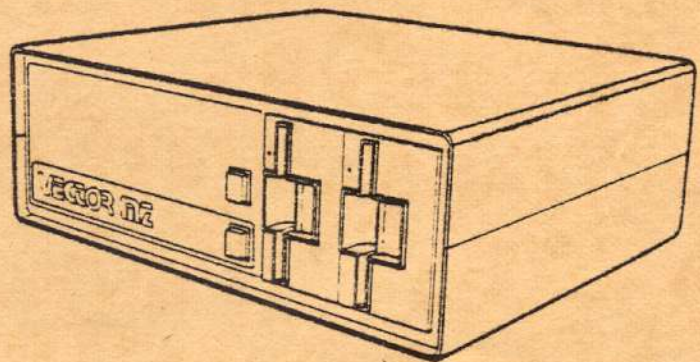
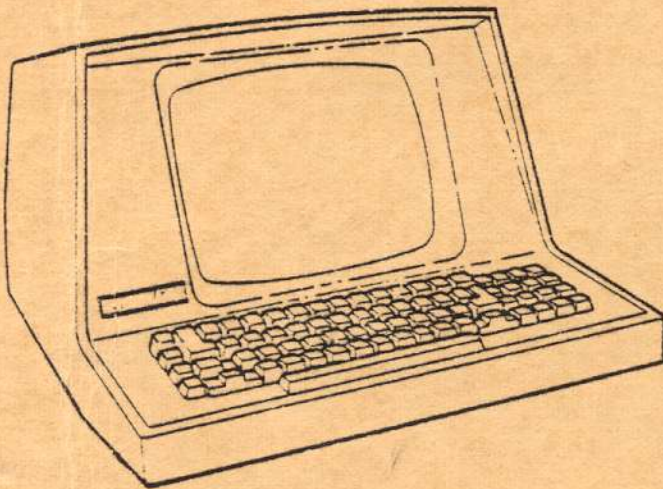


PROM RAM III

USER'S MANUAL



VECTOR
VECTOR GRAPHIC, INC.

PROM/RAM III BOARD

Revision 1

and

PROM PROGRAMMING PROGRAM

Revision 1

USERS MANUAL

Revision A

July 16, 1979

Copyright 1979 by Vector Graphic Inc.
All rights reserved.

Disclaimer

Vector Graphic makes no representations or warranties with respect to the contents of this manual itself, even if the product it describes is covered by a warranty or repair agreement. Further, Vector Graphic reserves the right to revise this publication and to make changes from time to time in the content hereof without obligation of Vector Graphic to notify any person of such revision or changes, except when an agreement to the contrary exists.

Revision Numbers

The date and revision of each page herein appears at the bottom of each page. The revision letter such as A or B changes if the manual has been improved but the product itself has not been significantly modified. The date and revision on the Title Page corresponds to that of the page most recently revised. When the product itself is modified significantly, the product will get a new revision number, as shown on the manual's title page, and the manual will revert to revision A, as if it were treating a brand new product. THIS MANUAL SHOULD ONLY BE USED WITH THE PRODUCT(S) IDENTIFIED ON THE TITLE PAGE.

REPAIR AGREEMENT

The PROM RAM III Board sold hereunder is sold "as is", with all faults and without any warranty, either expressed or implied, including any implied warranty of fitness for intended use or merchantability. However, the above notwithstanding, VECTOR GRAPHIC, INC., will, for a period of ninety (90) days following delivery to customer, repair or replace any PROM RAM III Board that is found to contain defects in materials or workmanship, provided:

1. Such defect in material or workmanship existed at the time the PROM RAM III Board left the VECTOR GRAPHIC, INC., factory;
2. VECTOR GRAPHIC, INC., is given notice of the precise defect claimed within ten (10) days after its discovery;
3. The PROM RAM III Board is promptly returned to VECTOR GRAPHIC, INC., at customer's expense, for examination by VECTOR GRAPHIC, INC., to confirm the alleged defect, and for subsequent repair or replacement if found to be in order.

Repair, replacement or correction of any defects in material or workmanship which are discovered after expiration of the period set forth above will be performed by VECTOR GRAPHIC, INC., at Buyer's expense, provided the PROM RAM III Board is returned, also at Buyer's expense, to VECTOR GRAPHIC, INC., for such repair, replacement or correction. In performing any repair, replacement or correction after expiration of the period set forth above, Buyer will be charged in addition to the cost of parts the then-current VECTOR GRAPHIC, INC., repair rate. At the present time the applicable rate is \$35.00 for the first hour, and \$18.00 per hour for every hour of work required thereafter. Prior to commencing any repair, replacement or correction of defects in material or workmanship discovered after expiration of the period for no-cost-to-Buyer repairs, VECTOR GRAPHIC, INC., will submit to Buyer a written estimate of the expected charges, and VECTOR GRAPHIC, INC., will not commence repair until such time as the written estimate of charges has been returned by Buyer to VECTOR GRAPHIC, INC., signed by duly authorized representative authorizing VECTOR GRAPHIC, INC., to commence with the repair work involved. VECTOR GRAPHIC, INC., shall have no obligation to repair, replace or correct any PROM RAM III Board until the written estimate has been returned with approval to proceed, and VECTOR GRAPHIC, INC., may at its option also require prepayment of the estimated repair charges prior to commencing work.

Repair Agreement void if the enclosed card is not returned to VECTOR GRAPHIC, INC. within ten (10) days of end consumer purchase.

