltage. If the voltage is correct, the assembly procedure is complete. Install address select jumpers at SW1 as detailed in Section 4.1. Set the ROM select and trap options in Switch SW-2 as described in Section 4.2 - 4.5.

7.0 DEBUG MONITOR

Debug monitor is a small system monitor program designed to make it easier to use the Pragmatic Designs DBM-1 development module for applications program development. The monitor maps the application memory space into the DBM-1 memory addresses, controls the setting of the DBM-1 trap addresses, allows users to examine, display and fill either system or DBM-1 memory locations, and performs several other useful utility operations.

Debug monitor is provided in source listing form. This allows users to make any required changes and to locate the program inside their own system. The source listing is for a version of the program designed to run with the CP/M operating system. Users with this system may use the monitor as provided. Section 7.13 describes what portions of the program must be modified to place the program in PROM or to use it with a different operating system.

7.1 Debug Monitor Commands

Debug monitor has ten basic commands. The commands are input as a single character designator followed by the data required by the command. The commands are as follows:

A	Hexadecimal Arithmetic
C	Set/Reset DBM-1 Chain flag
D	Dump memory contents
F	Fill memory with constant
L	Clear Trap LED
M	Move memory block
O	Set DBM-1 memory offset

S Examine/substitute memory contents

T Set DBM-1 Trap address

V Test R/W memory

Each of these commands is explained in the following sections.

7.2 Mapped Commands

The commands D (dump), F (fill), M (move), and S (substitute) are mapped commands. This means that they can operate on either the system memory (normal operation) or the target processor/DBM memory (mapped operation). In normal operation, the commands use the address inputs as they are given by the user. In the mapped mode the command adds the current offset bias (as set by the O command) to the input address. Mapped operation is indicated by preceeding the input with a period (.).

The mapped commands are useful to allow the use of the original application source program addresses when the DBM-1 is actually located at a different place in the S-100 system memory map.

For example, suppose the DBM-1 is located in address block 8000H in the system memory. When using DBM-1 with a target program ORGed at 0, the user would have to mentally add 8000H to all program addresses when changing or examining the memory. The mapped commands perform this operation automatically.

Example:

In the above case (DBM-1 at 8000H), the following would be needed to set the DBM/target address mapping

```
>08000          (set the offset)
>.S0            (examine target address 0/DBM address 8000H)
```


system may use the monitor as provided

Only commands which access memory can be mapped; preceeding any of the other monitor commands with a period has no effect.

The mapped commands are designed to map a target address space from 0 - FFFH (0 - 4096 bytes) into one or two DBM-1s. More than 4K of contiguous debug storage can be used by adding more cards and using the same procedure to set the address offset for the S-100 address block, i.e., three DBM-1s from 8000H to 97FFH in the S-100 system would still use offset = 8000H.

Using more than two DBM-1s has an effect on the address trap. The trap is only 12 bits long, and is thus limited to a range of 4096 bytes. The trap logic and how it is set is discussed more completely in section 7.12.

7.3 General Command Format

All debug monitor commands are input from the system keyboard, and all outputs are directed to the system display. These drivers are located within the monitor and may be changed to adapt the program to any system.

The monitor command prompt is the greater than symbol (>). Whenever this prompt is displayed the program is waiting for user input. All input lines are terminated by carriage return. If more than one value is required by a command, all required values should be enterred separated by commas, the last value terminated with a carriage return.

All numberic data is input in hexadecimal form. Leading 0's do not need to be enterred, and enterring more than 4 digits causes the last four digits enterred to be used; i.e., enterring 10001 would be the same as enterring 1.

All monitor error conditions display the error prompt '?' and re-prompt for a new command.

7.4 A Command - Hexadecimal Arithmetic

The A command is used to add, subtract, multiply, and divide two hexadecimal numbers. The first number is input, followed by one of the operators + - * /, followed by the second number. The result is displayed on the next line.

Example:

```
>A1000+6
1006
>A1000-5
FFFB
>A135E*9
AE4E
>A1000/6
02AA
>
```

The arithmetic commands are useful for computing offset addresses, program constants, memory displacements, or data values.

7.5 C Command - Set Multiple DBM-1 Chain Flag

The C command is used if two DBM-1s are being used together to form a single 4096 byte address block. This is done by connecting the DBM-1 trap daisy chain lines together (see section 5.1). In these applications the trap address is written at locations 4094 and 4095 of the block, rather than at locations 2046 and 2047 of a single DBM-1. The monitor initializes to one DBM-1, but users with two DBM-1s can modify the monitor to change the default.

When using the C command, C1 means set one DBM-1, C2 means set two DBM-1s. 1 and 2 are the only legal operators.

7.6 D Command - Display Memory Block

The D command is used to display a block of memory locations. The command expects two parameters, separated by commas. The contents of the range of locations specified will be displayed on the console. The command is valid for mapped operation, and will display the location specified offset by the current DBM-1 memory block offset constant (see Sections 7.2 and 7.9).

Example:

```
>D100,11F
0100 00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F
0110 10 11 12 13 14 15 16 17 18 19 1A 1B 1C 1D 1E 1F
>
```

7.7 F Command - Fill Memory With Constant

The F command is used to fill a block of memory with an eight bit data constant. Three parameters are input separated by commas. The first parameter is the start address, the second is the end address, and the third is the hexadecimal data constant. The command is valid for mapped operations.

Example:

The following example would fill locations 100-1FF with zeros.

```
>F100,1FF,0
>
```

7.8 L Command - Clear Trap LED

The L command accesses the memory block to turn off the trap LED. This is useful during program debugging. Program execution can be traced by setting traps, watching the trap LED, clearing the LED, debugging the program section, and setting a new trap. (Note: setting a new trap address automatically clears the trap LED.)

7.9 M Command - Move Memory Block

The M command is used to move a block of memory locations from one point in the memory to another. The command expects three parameters. The first is the starting address of the block to be moved, the second is the end address of the block to be moved, and the third is the first address of the destination. The command is valid for mapped operations.

Example:

The following example would copy the block of locations from 1000-100F to locations 1100-110F.

```
>M1000,100F,1100
>
```

7.10 O Command - Set DBM-1 Memory Offset

The O command sets the memory mapping offset for use with the other debug monitor commands. The offset will be added to all mapped commands when they are input preceeded by a period. The use of the offset simplifies program changes and examinations by eliminating the need to mentally add the bias to all memory access commands involving the DBM-1. The default offset is 0 (not mapped), but this can easily by changed to suit the user's particular system.

7.11 S Command - Examine/Substitute Memory

The S command is used to examine and change the contents of system memory locations. The command is valid for mapped operations.

The command initially expects one parameter, namely the location where the operation is to begin. It displays the contents of that location and waits for a user input. A carriage return causes the next location to be displayed, a minus sign (-) causes the previous location to be displayed, a number from 00-FF will replace the current contents of the location and a period (.) will return to the monitor command loop.

Example:

The following example shows the use of the S command.

```
>S100
100 00
101 01 35
102 02 -
101 35 .
>
```

7.12 T Command - Set DBM-1 Address Trap

The T command is used to set the DBM-1 hardware address trap. The trap is used to indicate when the target program has reached a certain point. The trap address is stored in the two most significant bytes of the DBM-1 memory block. If a single DBM-1 is used the locations are 2046 and 2047 (0FFEH and 0FFFH); if two DBM-1s are chained (indicated by using the C2 command) the locations are 4094 and 4095 (OFFEH and OFFFH).

Example:

The following example shows how to set the DBM-1 address trap at location 135 of the target program. When this location is accessed by the target processor, the trap LED will light and the address trap match signals will be generated by the DBM-1 hardware.

```
>T135
>
```

The trap will remain set until a new value is enterred.

When using the trap address it is important to understand both the trap logic and how it relates to the DBM-1 address space. The algorithm used for computing the locations where the DBM-1 trap address is written is as follows:

If one DBM-1,

M = OFFSET + 800H - 1

If two DBM-1s,

M = OFFSET + 1000H - 1

For example, in a system with a single DBM-1 located at 8000H in the S-100 system, M = 8000H + 800H - 1 = 87FFH. The most significant byte of the trap address would be written at location 87FFH; the least significant byte would be written at 87FEH.