TABLE OF CONTENTS

<u>Section</u>	<u>Page</u>
Repair Agreement	
Table of Contents	
I. Introduction	
1.1 Specifications.....	1-1
1.2 Description of the PROM/RAM III Board.....	1-3
II. Users Guide	
2.1 PROM Selection and Use.....	2-1
2.2 RAM.....	2-1
2.3 Block A and Block B - General.....	2-2
2.4 Block A.....	2-3
2.5 Block B.....	2-3
Figures 2 & 3 - Examples of Block B Configurations.....	2-5
2.6 Block Select Addressing.....	2-6
2.7 PROM/Scratchpad Memory Invert.....	2-6
2.8 RAM Memory Address Select in Block B.....	2-7
2.9 Disable 3K of Address Space in Block B.....	2-7
2.10 Power-on/Reset Jump - Description.....	2-8
2.11 Use PRESET or POC for Power-on/Reset Jump.....	2-8
2.12 Phantom Generated if Power-on/Reset.....	2-9
2.13 Jump to PROM/RAM III Board if Power-on/Reset.....	2-9
2.14 Block Swap.....	2-10
2.15 Disable Power-on/Reset Response.....	2-10
2.16 MWRITE.....	2-10
2.17 Wait State Generation.....	2-11
Tables 1, 2, and 3.....	2-12
2.18 Programming a PROM - Normal Procedure.....	2-13
2.19 Writing a PROM Programming Program.....	2-16
2.20 Re-assembling the PROM Programming Program.....	2-17
2.21 PROM Programming Program Listing.....	2-18 - 2-24
III. Theory of Operation	
3.1 Addressing.....	3-1
3.2 Data Input/Output.....	3-2
3.3 Control Signals.....	3-2
3.4 PROM Programming.....	3-3
3.4 Power Supplies.....	3-4

IV. Schematics

Board Layout.....	4-1
Schematic Errata.....	4-2
Schematic.....	4-3

I. INTRODUCTION

1.1 SPECIFICATIONS

Bus Compatibility	S-100
Memory Capacity	RAM: 1K, included with the board PROM: Sockets for 12 PROMs.
PROM Programming	Can program 2708 or 2704 EPROMs
PROM Programming Program	Listing included in manual Executable version on MDOS System Diskettes 8.4 and later.
PROMs Included with Board	NONE
Memory Speed	RAM: 300 ns. PROM: User selected (450 ns. typ)
Memory Types	RAM: 2114 static PROM: 2708 (1K each) or 2704 (1/2K each)
Board Addressing	Two blocks (A and B) are separately addressed Block A has 8 PROM sockets Block B has 4 PROM sockets and 1K RAM
Addressing Options (jumper)	Base address of the two 8K blocks Block B PROM at top or bottom of block Address of 1K RAM within remaining 4K Disable unused 3K, for use by other boards
Standard Addressing	Block A: disabled Block B base address: C000H Block B PROMs: C000H - CFFFH Block B RAM: DC00H - DFFFH Block B disabled 3K: D000H - DBFFH
Standard Location of Systems Monitor PROM	C000H

(continued on back)

Power-on/Reset Jump	PRESET or POC causes jump to board
Power-on/Reset Jump Options (jumper)	Use PRESET or POC Jump to first instruction of Block A or B. Disable phantom generation Disable jump to on-board memory
Standard Power-on/Reset Jumpers	POC is used Jump to beginning of Block B Phantom and jump to on-board both enabled
MWRITE	Jumper option to generate MWRITE on board Standard: option not enabled
Wait state generation	Jumper option to generate one wait state each time board is addressed Standard: option not enabled
Bus load	1 TTL load on all inputs
Card extractors	Standard
Power	+8Vdc @ 450 mA (Typ) +18Vdc @ (depends on quantity of PROM) -18Vdc @ (depends on quantity of PROM)

1.2 DESCRIPTION OF THE PROM RAM III BOARD

Vector Graphic's PROM RAM III Board is a versatile, S-100 bus compatible, high density memory board combining the memory technologies of erasable programmable read only memories (EPROMs) and high speed random access memory (RAM). Of unique value, one of the PROM sockets on the board can be used to program a 2708 or 2704 EPROM, enabling any owner to create PROM-based software for use on this board or in any other microprocessor device. 1K of RAM is provided on the board, but no PROMs are included with purchase. The software which is used to program PROMs is provided as a listing in this manual, and is included on disk with all Vector Graphic systems shipped with this board.

By combining the use of MSI decoding logic and unique addressing features, a wide range of applications requirements may be met by this memory board. The addressing flexibility is as follows. The board offers two independently addressable 8K blocks of memory (A and B). You use jumpers to specify the two separate 8K addressing spaces assigned to these blocks. Block A can be used for up to 8K of PROM. Block B contains 1K of on-board RAM plus up to 4K of PROM.

For block B, you use jumpers to specify whether the PROM is at the top or the bottom of the 8K allocation, and then, within the remaining 4K, where the 1K of RAM is addressed. Once this is done, there are also jumper options for DISABLING some or all of the remaining 3K of addressing space allocated to block B, so that other boards in the system can use those addresses.

The addressing spaces are fully utilized if 2708 1K PROMs are used. If 2704 1/2K PROMs are used, then every other 1/2K of PROM allocation will be used, with 1/2K gaps between. Other features offered by the board are: jump on power-on or reset to on-board memory, with phantom generated to temporarily disable other memory boards, and a jumper option to use PRESET instead of POC to cause this jump; jumper option for on-board generation of the S-100 MWRITE signal; and a jumper option to generate a one-cycle wait-state each time the board is addressed.

Full buffering of all inputs and outputs is provided to minimize loading of the system S-100 bus to at most one TTL load. On-board power regulation and filtering is provided using IC regulators and heat sinks for power dissipation. Careful attention to good design practice and an awareness of the need for flexibility has resulted in a reliable board useful in a wide variety of systems and applications.

II. USERS GUIDE

This Users Guide begins with a description of the amount and kind of PROM which can be used on this board, followed by a description of the RAM included with the board, then a detailed description of the various options you have for addressing the PROMs and the RAM. Read it before attempting to re-jumper the board addressing. Following this section are a description of each of the jumper options possible on the board, including addressing options, power-on/reset jump, MWRITE input, and wait state generation. The diagrams of jumper pads show each of the pads as it is pre-jumpered at the factory. The guide ends with instructions for operating the PROM programming software provided with the board, as well as instructions for writing your own if desired. The listing of the program is provided.

2.1 PROM SELECTION AND USE

A maximum of 12K bytes (where K = 1024) of 2708 type PROMs may be installed in available sockets on the board. NO PROMS ARE INCLUDED WITH PURCHASE OF THE BOARD ALONE. Jumpers are used to determine where the PROMs are addressed.

The following discussion assumes that 2708 type PROMs (having 1K of 8-bit bytes each) are used. If 2704 PROMs (having 1/2K bytes each) are used, the issues are the same; the only difference is that wherever a 2704 PROM is used, there will be 1/2K bytes of PROM accessible by the system, followed immediately by a 1/2K gap which will not contain any memory at all.

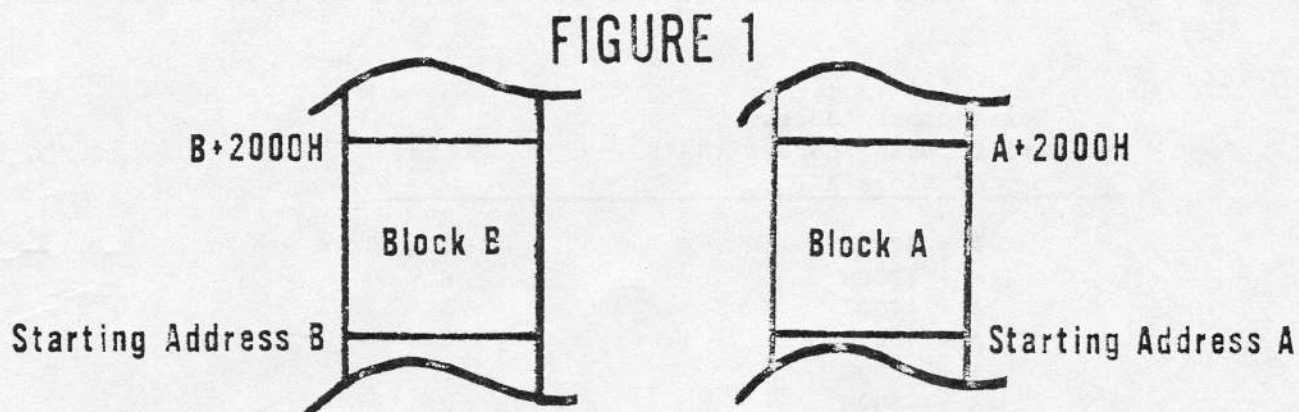
The numbers 2708 and 2704 are Intel generic part numbers. Many other manufacturers make equivalents, with 2708 or 2704 as part of their proprietary part number. All 2708 or 2704 pin for pin equivalents can be used on this board.

2.2 RAM

In addition to the PROM sockets, there is 1K of static RAM on the board, which IS included with purchase of the board alone. Jumpers are used to determine where this 1K of RAM is addressed.

2.3 BLOCK A AND BLOCK B - GENERAL

To begin specifying the addresses for the memory, there are two separately addressable blocks of memory space available on the board, called blocks A and B. Jumpers are used to specify what the base address is for each of these two blocks, within a 64K total memory space. Alternately, one (or both) blocks can be disabled completely. Jumper area F is normally used to specify the base address of (or disable) block A and jumper area E is normally used to specify the base address of (or disable) block B. If a block is not disabled, then that block will occupy exactly 8K bytes of memory, beginning at its base address. This is true for both blocks, as shown in Figure 1.



Note that both blocks together occupy 16K of memory. However, there are only 12 sockets for PROMs, and only 1K of RAM on the board, totalling 13K. What happens if the processor addresses memory in the remaining 3K portion? This memory space is NOT necessarily empty. A set of jumpers is provided which in effect specify that the unused 3K, within the 16K, is not on the PROM RAM III board at all, and therefore may be used on other boards.

It must be emphasized that except for the 3K specified as unused by jumper, the addresses assigned to the board for blocks A and B cannot be used by any other board, even if some of the PROM sockets are left empty. However, remember that you may choose not to use one (or both) of the blocks at all, by disabling it completely in jumper areas E and F. If you do this, then the corresponding memory space CAN be assigned to another board, and no space is wasted.

If the jumpers in area G are switched from the way the board is normally shipped, then the base address of block A will be controlled by jumper area E and the base address of block B will be controlled by jumper area F, instead of the other way around. If this is done, then the address which is accessed for power-on jump will also be switched, becoming the first address in block A instead of the first address in block B. This is the purpose for using this option. (See Section 2.14) For simplicity of language, the Users Guide is written assuming that jumper area G is left as manufactured.

2.4 BLOCK A

Block A refers to the 8 PROM sockets at the top of the board (labeled 0 through 7). Insert PROMs which you want in block A into these sockets. Socket 0 corresponds to the 1K block beginning at the base address of block A. Socket 1 corresponds to the next 1K and so on, as shown in the following table:

Hexadecimal Address Relative to Base Address ("A") of Block A	Socket
A + 1C00H	7
A + 1800H	6
A + 1400H	5
A + 1000H	4
A + C00H	3
A + 800H	2
A + 400H	1
A	0

Jumper area F is normally used to determine the base address of block A, or to disable block A. When the board is sold, jumper area F is pre-wired to disable block A. No particular base address is thus specified until you install the jumpers.

2.5 BLOCK B

Block B includes the lower four PROM sockets on the board, labeled 8 through 11. The other 4K in block B is filled with the 1K of RAM on the board, plus the 3K of address space which can be, at your discretion, returned for use by

other boards. The way you specify the address spaces within block B is as follows: First, you specify the base address of Block B using jumper area E (or you specify in area E that the block is disabled). If it is not disabled, then you use jumper area J to specify whether the 4K of PROM occupies the top or the bottom 4K of the block. These are the only two choices. The board is pre-jumpered so that the PROM occupies the lower 4K. Then, you specify using jumper area I which 1K within the other 4K is used for the on-board RAM. Lastly, you specify using jumper area H whether one of more of the last three 1K blocks is to be returned for use by other boards. (Normally you specify that all three of them are returned.)