The trap address itself is only 12 bits long and can only be set for a 4096 byte range of addresses. The debug monitor sets the address trap for the 4K block; the most significant bits are not used. Thus, enterring a trap address of 16FEH would result in a trap

address of 6FEH being used. Users with more than 4K of debug storage or emulating application computer memory outside the range 0 - FFFH may want to modify the trap and/or offset routines to match their system requirements.

7.13 V Command - Verify Memory

The V command is a simple memory test designed to detect bad devices. The command expects two parameters. The first is the start address of the block to be tested and the second is the end address of the block. The program writes 256 patterns (0 - FF) into each location in the block. The original contents of the block are not changed. If the block passes, the monitor returns with the normal prompt. If a location fails, the address, value written, and value read are displayed. The test stops on the first location which fails.

Example:

The following example shows the output if, while testing the memory block from 1000H to 1FFFH, locations 1067 is written with a 10H and reads back a 11H (bit 0 stuck).

```
>V1000,1FFF
1067 W10 R11
>
```

7.14 Monitor Modifications

Debug monitor is provided in source form to make it easy to implement user modifications. The most common modifications will involve the I/O and the location of the actual program. These are discussed in the following sections.

7.14.1 I/O Modifications

All debug monitor I/O is performed using the routines CONIN (Console Input) and CONOUT (Console Output). CONIN should return an upper case ASCII character with the parity bit set to 0 in the A register. CONOUT should be passed an ASCII character in the C register.

The system where debug monitor was developed (CP/M) automatically echoes characters input via CONIN to the console. The user system may need to add a routine to do this. For example:

```
CALL  CONIN
MOV   C,A
CALL  CONOUT
MOV   A,C
```

7.14.2 Memory Changes

Debug monitor is ORGed at 100H to run with a CP/M development system. This can be changed by changing the ORG statement. The program can be placed in PROM by ORGing the program in the PROM area and ORGing the system RAM in an area of scratch RAM. The program is under 1K byte long and will fit into a single 2708/2758 EPROM. The program uses 51 bytes of RAM, and most of this is reserved for the 8085 stack.

Example:

The following ORG statements would locate Debug Monitor at 0C000H with its RAM at location 100H.

```
ORG 0C000H          ;PLACE PROGRAM AT 0C000H

(program body)

ORG 100H            ;PLACE RAM AT 100H
```

7.15 Loading and Saving Programs

Because of the wide variety of systems which can use DBM-1, the Debug Monitor does not include any commands for loading or saving the object program to be run by the application computer. Small programs can be enterred directly using the S command. Larger programs can normally be loaded directly into the DBM-1 using the system disk or tape load utilities. The spare commands in the monitor master command table can be used to add the system calls required to perform file loading and saving in the users system.

Users may also want to add command to control an EPROM programmer. Once the program is debugged using DBM-1 it can be transferred to an EPROM and installed in the target computer for final testing.

```
F/M MACRO ASSEM 2.0     #001    PRAGMATIC DESIGNS DEBUG MONITOR

                        ;
                        ;       TITLE   'PRAGMATIC DESIGNS DEBUG MONITOR'
                        ;
                        ;
                        ;
                        ;       PRAGMATIC DESIGNS, INC. DEBUG MONITOR
                        ;
                        ;       REV. 1.0
                        ;
                        ;       COPYRIGHT (C) 1978, 1979
                        ;       ALL RIGHTS RESERVED
                        ;
                        ;
000D =                  CR      EQU     0DH
000A =                  LF      EQU     0AH
003E =                  PROMPT  EQU     '>'
003F =                  HUH     EQU     '?'
0005 =                  SYSTEM  EQU     5
                        ;
                        ;
                        ;
                        ;       INITIALIZE SYSTEM
                        ;
 100                            ORG     100H    ;ORG IN CP/M TPA
 100 C34B01                     JMP     BEGIN
0103 0D0A505241SIGNON:  DB      CR,LF,'PRAGMATIC DESIGNS DEBUG MONITOR, REV. 1.0'
012E 0D0A0D0A43                DB      CR,LF,CR,LF,'COPYRIGHT (C) 1978, 1979',0FFH
                        ;
                        ;
014B =                  BEGIN   EQU     $
 14B 31C904                     LXI     SP,STACK
014E 210000                     LXI     H,0     ;SET INITIAL OFFSET = 0
0151 22CA04                     SHLD    OFFSET
0154 AF                         XRA     A       ;SET INITIAL VALUE = NOT CHAINED
0155 32CC04                     STA     CHAINF
0158 210301                     LXI     H,SIGNON
015B CD0E04                     CALL    TYPE    ;DISPLAY LOGON MESSAGE
015E CDFF03                     CALL    CRLF
                        ;
                        ;       MAIN COMMAND LOOP
                        ;
0161 =                  START   EQU     $
0161 FB                         EI              ; ENABLE INTERRUPTS
0162 CDFF03                     CALL    CRLF
0165 0E3E                       MVI     C,PROMPT
0167 CDE103                     CALL    CONOUT  ;DISPLAY PROMPT
016A AF                         XRA     A       ;ASSUME NORMAL COMMAND
016B 32CD04                     STA     CFLAG
016E CDEE03                     CALL    CONIN   ; GET A CHARACTER
0171 FE2E                       CPI     '.'     ;. => MAPPED COMMAND
0173 C27E01                     JNZ     START1
0176 3E01                       MVI     A,1
0178 32CD04                     STA     CFLAG   ;SET FLAG
017B CDEE03                     CALL    CONIN   ;GET ACTUAL COMMAND
                        ;
017E =                  START1  EQU     $
```