Two typical configurations of Block B are shown in figures 2 and 3. Figure 2 is the standard - the one for which the board is pre-wired. Since in the pre-wired version, block B begins at C000H, Figure 2 shows that the standard address for scratch-pad RAM is DC00H, and the standard address for the System's Monitor PROM is C000H. Figure 3 shows the result of putting the PROM in the upper 4K and specifying that the RAM occupy the second 1K portion.

FIGURE 3

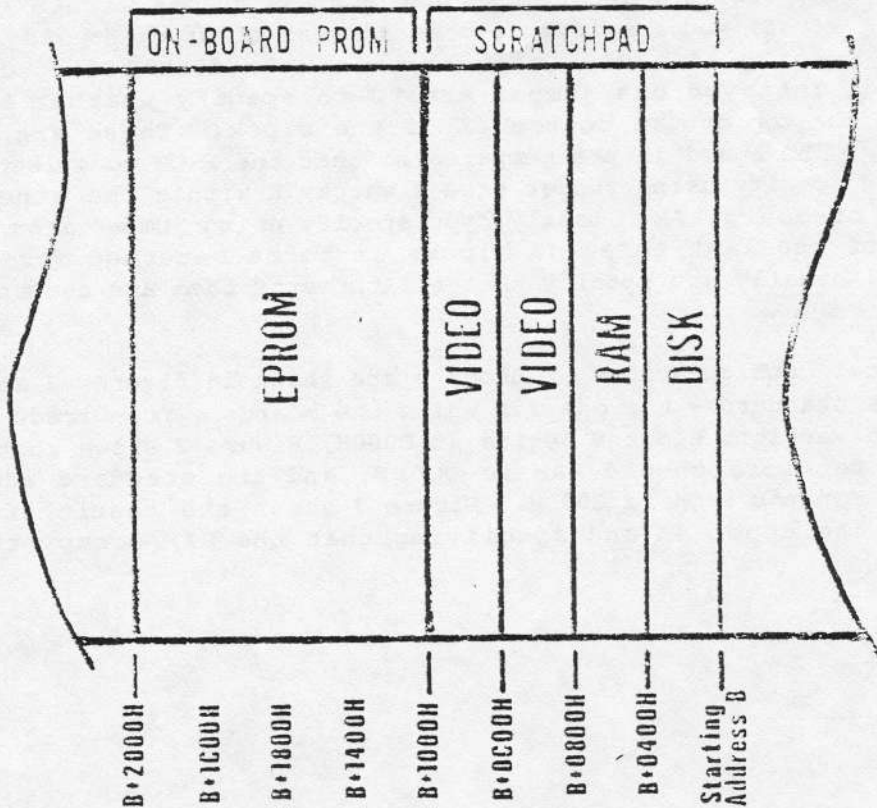
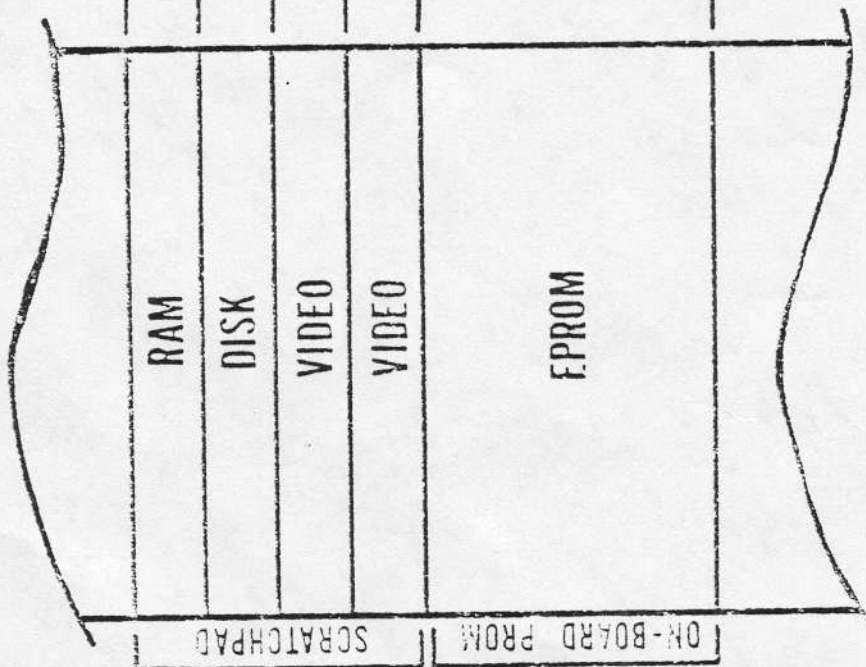
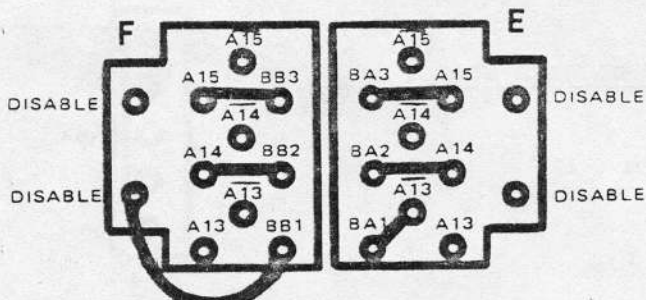


FIGURE 2



2.6 BLOCK SELECT ADDRESSING

Jumper areas: E & F



Jumper names: A13, A13, A14, A14, A15, A15 = address lines
 BA1, BA2, BA3 = block B address pads
 BB1, BB2, BB3 = block A address pads

NOTE: The second letter in the block B address pads is "A", while the second letter in the block A address pads is "B". This occurs because historically, the pads were named before it was decided to manufacture the board with the "block swap" jumpers in area G reversed.

Function: Address lines A13, A14, A15 form the most significant bits of the address from the CPU. These three bits can select any of 8 possible 8K blocks of memory in a 64K memory space. See table 1.

Options: Table 2 tells you what jumpers to connect to specify any particular 8K block starting address.

2.7 PROM/SCRATCHPAD MEMORY INVERT

Jumper area: J

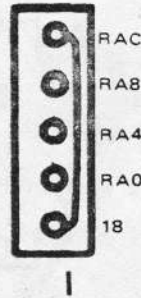


Function: The pre-wired connection specifies that the low order 4k bytes of block B consists of PROM. This jumper area is used to reverse this, putting the PROM at the high end of block B.

Options: If the PROM is to occupy the high order addresses of this block cut the jumper from 6 to 7 and tie 6 to 8.

2.8 RAM MEMORY ADDRESS SELECT IN BLOCK B

Jumper area: I

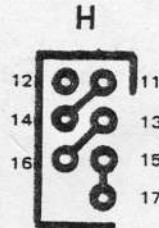


Function: These jumpers allow the user to selectively determine where the RAM addresses are to be located. With the board jumpered as manufactured, the 1K of RAM occupies the top-most 1K of addresses of the 4K scratchpad memory block.

Options: If you wish to alter the factory supplied connections, the following procedure is recommended: Cut the jumper from 18 to RAC. Then, determine the desired address for the 1K RAM from Table 3 and connect a jumper as specified. The third part of Table 3 is not relevant to this jumper area.

2.9 DISABLE 3K OF ADDRESS SPACE IN BLOCK B

Jumper area: H



Function: These jumpers allow the user to selectively determine which 3 of 4 1K blocks of memory are returned for use by other boards. These jumpers are selected in conjunction with the RAM memory address jumper in area I, so that together, all 4K of the non-PROM (scratchpad) address space in block B are accounted for. The factory supplied connections complement the factory supplied RAM address jumper, so that the bottom 3K of the scratchpad memory is allocated for use by other boards.

Options: If it is desired to alter the factory supplied connections, the following procedure is recommended: Verify the RAM memory address selected previously. Then, refer to Table 3 to find the RAM address selected, and connect jumpers as specified in the third part of the table.

2.10 POWER-ON/RESET JUMP - DESCRIPTION

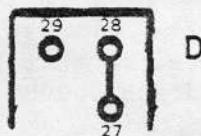
A power on/reset jump feature is also provided on this board. When the \overline{POC} or \overline{PRESET} (your choice of which, by jumper selection) line is low, the instruction stored in the first address of block A or B (determined by the jumper in area G, as explained below) will be executed by the CPU, and a "phantom" signal will be issued by the board on bus line 67 which disables other system memory boards.

After this initial instruction execution, the other memory boards will be re-enabled. However, if the instruction is a jump to the next instruction in the same block, then control will have been effectively transferred to that block on the PROM/RAM III board. Therefore, the second instruction should be the beginning of a system initialization routine followed by a systems executive. This is always the case in standard Vector Graphic computers.

Two additional jumper areas are provided, one to disconnect the phantom signal if it is not desired, and the other to disconnect the jump to the on-board PROM if this is not desired. These options give you maximum control over use of the board.

2.11 USE PRESET OR POC FOR POWER-ON/RESET JUMP

Jumper area: D



Function: In the factory version of the board, the \overline{POC} signal is connected to the power-on/reset jump circuitry on the board. This is appropriate for standard Vector Graphic computers, because in these systems, both the RESET switch on the front panel and the initial power-on condition cause an active low pulse on the POC line, via circuitry on the E80 board. If the CPU board used in your system does not have this feature, the \overline{PRESET} signal can be connected to the power-on/reset circuitry by changing the jumper area D.

Options: To connect \overline{PRESET} to the power-on/reset circuitry, cut the trace between 27 and 28 and tie 28 to 29.

2.12 PHANTOM GENERATED IF POWER-ON/RESET

Jumper area: C



C

Function: When 1 and 2 are tied together, the phantom signal is generated whenever a POC or PRESET signal is received. Phantom disables other system memory boards. The Z80 (and 8080) processor chip immediately executes the instruction at 0000H when the POC or PRESET signal appears on the bus, assuming the CPU board is so designed. With the other memory boards in the system disabled, the PROM/RAM III Board is free to supply the instruction for address 0000H.

Options: To disable the generation of the phantom signal, cut the jumper from 1 to 2.

2.13 JUMP TO PROM/RAM III BOARD IF POWER-ON/RESET

Jumper area: A



A

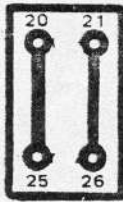
Function: When the POC or PRESET signal is received, a jumper in area A causes the board to respond to the address 0000H from the CPU. At your option, you may disable this feature, so that the PROM/RAM III board is NOT the board which responds to the address 0000H.

Options: To cause the board NOT to respond to address 0000H when POC or PRESET is received, cut the jumper from 3 to 4 and tie 4 to 5.

2.14 BLOCK SWAP

Jumper area: G

G



Function: With the board as manufactured, jumper area E is used to address block B, and jumper area F is used to address block A. Furthermore, if the power-on/reset jump feature is used, the jump will take place to the first address in block B.

Options: If you want to jump to block A instead, cut the jumpers from 20 to 21 and 25 to 26; tie 20 to 25 and 21 to 26. This change will also reverse the use of area E and F, so that area E is used to address block A, and area F is used to address block B.

2.15 DISABLE POWER-ON/RESET RESPONSE ✓

To disable the power-on/reset response of the PROM/RAM III board entirely, disable both the generation of phantom and the jump to PROM/RAM III board. See Sections 2.12 and 2.13.

2.16 MWRITE INPUT

Jumper area: B



B

Function: If this board is installed in a system without a front panel, or other source of MWRITE, an MWRITE signal can be generated on board both for use on board and for feeding back to the bus as a fully buffered S-100 signal. This is not needed in Vector Graphic systems shipped after April 9, 1979, because the Z-80 boards in these systems now generate MWRITE.