...can be used to add the system calls

```
P/M MACRO ASSEM 2.0     #002    PRAGMATIC DESIGNS DEBUG MONITOR

017E D641                       SUI     'A'         ; TEST FOR A-X
0180 FA6101                     JM      START       ; LESS THAN 'A' RETURN TO START
0183 FE16                       CPI     'V'-'A'+1
0185 F2D303                     JP      ERROR       ; GREATER THAN 'S' RETURN TO START
0188 87                         ADD     A           ; * 2
0189 219301                     LXI     H,CTBLE     ; FWA OF COMMAND TABLE
018C 85                         ADD     L
018D 6F                         MOV     L,A         ; COMPUTE TABLE ADDRESS, PUT IN H&L
018E 7E                         MOV     A,M         ; GET LSB OF ADDRESS
018F 23                         INX     H           ; POINT TO NEXT ADDRESS
0190 66                         MOV     H,M         ; GET MSB OF ADDRESS
0191 6F                         MOV     L,A         ; LSB TO L REGISTER
0192 E9                         PCHL                ; BRANCH TO COMMANDED ROUTINE
                        ;
                        ;       COMMAND BRANCH TABLE
                        ;
                        ;
0193 =                  CTBLE   EQU     $
0193 BF01                       DW      ARITH       ; A - ARITHMETIC WITH HEX NUMBERS
0195 D303                       DW      ERROR       ; B - SPARE COMMAND
0197 2702                       DW      CHAIN       ; C - SET CHAIN FLAG
0199 3B02                       DW      DUMP        ; D - DUMP MEMORY
019B D303                       DW      ERROR       ; E - SPARE COMMAND
019D 7D02                       DW      FILL        ; F - FILL MEMORY WITH A CONSTANT
019F D303                       DW      ERROR       ; G - SPARE COMMAND
01A1 D303                       DW      ERROR       ; H - SPARE COMMAND
01A3 D303                       DW      ERROR       ; I - SPARE COMMAND
01A5 D303                       DW      ERROR       ; J - SPARE COMMAND
01A7 D303                       DW      ERROR       ; K - SPARE COMMAND
01A9 A802                       DW      LTRAP       ; L - CLEAR TRAP LED
01AB AF02                       DW      MOVE        ; M - MOVE MEMORY
01AD D303                       DW      ERROR       ; N - SPARE COMMAND
01AF 1D03                       DW      SETOFF      ; O - SET DBM OFFSET ADDRESS
01B1 D303                       DW      ERROR       ; P - SPARE COMMAND
01B3 D303                       DW      ERROR       ; Q - SPARE COMMAND
01B5 D303                       DW      ERROR       ; R - SPARE COMMAND
01B7 2703                       DW      SUBS        ; S - SUBSTITUTE MEMORY
01B9 7A03                       DW      TRAP        ; T - SET DBM-1 ADDRESS TRAP
01BB D303                       DW      ERROR       ; U - SPARE COMMAND
01BD 8A03                       DW      VERIFY      ; V - VERIFY MEMORY (TEST RAM)
                        ;
                                PAGE
```

```
                        ;
                        ;       ARITH - PERFORM HEX ARITHMETIC
                        ;
01BF =          ARITH   EQU     $
01BF CD4204             CALL    ARIN    ;GET VALUE AND DELIMITER
01C2 78                 MOV     A,B     ;TEST DELIMITER
01C3 FE0D               CPI     CR      ;ILLEGAL HERE
01C5 CAD303             JZ      ERROR
01C8 48                 MOV     C,B     ;SAVE AS OPERATOR
01C9 EB                 XCHG            ;FLIP VALUE INTO DE
01CA CD4204             CALL    ARIN    ;GET SECOND
01CD 78                 MOV     A,B     ;TEST DELIMTER
01CE FE0D               CPI     CR
01D0 C2D303             JNZ     ERROR   ;ONLY LEGAL
01D3 41                 MOV     B,C     ;RESTORE DELIMITER
01D4 EB                 XCHG            ;PARM1 -> HL ; PARM2 -> DE
01D5 CDFF03             CALL    CRLF    ;OUTPUT LINES
01D8 78                 MOV     A,B
01D9 FE2B               CPI     '+'
01DB CAF001             JZ      ADDS    ;ADD
01DE FE2D               CPI     '-'
01E0 CAF401             JZ      SUBTS   ;SUBTRACT
01E3 FE2A               CPI     '*'
01E5 CAFD01             JZ      MULTS   ;MULTIPLY
01E8 FE2F               CPI     '/'
01EA CA0C02             JZ      DIVS    ;DIVIDE
01ED C3D303             JMP     ERROR   ;ILLEGAL
                        ;
                        ;  ADD NUMBERS
                        ;
01F0 =          ADDS    EQU     $
01F0 19                 DAD     D
01F1 C32102             JMP     ARITH1  ;THATS ALL FOLKS
                        ;
                        ;  SUBTRACT
                        ;
01F4 =          SUBTS   EQU     $
01F4 7D                 MOV     A,L
01F5 93                 SUB     E
01F6 6F                 MOV     L,A
01F7 7C                 MOV     A,H
01F8 9A                 SBB     D
01F9 67                 MOV     H,A
01FA C32102             JMP     ARITH1
                        ;
                        ;  MULTIPLY - (REPEAT ADD)
                        ;
01FD =          MULTS   EQU     $
01FD 4D                 MOV     C,L
01FE 44                 MOV     B,H
01FF 210000             LXI     H,0
                        ;
0202 =          MULTS1  EQU     $
0202 7B                 MOV     A,E
0203 B2                 ORA     D
```

```
CP/M MACRO ASSEM 2.0     #004     PRAGMATIC DESIGNS DEBUG MONITOR

0204 CA2102              JZ       ARITH1
0207 09                  DAD      B          ;ADD 1
0208 1B                  DCX      D          ;DECREMENT COUNTER
0209 C30202              JMP      MULTS1
                    ;
                    ;  DIVIDE - (REPEAT SUBTRACT)
                    ;
020C =              DIVS EQU      $
020C 7B                  MOV      A,E
020D B2                  ORA      D          ;DIVIDE BY 0?
020E CAD303              JZ       ERROR      ;ILLEGAL
0211 7B                  MOV      A,E        ;FORM 2'S COMPLEMENT
0212 2F                  CMA
0213 4F                  MOV      C,A
0214 7A                  MOV      A,D
0215 2F                  CMA
0216 47                  MOV      B,A
0217 03                  INX      B
0218 11FFFF              LXI      D,-1
                    ;
021B =              DIVS1 EQU     $
021B 13                  INX      D
021C 09                  DAD      B          ;SUBTRACT
021D DA1B02              JC       DIVS1
0220 EB                  XCHG                ;GET QUOTIENT
                    ;
                    ;
0221 =              ARITH1 EQU    $
0221 CD7104              CALL     LADR       ;PRINT ADDRESS
0224 C36101              JMP      START
                    ;
                    ;        CHAIN - SET DBM-1 CHAIN FLAG.  1 = 1 DBM-1 (DEFAULT)
                    ;                                       2 = 2 DBM-1'S
                    ;
                    ;
0227 =              CHAIN EQU     $
0227 CD1A04              CALL     EXPR1      ;GET 1 VALUE
022A E1                  POP      H
022B 7D                  MOV      A,L        ;USE LSB AS FLAG
022C 3D                  DCR      A
022D FAD303              JM       ERROR
0230 FE02                CPI      2          ;  1 & 2 ONLY LEGAL VALUES
0232 F2D303              JP       ERROR
0235 32CC04              STA      CHAINF
0238 C36101              JMP      START
                    ;
                    ;
                    ;        DUMP - DISPLAY MEMORY CONTENTS ON CONSOLE
                    ;
023B =              DUMP   EQU    $
023B CD1F04              CALL     EXPR2      ;GET 2 PARAMETERS
023E D1                  POP      D          ; GET HIGH ADDRESS INTO D
023F E1                  POP      H          ; GET LOW ADDRESS INTO H
                    ;
0240 =              DUMP1  EQU    $
0240 CDFF03              CALL     CRLF
0243 CD7104              CALL     LADR       ; PRINT MEMORY ADDRESS
```

```
                        ;
0246 =                  DUMP2   EQU     $
0246 CD0904                     CALL    SPACE   ; PRINT A SPACE
0249 3ACD04                     LDA     CFLAG   ;MAPPED?
024C B7                         ORA     A
024D CA5B02                     JZ      DUMP3   ;NO
0250 22D004                     SHLD    STARTB  ;SAVE ORIGINALS
0253 EB                         XCHG
0254 22D204                     SHLD    ENDB
0257 2ACA04                     LHLD    OFFSET  ;ADD OFFSET BIAS
025A 19                         DAD     D
                        ;
025B =                  DUMP3   EQU     $
025B 7E                         MOV     A,M
025C CD7604                     CALL    LBYTE   ; PRINT DATA
025F 3ACD04                     LDA     CFLAG   ;MAPPED?
0262 B7                         ORA     A
0263 CA6D02                     JZ      DUMP4   ;NO
0266 2AD204                     LHLD    ENDB    ;RECOVER ORIGINALS
0269 EB                         XCHG
026A 2AD004                     LHLD    STARTB
                        ;
026D =                  DUMP4   EQU     $
026D 23                         INX     H       ;MOVE TO NEXT LOC.
026E CD6804                     CALL    CMPHLDE ; TEST FOR COMPLETION
0271 DA6101                     JC      START   ; RETURN TO COMMAND ROUTINE
0274 7D                         MOV     A,L
0275 E60F                       ANI     0FH     ; PRINT CRLF, ADDRESS ON MULTIPLE OF 16
0277 C24602                     JNZ     DUMP2
027A C34002                     JMP     DUMP1
                        ;
                        ;       FILL - FILL RAM WITH A CONSTANT
                        ;
027D =                  FILL    EQU     $
027D CD2404                     CALL    EXPR3   ;GET 3 PARAMETERS
0280 C1                         POP     B       ; PUT CONSTANT IN B
0281 D1                         POP     D       ; GET HIGH ADDRESS
0282 3ACD04                     LDA     CFLAG   ;MAPPED?
0285 B7                         ORA     A
0286 CA8E02                     JZ      FILL1   ;NO
0289 2ACA04                     LHLD    OFFSET  ;ADD BIAS
028C 19                         DAD     D
028D EB                         XCHG
                        ;
028E =                  FILL1   EQU     $
028E E1                         POP     H       ; GET LOW ADDRESS