Options: If the board is installed in a system without a source of MWRITE, add a jumper from 9 to 10.

2.17 WAIT STATE GENERATION

Jumper area: K



Function: The PRDY signal may be jumpered to the WAIT input in order to create one wait state each time the board is addressed. This is necessary when using memory slower than about 300 ns. in a 4 MHz (Z-80) system. PRDY is not connected to WAIT on the PROM/RAM III board as manufactured, because the Vector Graphic Z-80 board used in Vector Graphic systems generates the wait-state. You would want to generate the wait-state on the PROM/RAM III board if you are using memory faster than 300 ns. on other memory boards in the system, allowing you to disable the wait state that is built into the Vector Graphic Z-80 board (and some other manufacturers' Z-80 boards) yet continue to use a wait-state for the slower memory on the PROM RAM/III board.

For some Z-80 based CPU boards the WAIT output is not synchronized properly. If the WAIT is jumpered to the PRDY signal when such a Z-80 board is used, a possible oscillatory condition can arise on the PRDY and WAIT lines. Therefore, caution must be exercised in how this jumper is utilized. The Vector Graphic Z-80 board has a properly synchronized WAIT, so that with this Z-80 board, PRDY may be safely tied to WAIT, insuring reliable memory operation at high speeds.

Options: To tie PRDY to WAIT, jumper 22 to 23.

TABLE 1

A15	A14	A13	8K BLOCK (A or B) STARTING ADDRESS
0	0	0	0000H = 0000D
0	0	1	2000H = 8192D
0	1	0	4000H = 16384D
0	1	1	6000H = 24576D
1	0	0	8000H = 32768D
1	0	1	A000H = 40960D
1	1	0	C000H = 49152D
1	1	1	E000H = 57344D

H = Hexidecimal D = Decimal

TABLE 2

DESIRED 8K BLOCK STARTING ADDRESS	CONNECT		
	Bx1 to:	Bx2 to:	Bx3 to:
0000H	A13	A14	A15
2000H	A13	A14	A15
4000H	A13	A14	A15
6000H	A13	A14	A15
8000H	A13	A14	A15
A000H	A13	A14	A15
C000H	A13	A14	A15
E000H	A13	A14	A15

x = Block A or B

If any Bx1, Bx2, Bx3 is tied to disable, that block of memory is disabled.

TABLE 3

ADDRESS OF 1K RAM WITH RESPECT TO THE STARTING ADDRESS OF THE 4K BLOCK	JUMPERS FOR RAM ADDRESS WITHIN 4K BLOCK	JUMPERS FOR BUS DISABLE
0000H	18 to RA0	15 to 16, 13 to 14, 11 to 12
0400H	18 to RA4	15 to 17, 13 to 14, 11 to 12
0800H	18 to RA8	15 to 17, 13 to 16, 11 to 12
0C00H	18 to RA12	11 to 14, 13 to 16, 15 to 17

2.18 PROGRAMMING A PROM

This board is accompanied by a program which allows you to program any 2704 or 2708 type EPROM. The listing of this program is found in Section 2.21, below. This same program is found on MDOS System Diskettes, version 8.4 and later, which accompany all Vector Graphic computers that are equipped with PROM/RAM III boards. The program exists on the disk as an immediately executable utility. The program is written in machine language and is not dependent on any operating system (except that it uses the Extended Systems Monitor in Vector Graphic systems for console I/O.) The utility (called "PROM") runs beginning at address 2B00 Hex and takes up less than 1K. If you want to run it elsewhere, or want to revise it, reassemble it as described in Section 2.20.

If you use an operating system other than MDOS, but you have the MDOS diskette, simply load the program under MDOS and copy it to a disk using the other system. To load it, just type PROM (return) followed by control-C, under MDOS. If you do not have the MDOS diskette, enter the program from the listing. Once it is loaded in memory, you can execute it from any executive, including the Extended Systems Monitor executive. The following explains the use of this program. If you are not using MDOS, then substitute the MDOS commands given here by those that are relevant to you.

1. Make sure the computer power is OFF. Wait at least five seconds before pulling out any circuit boards.
2. Unscrew and remove the cover of the computer.
3. Find the PROM/RAM III board. If you cannot easily reach PROM socket 11 with your hand, pull the board out.
4. Insert the PROM you wish to program in socket 11. This is the right-hand socket in the second row. Make sure to insert the PROM with its notch pointed to the top of the board. The PROM used MUST have been erased using ultraviolet erasing techniques, unless it is new. The computer cannot simply write over any previously used PROM, because programming involves turning logical 1's into 0's, but cannot go the other way. Erasing fills the PROM with 1's, like a new PROM.
5. Return the board to a slot which allows you to reach socket 11 without pulling the board out in the future, if possible.
6. Turn computer power ON.
7. If the system is not in the Extended Systems Monitor executive (indicated by the Monitor prompt *) then depress RESET on the computer front panel.
8. Mount the MDOS system diskette in drive 0 (the right-hand drive.) Then,

depress B on the keyboard. MDOS will take control, as indicated by the MDOS prompt >.

9. Load the object code to be stored on PROM into a free area of memory. Alternately, you may generate the desired code by assembling or compiling a higher level program.
10. Following the MDOS prompt >, type PROM (return). The PROM programming program will take control.
11. In response to the question "Starting from:", type the address in Hex of the first location you wish to program, within the block of memory assigned to PROM socket 11. Then press the RETURN key. Usually this starting address will be CC00. If programming less than the entire PROM, it can be any address between CC00 and CFF0. It must be an address ending in 0. If not, the machine will report "bad boundary address" and give you another chance. Letters must be in upper case. Do not tack on an H or any other symbol.
12. In response to the question "terminating at:", type the address in Hex of the last location you wish to program, within the block of memory assigned to PROM socket 11. Then press the RETURN key. Usually this terminating address will be CFFF for 2708 PROMs and CDFF for 2704 PROMs. If programming less than the entire PROM, it can be any address between CCOF and CFFF. It must be an address ending in F, and must be greater than the starting address. If not ending in F, the machine will report "bad boundary address" and then give you another chance.

As with the starting address, if you enter an address outside the range CCOF to CFFF, the program will not accept it, and will report "Out of range" and then give you another chance. Therefore, if the addressing jumpers determining the location of socket 11 have been modified, you must modify the PROM programming program to accept other addresses.

After entering the terminating address, the computer will either continue with the next question, or it will report "specified portion of PROM is not erased." This message means either that the terminating address is less than the starting address, or that the PROM is not new and was not properly erased. This message is strictly a warning, because in certain rare cases you may want to write over an unerased PROM. After the message, the system will continue with the next question. If you want to start over to correct your mistake, instead of continuing, then depress the ESC key. This takes the system back to the Monitor. To get back to MDOS from the Monitor, depress J. Then begin the program again at step 10, above.

13. In response to the question "Source address:", type the starting address in memory of the material you want to store on PROM. This can be any address in memory. Then press the RETURN key.
14. Slide the "programming" switch at the upper right-hand corner of the PROM/RAM III board to the LEFT.
15. Now, press the RETURN key again. This will begin programming of the PROM. The computer must pass through the range of target addresses 256 times. A message will appear on the screen showing which pass the machine is currently on.
16. When programming is complete, one of two events will take place. If the computer detects no errors in comparing the programmed PROM without the original code, then the system will return to the MDOS executive or whichever other executive was used to call the programming program. If an error is discovered however, the screen will show the first address within the PROM at which a verification error was found. For example, if you forgot to slide the programming switch to the left, then, since the PROM will not have been programmed at all, the first address will be incorrect, so that the system will report an error at address CC00, or whatever was the starting address you had specified. After reporting the error, the system will return to the MDOS executive, so that you can start over.
17. When programming is complete, immediately slide the programming switch on the PROM/RAM III board to the RIGHT. Do not postpone this.
18. Remove the programmed PROM from socket 11. Alternately, you may use the PROM without removing it. For example, you may run a checksum of the PROM using the Extended System Monitor's Q command. To do this, depress control-Q or whichever other command your system uses to get to the Monitor executive. Then type Q CC00 CFFF. (The spaces will occur automatically.) The checksum, will appear immediately. (If PROM socket 11 has been readdressed, then use the appropriate addresses.) To return to MDOS from the Monitor, depress J.

depress B on the keyboard. MDOS will take control, as indicated by the MDOS prompt >.

9. Load the object code to be stored on PROM into a free area of memory. Alternately, you may generate the desired code by assembling or compiling a higher level program.
10. Following the MDOS prompt >, type PROM (return). The PROM programming program will take control.
11. In response to the question "Starting from:", type the address in Hex of the first location you wish to program, within the block of memory assigned to PROM socket 11. Then press the RETURN key. Usually this starting address will be CC00. If programming less than the entire PROM, it can be any address between CC00 and CFF0. It must be an address ending in 0. If not, the machine will report "bad boundary address" and give you another chance. Letters must be in upper case. Do not tack on an H or any other symbol.

CC00 is the starting address of PROM socket 11 if the board is left in factory-supplied format. If you enter an address outside the range CC00 to CFF0, the program will not accept it, and will report "out of range" and then give you another chance. If the addressing jumpers determining the location of socket 11 have been modified, you must modify the program to accept other addresses.

12. In response to the question "terminating at:", type the address in Hex of the last location you wish to program, within the block of memory assigned to PROM socket 11. Then press the RETURN key. Usually this terminating address will be CFFF for 2708 PROMs and CDFE for 2704 PROMs. If programming less than the entire PROM, it can be any address between CCOF and CFFF. It must be an address ending in F, and must be greater than the starting address. If not ending in F, the machine will report "bad boundary address" and then give you another chance.

As with the starting address, if you enter an address outside the range CCOF to CFFF, the program will not accept it, and will report "Out of range" and then give you another chance. Therefore, if the addressing jumpers determining the location of socket 11 have been modified, you must modify the PROM programming program to accept other addresses.

After entering the terminating address, the computer will either continue with the next question, or it will report "specified portion of PROM is not erased." This message means either that the terminating address is less than the starting address, or that the PROM is not new and was not properly erased. This message is strictly a warning, because in certain rare cases you may want to write over an unerased PROM. After the message, the system will continue with the next question. If you want to start over to correct your mistake, instead of continuing, then depress the ESC key. This takes the system back to the Monitor. To get back to MDOS from the Monitor, depress J. Then begin the program again at step 10, above.

13. In response to the question "Source address:", type the starting address in memory of the material you want to store on PROM. This can be any address in memory. Then press the RETURN key.
14. Slide the "programming" switch at the upper right-hand corner of the PROM/RAM III board to the LEFT.
15. Now, press the RETURN key again. This will begin programming of the PROM. The computer must pass through the range of target addresses 256 times. A message will appear on the screen showing which pass the machine is currently on.
16. When programming is complete, one of two events will take place. If the computer detects no errors in comparing the programmed PROM without the original code, then the system will return to the MDOS executive or whichever other executive was used to call the programming program. If an error is discovered however, the screen will show the first address within the PROM at which a verification error was found. For example, if you forgot to slide the programming switch to the left, then, since the PROM will not have been programmed at all, the first address will be incorrect, so that the system will report an error at address CC00, or whatever was the starting address you had specified. After reporting the error, the system will return to the MDOS executive, so that you can start over.
17. When programming is complete, immediately slide the programming switch on the PROM/RAM III board to the RIGHT. Do not postpone this.
18. Remove the programmed PROM from socket 11. Alternately, you may use the PROM without removing it. For example, you may run a checksum of the PROM using the Extended System Monitor's Q command. To do this, depress control-Q or whichever other command your system uses to get to the Monitor executive. Then type Q CC00 CFFF. (The spaces will occur automatically.) The checksum, will appear immediately. (If PROM socket 11 has been readdressed, then use the appropriate addresses.) To return to MDOS from the Monitor, depress J.