028F 3ACD04                     LDA     CFLAG   ;MAPPED?
0292 B7                         ORA     A
0293 CA9D02                     JZ      FILL2   ;NO
0296 D5                         PUSH    D
0297 EB                         XCHG
0298 2ACA04                     LHLD    OFFSET  ;ADD OFFSET
029B 19                         DAD     D
029C D1                         POP     D
                        ;
029D =                  FILL2   EQU     $
```

```
P/M MACRO ASSEM 2.0     #006    PRAGMATIC DESIGNS DEBUG MONITOR

029D 71                 MOV     M,C     ; STORE CONSTANT IN MEMORY
029E 23                 INX     H       ;MOVE TO NEXT LOCATION
029F CD6804             CALL    CMPHLDE ; TEST FOR COMPLETION
02A2 D29D02             JNC     FILL2   ; CONTINUE IF NOT DONE
02A5 C36101             JMP     START   ; RETURN TO COMMAND ROUTINE
                ;
                ;       LTRAP - CLEAR TRAP LED
                ;
02A8 =          LTRAP   EQU     $
02A8 CD5604             CALL    LTOP    ;GET TOP ADDRESS OF BLOCK
02AB 7E                 MOV     A,M     ;READ CLEARS LED
02AC C36101             JMP     START
                ;
                ;       MOVE - MOVE A BLOCK OF MEMORY
                ;
02AF =          MOVE    EQU     $
02AF CD2404             CALL    EXPR3   ;GET 3 PARAMETERS
02B2 E1                 POP     H       ;GET DESTINATION ADDRESS
02B3 3ACD04             LDA     CFLAG   ;MAPPED?
02B6 B7                 ORA     A
02B7 CABF02             JZ      MOVE1
02BA EB                 XCHG
02BB 2ACA04             LHLD    OFFSET  ;ADD OFFSET
02BE 19                 DAD     D
                ;
02BF =          MOVE1   EQU     $
02BF 22CE04             SHLD    DESTB   ;SAVE IT
02C2 E1                 POP     H       ;GET END ADDRESS
02C3 CAC802             JZ      MOVE2   ;NOT MAPPED
02C6 EB                 XCHG
02C7 2ACA04             LHLD    OFFSET  ;ADD OFFSET
02CA 19                 DAD     D
                ;
02CB =          MOVE2   EQU     $
02CB 22D204             SHLD    ENDB    ;SAVE IT
02CE EB                 XCHG            ;FLIP INTO DE
02CF E1                 POP     H       ;GET START ADDRESS
02D0 CADA02             JZ      MOVE3   ;NOT MAPPED
02D3 D5                 PUSH    D
02D4 EB                 XCHG
02D5 2ACA04             LHLD    OFFSET  ;ADD OFFSET
02D8 19                 DAD     D
02D9 D1                 POP     D
                ;
02DA =          MOVE3   EQU     $
02DA 22D004             SHLD    STARTB
02DD 7B                 MOV     A,E     ;COMPUTE BLOCK SIZE
02DE 95                 SUB     L
02DF 4F                 MOV     C,A     ;AND PLACE IT IN BC
02E0 7A                 MOV     A,D
02E1 9C                 SBB     H
02E2 47                 MOV     B,A
02E3 DAD303             JC      ERROR   ;IF DE < HL, ERROR
02E6 03                 INX     B
02E7 2ACE04             LHLD    DESTB
02EA 7B                 MOV     A,E     ;TEST FOR DEST < END
```

```
P/M MACRO ASSEM 2.0     #007    PRAGMATIC DESIGNS DEBUG MONITOR

0 EB 95                 SUB     L
02EC 7A                 MOV     A,D
02ED 9C                 SBB     H
0 EE EB                 XCHG                ;NOW SEE IF DEST > START
0 EF 2AD004             LHLD    STARTB
02F2 DAFC02             JC      MOVE4       ;CARRY => NORMAL MOVE
02F5 7B                 MOV     A,E
0 F6 95                 SUB     L
02F7 7A                 MOV     A,D
02F8 9C                 SBB     H
0 F9 D20903             JNC     MOVE5       ;CARRY => NORMAL MOVE
                ;
                ;  NORMAL MOVE.  BC = LENGTH, HL = START, DE = DEST
                ;
0 FC =          MOVE4   EQU     $
02FC 7E                 MOV     A,M
02FD 12                 STAX    D
0 FE 23                 INX     H
02FF 13                 INX     D
0300 0B                 DCX     B
0 01 79                 MOV     A,C
0 02 B0                 ORA     B
0303 C2FC02             JNZ     MOVE4
0306 C36101             JMP     START
                ;
                ;  INVERTED MOVE.
                ;
 309 =          MOVE5   EQU     $
0309 EB                 XCHG                ;ADD LENGTH TO DEST
030A 09                 DAD     B
 30B 2B                 DCX     H           ;COMPENSATE FOR LENGTH
 30C EB                 XCHG
030D 2AD204             LHLD    ENDB        ;ADD LENGTH TO END
                ;
 310 =          MOVE6   EQU     $
0310 7E                 MOV     A,M         ;DO INVERTED MOVE
0311 12                 STAX    D
 312 2B                 DCX     H
 313 1B                 DCX     D
0314 0B                 DCX     B
 315 79                 MOV     A,C
 316 B0                 ORA     B
0317 C21003             JNZ     MOVE6
031A C36101             JMP     START
                ;
                ;       SETOFF - SET DBM-1 OFFSET ADDRESS
                ;
 31D =          SETOFF  EQU     $
 31D CD1A04             CALL    EXPR1       ;GET 1 PARAMETER
0320 E1                 POP     H
 321 22CA04             SHLD    OFFSET      ;SAVE IT
 324 C36101             JMP     START
                ;
                ;       SUBS - SUBSTITUTE MEMORY CONTENTS ROUTINE
                ;
                ;
```

```
*/M MACRO ASSEM 2.0     #008    PRAGMATIC DESIGNS DEBUG MONITOR

0327 =          SUBS    EQU     $
0327 CD1A04             CALL    EXPR1   ;GET 1 PARAMETER
032A E1                 POP     H
                ;
032B =          SUBS1   EQU     $
032B CDFF03             CALL    CRLF
032E CD7104             CALL    LADR    ;DISPLAY ADDRESS
0331 CD0904             CALL    SPACE   ;DISPLAY SPACE
0334 22D004             SHLD    STARTB  ;SAVE USER INPUT ADDRESS
0337 3ACD04             LDA     CFLAG   ;MAPPED?
033A B7                 ORA     A
033B CA4303             JZ      SUBS2
033E EB                 XCHG
033F 2ACA04             LHLD    OFFSET  ;ADD OFFSET
0342 19                 DAD     D
                ;
0343 =          SUBS2   EQU     $
0343 7E                 MOV     A,M     ; DISPLAY DATA
0344 CD7604             CALL    LBYTE   ;DISPLAY DATA
0347 CD0904             CALL    SPACE
034A CDEE03             CALL    CONIN   ;GET USER CHARACTER
034D FE0D               CPI     CR      ;CR -> GET NEXT BYTE
034F CA6C03             JZ      SUBS3
0352 FE2D               CPI     '-'     ;- => BACK UP
0354 CA7303             JZ      SUBS4
0357 FE2E               CPI     '.'     ;. => DONE
0359 CA6101             JZ      START
                ;
                ;   INPUT DATA AND USE IT TO REPLACE MEMORY BYTE
                ;
035C E5                 PUSH    H       ; SAVE MEMORY ADDRESS
035D 210000             LXI     H,0     ;ENTER EXPRESSION ROUTINE
0360 CD4804             CALL    ARIN2   ;WITH FIRST CH. SET
0363 EB                 XCHG            ; E = NEW VALUE
0364 E1                 POP     H       ; RESTORE MEMORY ADDRESS
0365 73                 MOV     M,E     ; STORE NEW VALUE
0366 78                 MOV     A,B     ; TEST DELIMITER
0367 FE0D               CPI     CR
0369 C2D303             JNZ     ERROR   ;ONLY LEGAL DELIMITER
                ;
036C =          SUBS3   EQU     $
036C 2AD004             LHLD    STARTB  ;RECOVER INPUT ADDRESS
036F 23                 INX     H
0370 C32B03             JMP     SUBS1
0373 =          SUBS4   EQU     $
0373 2AD004             LHLD    STARTB  ;RECOVER INPUT ADDRESS
0376 2B                 DCX     H
0377 C32B03             JMP     SUBS1
                ;
                ;       TRAP - SET DBM-1 ADDRESS TRAP
                ;
037A =          TRAP    EQU     $
037A CD1A04             CALL    EXPR1   ;GET 1 PARAMETER
037D C1                 POP     B       ;SET ADDRESS IN BC
037E CD5604             CALL    LTOP    ;COMPUTE TOP ADDRESS OF DBM BLOCK
0381 78                 MOV     A,B     ;GET MSB
```

```
 382 E60F                ANI     0FH     ;MASK TO 4K RANGE
0384 77                  MOV     M,A     ;SET MSB
0385 2B                  DCX     H
 386 71                  MOV     M,C
 387 C36101              JMP     START
                ;
                ;       VERIFY - TEST MEMORY
                ;
038A =          VERIFY  EQU     $
038A CD1F04             CALL    EXPR2   ;GET 2 PARAMETERS
 38D D1                 POP     D       ;GET END
 38E E1                 POP     H       ;GET START
038F 7B                 MOV     A,E
 390 95                 SUB     L       ;COMPUTE COUNT
 391 4F                 MOV     C,A
0392 7A                 MOV     A,D
 393 9C                 SBB     H
 394 47                 MOV     B,A
0395 DAD303             JC      ERROR   ; END < START ILLEGAL
0398 03                 INX     B
                ;
 399 =          VERIF1  EQU     $
0399 56                 MOV     D,M     ;SAVE CONTENTS
 39A 1E00               MVI     E,0
                ;
039C =          VERIF2  EQU     $
 39C 73                 MOV     M,E     ;WRITE
 39D 7E                 MOV     A,M
039E BB                 CMP     E
039F C2B103             JNZ     VERIF3
 3A2 1D                 DCR     E       ;WRITE 256 PATTERNS/LOC
 3A3 C29C03             JNZ     VERIF2
03A6 72                 MOV     M,D     ;RESTORE ORIGINAL
 3A7 23                 INX     H
 3A8 0B                 DCX     B
03A9 79                 MOV     A,C
03AA B0                 ORA     B
 3AB C29903             JNZ     VERIF1
 3AE C36101             JMP     START
                ;
 3B1 =          VERIF3  EQU     $
 3B1 57                 MOV     D,A     ;SAVE ERRONEOUS VALUE
 3B2 CDFF03             CALL    CRLF
 3B5 CD7104             CALL    LADR    ;WRITE ADDRESS
 3B8 CD0904             CALL    SPACE
 3BB 0E57               MVI     C,'W'   ;SHOW WRITTEN
 3BD CDE103             CALL    CONOUT
 3C0 7B                 MOV     A,E
 3C1 CD7604             CALL    LBYTE
 3C4 CD0904             CALL    SPACE
 3C7 0E52               MVI     C,'R'
 3C9 CDE103             CALL    CONOUT
 3CC 7A                 MOV     A,D     ;SHOW READ
 3CD CD7604             CALL    LBYTE
 3D0 C36101             JMP     START
                ;
```


3F5 7B MOV

```
                ;       ERROR EXIT
                ;
                ;       THIS ABNORMAL EXIT IS EXECUTED FOR ALL MONITOR ERROR
                ;       CONDITIONS
                ;
03D3 =          ERROR   EQU     $
03D3 31C904             LXI     SP,STACK
03D6 CDFF03             CALL    CRLF
03D9 0E3F               MVI     C,HUH
03DB CDE103             CALL    CONOUT
03DE C36101             JMP     START   ; RETURN TO COMMAND ROUTINE
                ;
                ;**********  PROGRAM SUBROUTINES  **********
                ;
                ;
                ;       CONSOLE OUTPUT ROUTINE (CP/M)
                ;
                ;       ENT:    C = CHARACTER TO BE OUTPUT
                ;       EXIT:   N/A
                ;
03E1 =          CONOUT  EQU     $
03E1 C5                 PUSH    B       ;SAVE ALL REGISTERS
03E2 D5                 PUSH    D
03E3 E5                 PUSH    H
03E4 59                 MOV     E,C     ;SET UP FOR CP/M CALL
03E5 0E02               MVI     C,2     ;BDOS CALL 2
03E7 CD0500             CALL    SYSTEM  ;DO IT
03EA E1                 POP     H       ;RESTORE ALL
03EB D1                 POP     D
03EC C1                 POP     B
03ED C9                 RET
                ;
                ;       CONSOLE INPUT ROUTINE (CP/M)
                ;
                ;       ENT:    N/A
                ;       EXIT:   A = CHARACTER INPUT FROM CONSOLE
                ;
03EE =          CONIN   EQU     $
03EE C5                 PUSH    B       ;SAVE ALL REGISTERS
03EF D5                 PUSH    D
03F0 E5                 PUSH    H
03F1 0E01               MVI     C,1     ;BDOS CALL 1
03F3 CD0500             CALL    SYSTEM
03F6 E1                 POP     H
03F7 D1                 POP     D
03F8 C1                 POP     B
03F9 FE61               CPI     61H     ;UPPER CASE?
03FB F8                 RM
03FC E6DF               ANI     NOT 20H ;MASK OFF BIT 6
03FE C9                 RET
                ;
                ;       CRLF - PRINT CARRAGE RETURN AND LINE FEED
                ;
                ;       ENT:    N/A
                ;       EXIT:   N/A
                ;
```

```
 FF =           CRLF    EQU     $
 FF 0E0D                MVI     C,CR
0401 CDE103             CALL    CONOUT
 04 0E0A                MVI     C,LF
 06 C3E103              JMP     CONOUT
                ;
                ;       SPACE - PRINT SPACE
                ;
                ;       ENT:    N/A
                ;       EXIT:   N/A
                ;
 09 =           SPACE   EQU     $
0409 0E20               MVI     C,' '
 0B C3E103              JMP     CONOUT
                ;
                ;       TYPE - TYPE MESSAGE ON CONSOLE
                ;
                ;       ENT:    HL = FWA OF STRING (TERMINATE WITH 0FFH)
                ;       EXIT:   N/A
                ;
 0E =           TYPE    EQU     $
 0E 4E                  MOV     C,M     ; GET CHARACTER
040F 79                 MOV     A,C
 10 FEFF                CPI     0FFH    ; TEST FOR LAST CHARACTER
 12 C8                  RZ              ; RETURN IF DONE
0413 CDE103             CALL    CONOUT
0416 23                 INX     H
 17 C30E04              JMP     TYPE    ; GET NEXT CHARACTER
                ;
                ;       EXPR1, EXPR2, EXPR3 - INPUT FROM 1 - 3 PARAMETERS
                ;
                ;       ENT:    N/A
                ;       EXIT:   VALUES RETURNED IN STACK
                ;
 1A =           EXPR1   EQU     $
 1A 0E01                MVI     C,1
041C C32604             JMP     VALIN
                ;
 1F =           EXPR2   EQU     $
041F 0E02               MVI     C,2
 21 C32604              JMP     VALIN
                ;
0424 =          EXPR3   EQU     $
0424 0E03               MVI     C,3
                ;
                ;  DROP INTO VALIN
                ;
                ;       VALIN - EVALUATE INPUT EXPRESSION
                ;
                ;       ENT:    C = NUMBER OF PARAMETERS TO BE INPUT
                ;       EXIT:   VALUES RETURNED ON STACK
                ;
0426 =          VALIN   EQU     $
0426 CD4204             CALL    ARIN    ;GET VALUE
0429 E3                 XTHL            ; GET RETURN ADDRESS OFF STACK
042A E5                 PUSH    H       ; PUT HL ON
```



```
 2A =           ERROR   PUSH    B       ;SAVE ALL REGISTERS
```

```
'/M MACRO ASSEM 2.0     #012    PRAGMATIC DESIGNS DEBUG MONITOR

042B 78                 MOV     A,B
042C FE2C               CPI     ','     ;COMMA IS LEGAL DELIMITER
042E CA3B04             JZ      VALIN1  ;PROCEED
0431 FE0D               CPI     CR      ;CR => END INPUT
0433 C2D303             JNZ     ERROR
0436 0D                 DCR     C       ;SEE IF REALLY LAST
0437 C2D303             JNZ     ERROR   ;NO
043A C9                 RET
                ;
043B =          VALIN1  EQU     $
043B 0D                 DCR     C
043C C22604             JNZ     VALIN
043F C3D303             JMP     ERROR   ;TOO MANY PARAMETERS
                ;
                ;       ARIN - INPUT SINGLE 16 BIT NUMBER
                ;
                ;       ENT:    N/A
                ;       EXIT:   HL = 16 BIT VALUE, B = DELIMITER
                ;
0442 =          ARIN    EQU     $
0442 210000             LXI     H,0
                ;
0445 =          ARIN1   EQU     $
0445 CDEE03             CALL    CONIN
                ;
0448 =          ARIN2   EQU     $
0448 47                 MOV     B,A     ;SAVE DELIMITER
0449 CD9304             CALL    ASHEX   ;CONVERT INPUT TO HEX
044C D8                 RC
044D 29                 DAD     H
044E 29                 DAD     H
044F 29                 DAD     H
0450 29                 DAD     H
0451 B5                 ORA     L
0452 6F                 MOV     L,A
0453 C34504             JMP     ARIN1
                ;
                ;       LTOP - COMPUTE TOP ADDRESS OF DBM-1 BLOCK
                ;
                ;       ENT:    N/A
                ;       EXIT:   HL = TOP ADDRESS OF BLOCK
                ;
0456 =          LTOP    EQU     $
0456 2ACA04             LHLD    OFFSET  ;GET BASE ADDRESS
0459 EB                 XCHG            ;FLIP INTO DE
045A 210008             LXI     H,2048  ;BLOCK SIZE IF NOT CHAINED
045D 3ACC04             LDA     CHAINF  ;TEST FOR CHAINED
0460 B7                 ORA     A
0461 CA6504             JZ      $+4     ;0 => CHAINED
0464 29                 DAD     H       ;HL = 4096
0465 19                 DAD     D       ;ADD OFFSET
0466 2B                 DCX     H       ;FORM TOP - 1
0467 C9                 RET
                ;
                ;       CMPHLDE - COMPARE HL WITH DE
                ;
```

```
                        ;       IF HL < DE THEN CARRY = 0
                        ;       IF HL = DE THEN CARRY = 0
                        ;       IF HL > DE THEN CARRY = 1
                        ;
0468 =                  CMPHLDE EQU     $
0468 7C                         MOV     A,H     ; TEST FOR HL = 0
0469 B5                         ORA     L
046A 37                         STC
046B C8                         RZ
046C 7B                         MOV     A,E     ; DE - HL, SET/RESET CARRY
046D 95                         SUB     L
046E 7A                         MOV     A,D
046F 9C                         SBB     H
0470 C9                         RET
                        ;
                        ;       LADR - PRINT 16 BIT NUMBER ON CONSOLE
                        ;
                        ;       ENT:    16 BIT NUMBER IN HL
                        ;       EXIT:   N/A
                        ;
0471 =                  LADR    EQU     $
0471 7C                         MOV     A,H     ; PRINT MSB
0472 CD7604                     CALL    LBYTE
0475 7D                         MOV     A,L     ; PRINT LSB
                        ;
                        ;  DROP INTO LBYTE
                        ;
                        ;       LBYTE - LIST A BYTE AS 2 ASCII CHARACTERS
                        ;
                        ;       ENT:    A = 8 BIT HEX BYTE
                        ;       EXIT:   N/A
                        ;
0476 =                  LBYTE   EQU     $
0476 F5                         PUSH    PSW     ; SAVE A COPY OF 'A'
0477 0F                         RRC
0478 0F                         RRC
0479 0F                         RRC
047A 0F                         RRC
047B CD8A04                     CALL    HEXAS
047E 4F                         MOV     C,A     ;MOVE ASCII CH. TO C
047F CDE103                     CALL    CONOUT  ;PRINT
0482 F1                         POP     PSW     ; RETRIEVE ORIGINAL VALUE
0483 CD8A04                     CALL    HEXAS   ;CONVERT TO ASCII
0486 4F                         MOV     C,A
0487 C3E103                     JMP     CONOUT  ;PRINT IT
                        ;
                        ;       HEXAS - CONVERT HEX NIBBLE TO ASCII
                        ;
                        ;       ENT:    A = HEX NIBBLE, 0-F
                        ;       EXIT:   A = ASCII CHARACTER, 30H-39H, 41H-46H
                        ;
048A =                  HEXAS   EQU     $
048A E60F                       ANI     0FH     ;REMOVE MSB
048C C690                       ADI     90H
048E 27                         DAA
048F CE40                       ACI     40H
```

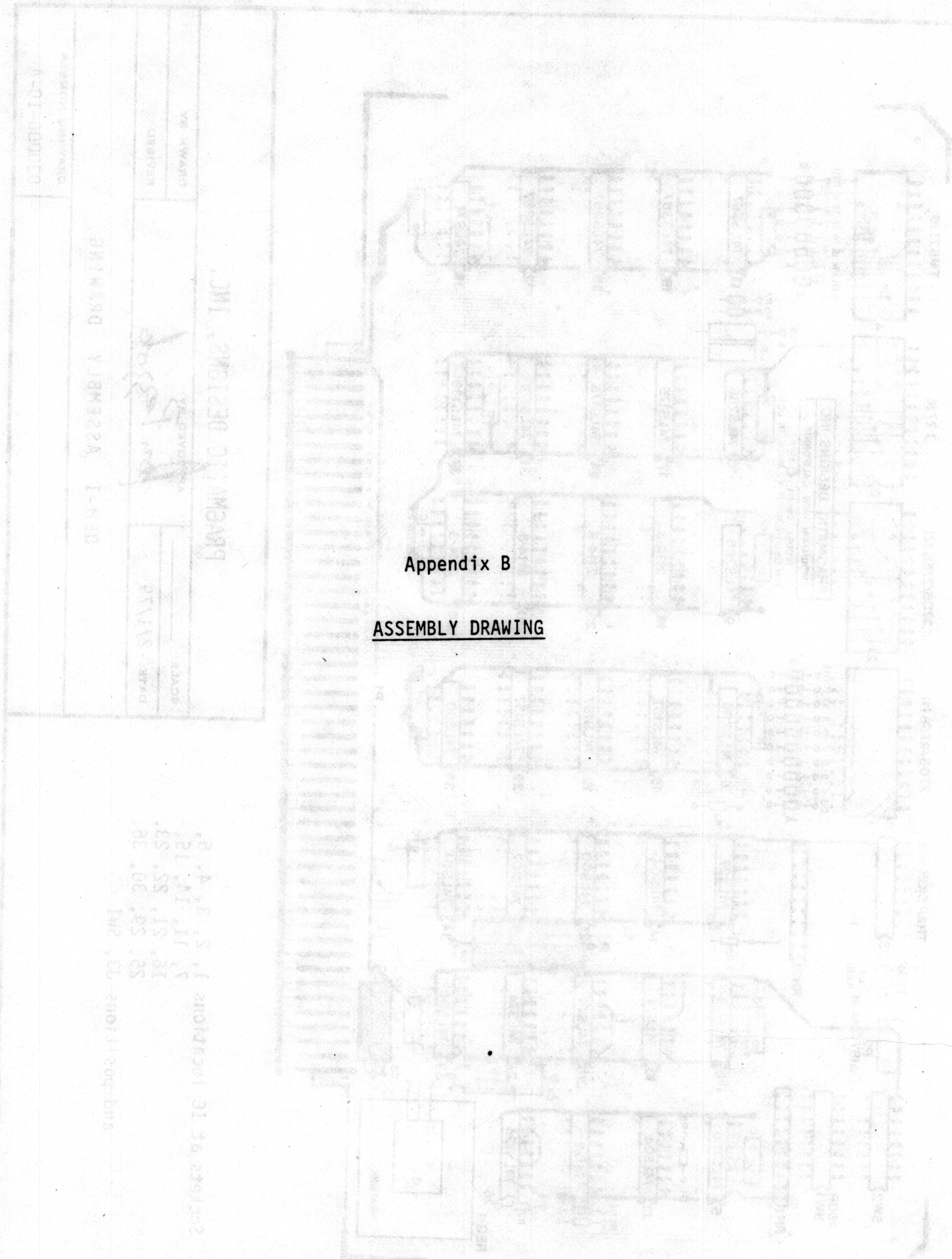


```
                        ;       EXPR1, EXPR2, EXPR3 - INPUT FROM 1 TO 3 PARAMETERS
```

```
0491 27                 DAA
0492 C9                 RET
                ;
                ;       ASHEX - CONVERT ASCII CHARACTER TO HEX
                ;
                ;       ENT:    A = ASCII CHARACTER 30H-39H, 41H-46H
                ;       EXIT:   A = HEX NIBBLE, 0 - F
                ;
0493 =          ASHEX   EQU     $
0493 FE47               CPI     'F'+1   ; > F
0495 3F                 CMC
0496 D8                 RC
0497 D630               SUI     '0'     ; < 0
0499 D8                 RC
049A FE0A               CPI     10      ; IF < 10, DONE
049C 3F                 CMC
049D D0                 RNC
049E D607               SUI     7       ;SCALE A - F DOWN
04A0 C9                 RET
                ;
                ;  RAM STORAGE (LOCATE IN ANY CONVENIENT PLACE)
                ;
04A1                    DS      40      ;64 BYTE STACK
04C9            STACK:  DS      1
04CA            OFFSET: DS      2       ;OFFSET ADDRESS
04CC            CHAINF: DS      1       ;CHAIN FLAG
04CD            CFLAG:  DS      1       ;S100/TARGET SYSTEM COMMAND FLAG
04CE            DESTB:  DS      2       ;BLOCK MOVE DESTINATION
04D0            STARTB: DS      2       ;BLOCK MOVE START
04D2            ENDB:   DS      2       ;BLOCK MOVE END
04D4                    END
```

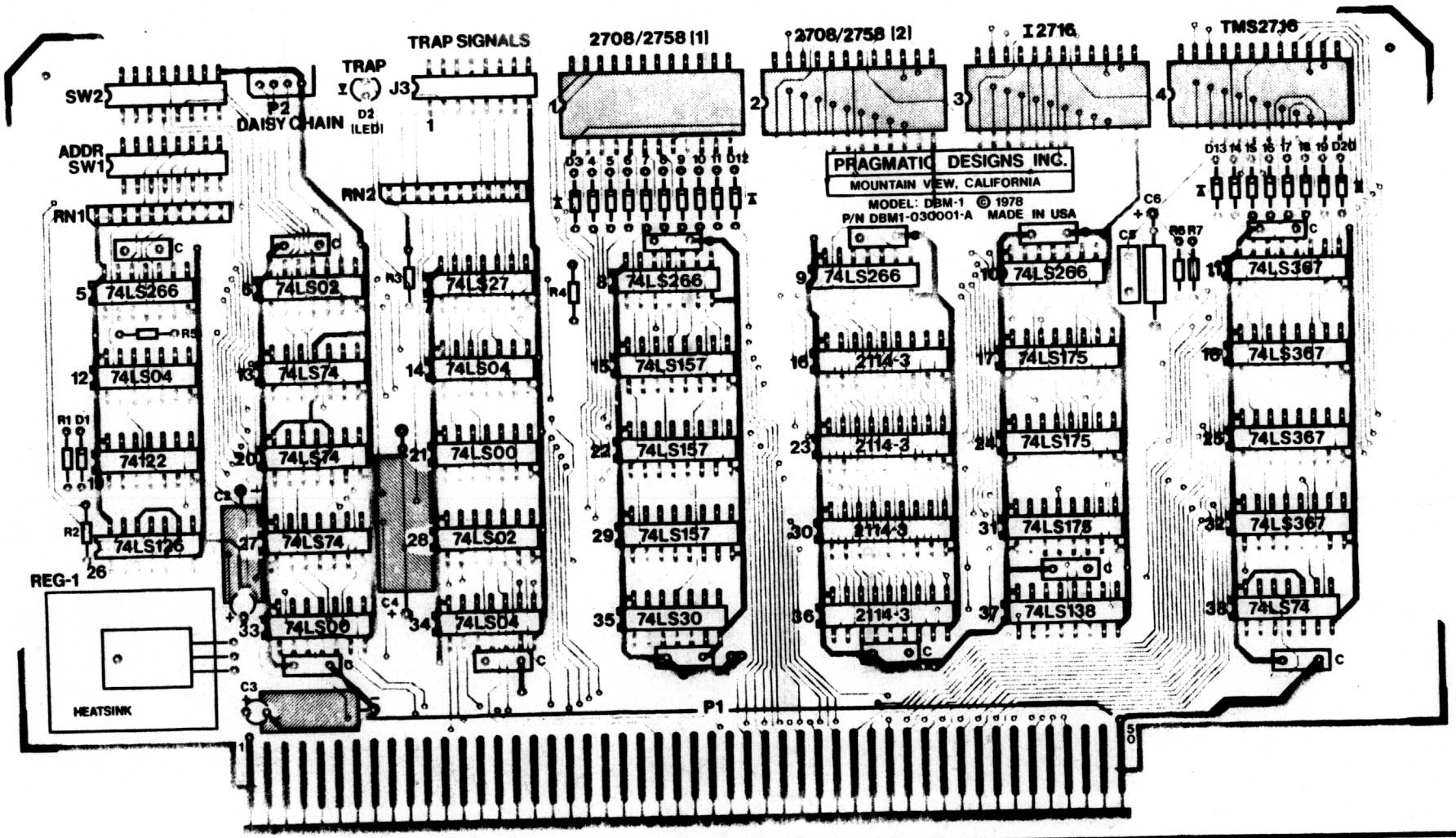
Appendix A

SCHEMATIC DIAGRAM



Appendix B

ASSEMBLY DRAWING



Sockets at IC locations 1, 2, 3, 4, 6,
7, 11, 14, 15,
16, 21, 22, 23,
25, 29, 30, 36,
and positions J3, SW1

PRAGMATIC DESIGNS, INC.		
SCALE:	APPROVED BY:	DRAWN BY
DATE: 2/1/79		REVISED
DBM-1 ASSEMBLY DRAWING		
		DRAWING NUMBER 030000-10-A

Appendix C

PARTS LIST

DRN	DATE	
CHKD	DATE	
APPD Tom Peyton	DATE 3/6/79	
ECO	BY	APP
ECO	BY	APP
ECO	BY	APP
ECO	BY	APP

PRAGMATIC DESIGNS, INC.

MOUNTAIN VIEW, CALIF.

TITLE	PART NUMBER
DBM-1 DEBUG MEMORY BOARD	030000-05-A

COMPONENT CLASS Product Family	NEXT ASSY	SHEET 1 OF 2

ITEM	PRAGMATIC PART NO	QTY	DRAWING REF	DESCRIPTION		
01	030001-01-A	1		P.C. Board	✓	
02	030002-09-A	1		DBM-1 Manual	✓	
03	1000	2	21,33	IC, Quad 2 Input NAND Gate #74LS00		
04	1002	2	6,28	IC, Quad 2 Input NOR Gate #74LS02		
05	1004	3	12,14,	IC, Hex Inverter #74LS04		
			34			
06	1018	1	7	IC, Triple 3 Input NOR Gate #74LS27		
07	1020	1	35	IC, 8 Input NAND Gate #74LS30		
08	1037	4	13,20,	IC, Dual Type D Flip-Flop #74LS74		
			27,38			
09	1058	1	26	IC, Quad Tri-State Buffer #74LS126		
10	1062	1	37	IC, 3-To-8 Line Decoder #74LS138		
11	1071	3	15, 22,	IC, Quad 2-To-1 Line Multiplexer #74LS157		
			29			
12	1084	3	17,24,	IC, Quad D-Type Flip-Flop #74LS175		
			31			
13	1113	4	5,8,9,	IC, Quad 2 Input Exclusive-NOR Gate #74LS266		
			10			
14	1132	4	11,18,	IC, Hex Bus Driver #74LS367		
			25,32			
15	1202	1	19	IC, Retriggerable One-Shot #74122		
16	1502	4	16,23	IC, 4K RAM, 300ns, P2114-3 Static		
			30,36			
17	1804	1	33	IC, Voltage Reg, Positive 5V #7805		
18	2600	19	D1,	Diode, Small Signal #1N4148		
			D3-20			
19	2603	1	D2	LED, Red #RL4850		
20	3000	13	C	Capacitor, .033µf, 25V Ceramic Disc		
21	3300	1	C1	Capacitor, 100µf, 25V Electrolytic		
22	3500	3	C2,3,6	Capacitor, 2.2µf Tantalum		
				C-2		

DRN	DATE
CHKD	DATE
APPD D. Peggs	DATE 3/6/79
ECO	BY / APP
ECO	BY / APP
ECO	BY / APP
ECO	BY / APP

PRAGMATIC DESIGNS, INC.

MOUNTAIN VIEW, CALIF.

TITLE	PART NUMBER
DBM-1 DEBUG MEMORY BOARD	030000-05-A

COMPONENT CLASS	NEXT ASSY	SHEET
Product Family		2 OF 2

ITEM	PRAGMATIC PART NO	QTY	DRAWING REF	DESCRIPTION
23	4046A	1	R2	Resistor, 82 Ohm 1/4W, 5% CF
24	4068A	1	R7	Resistor, 680 Ohm 1/4W, 5% CF
25	4072A	3	R3,4,5	Resistor, 1K 1/4W, 5% CF
26	4087A	1	R6	Resistor, 4.3K 1/4W, 5% CF
27	4096A	1	R1	Resistor, 10K 1/4W, 5% CF
28	4700	2	RN1,2	Resistor Network #4310R-101-103
29	5004	4		Socket, 14 Pin Lowprofile IC T.I. #C84-14-02
30	5005	7		Socket, 16 Pin Lowprofile IC T.I. #C84-16-02
31	5006	4		Socket, 18 Pin Lowprofile IC T.I. #C84-18-02
32	5007	4		Socket, 24 Pin Lowprofile IC T.I. #C84-24-02
33	5011	1		Heat Sink #372B
34	5023	1		Strip, 1/9 x CA-S36RSP100-230(T)-090
35	5025	1	S2	Switch, 5 Bit DIP CTS #206-5, Alco #DBS-5, Amp #435640-3
36	0	1		Nut, 4-40
37	0	1		Screw, 4-40x1/4" Pan Head Cad Plated

Appendix D

If the Debug Memory is being multiplexed between several projects, using different ROM sizes, and it is not desirable to have to access the DIP switch, the switch SW2-D may be left CLOSED, and the 1K/2K select may be performed with Bit 15 of the trap address word.

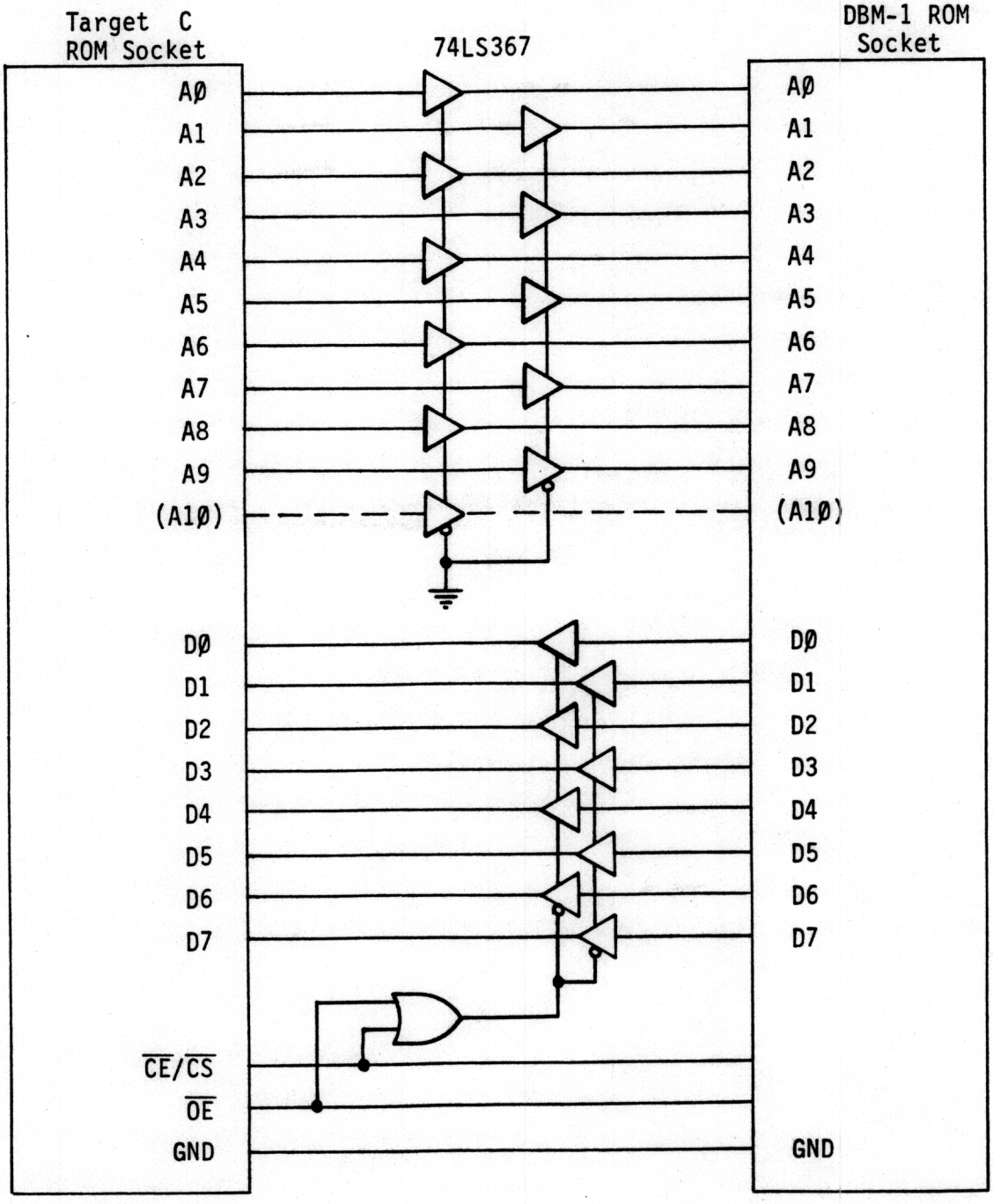
Bit 15	SW2-D	ROM Size
Ø	Closed	1K
1	Closed	2K
Ø	Open	2K
1	Open	2K

The software select of ROM size may be permanently defeated if desired by cutting the P.C. trace of IC28, Pin 8, and connecting IC28, Pin 8 to Pin 9. SW2-D then exclusively selects ROM size.

SOFTWARE SELECT OF SW-2D — 1K/2K ROM

Appendix E

If the use of very long cables (>3 feet) is desired, or if the user's target microcomputer ROM interface has little capacitive margin, the following buffer board may be used at the target microcomputer ROM socket.



BUFFER BOARD