2.19 WRITING A PROM PROGRAMMING PROGRAM

Although the PROM/RAM III board is supplied with a program for programming PROMs, this section explains the principles behind the program, for those wishing to write their own. The supplied program is listed in Section 2.21, for reference.

To program a 2708 or 2704 type EPROM, simply write the desired data to the locations assigned to PROM socket 11. The board hardware automatically interprets any writing of data to PROM socket 11 as an intent to program it. You do not have to program an entire PROM. You may program any part of it, down to blocks as short as 16 adjacent locations. Normally, you will program all 1K of a 2708 or all 512 bytes of a 2704. Write to all desired addresses in sequence. After finishing one such cycle, repeat it, using exactly the same data. You must repeat this cycle 256 times. In other words, you must write to each address 256 times, with a substantial delay between each time you write to each address. This delay is produced by the time taken to cycle through all the addresses, which is sufficiently long if 16 or more locations are programmed.

A good program has a comparison of the source and destination data, after programming the PROM is complete.

If your system has a dynamic memory board in it (such as all Vector Graphic systems shipped since about March 1, 1979), then there MUST be a delay loop after each byte is written to the PROM, so that the processor can refresh memory. The delay loop must execute at least 128 instructions each time it is accessed. You will find an example of this at the top of the fourth page in the listing in Section 2.21.

Before executing a programming procedure, you must slide the programming switch on the upper right-hand corner of the board TO THE LEFT. Then, put the PROM to be programmed into socket 11, which is the socket furthest to the right in the second row. After successfully programming it, slide the switch BACK. If you do not, you might accidently erase a PROM sitting in socket 11.

A PROM which you want to program must be either new or newly erased using the standard ultraviolet technique.

2.20 RE-ASSEMBLING THE PROM PROGRAMMING PROGRAM

The source code for the program is listed in Section 2.21 below. Enter the program using the MDOS editor LINEEDIT. You can assemble it wherever you like, although BC00 is not suggested because M.BASIC uses the very top of RAM for stack. The pre-assembled version on the diskette (under the name "PROM") is assembled to run at 2B00, at the beginning of the MDOS applications area. The program is less than 1K long.

You may modify PROM.S before you assemble it, by using the MDOS editor LINEEDIT. One modification which may be required are the addresses in the last two lines of PROM.S. You will have to change these if you change the jumpers on the PROM/RAM III board which assign the address of the on-board RAM. After entering and modifying the program, SAVE it on diskette under the name PROM.S. (Type NAME "PROM.S" (return) followed by SAVE (return) while in LINEEDIT.

To assemble PROM.S, use the ZSM assembler. With a diskette having both ZSM and PROM.S mounted in drive 0, and with MDOS in control, type ZSM "PROM.S" "PROM2" "E" (return). The assembler will ask where you want to run the program. Enter the address, for example 2B00H, that you want it to run at. Note that if the first character is a letter, it must be preceded by a 0 (zero), and the address must be followed by an H. The above ZSM statement will cause the program to be assembled with only errors printed. For other options possible with ZSM, see Section 4.5 of the User's Guide to Vector Graphic Systems Using MDOS.

After the assembly is complete, type TYPE "PROM2" 18 (return). This type will allow you to execute the program simply by typing PROM2 (return) while under MDOS.

If you want to put the PROM programming program on a PROM, in order to have a permanent PROM programming capability, first choose the memory location you want to give to this PROM, say E000, which is available on the PROM/RAM III board. Use this address when asked by the assembler where you want it to run at. Since there is no RAM at this address, you will have to load the assembled code into a different location before you can put it on a PROM. To do this change the type to 00 rather than 18, by typing TYPE "PROM2" 00 (return), after the assembly is complete. This will allow you to type LOAD "PROM2" 2B00 (return) after the MDOS prompt >, thus loading the code at RAM address 2B00, ready to be saved on a PROM.

2.21 PROM PROGRAMMING PROGRAM LISTING

```

Addr 31 82 83 84 E Label      Opco  Operand
0000          *****
0000          *
0000          * Prom Programming Program *
0000          *   Version 1
0000          *   for the Prom/Ram III *
0000          *
0000          *   by Lance Lewis,
0000          *   Vector Graphic Inc.
0000          *   20-July-79
0000          *
0000          *****
0000          *
0000          *
0000          * System equates
0000          *
0000 0003 = INPUT      EQU  00003H      ;character input (0003 on pre 3.0 monitors)
0000 0008 = OUT        EQU  00008H      ;video driver (0008 on pre 3.0 monitors)
0000          *
0000          * Definitions and Constants
0000          *
0000 0000 = PROM       EQU  00000H      ;prom address
0000 00FF = BLANK     EQU  00FFH       ;erased byte of prom
0000 000A = CRLF      EQU  000AH       ;carriage return linefeed
0000 000A = LF        EQU  0AH         ;linefeed
0000 000D = CR        EQU  0DH         ;carriage return
0000 0080 = MSB       EQU  80H         ;most significant bit
0000          *
0000 2800 = ORIG      REQ  'Program to run at?'
0000          ORG  ORIG          ;assemble here
2800          *
2800          * Here we go
2800          *
2800 85          PUSH  H          ;save HL
2801 85          PUSH  D          ;save DE
2802 85          PUSH  B          ;save BC
2803 85          PUSH  PSW        ;save AF
2804 21 00 00     LXI  H,0
2807 39          DAD  SP          ;HL=SP
2808 22 03 2E     SHLD STACK      ;store it
2808 31 00 00     LXI  SP,0000GH   ;reset stack pointer
280E          *
280E 00 43 20     CALL PRINT      ;send message
2811 00 0A        DD  CRLF
2813 20 20 20 20  DT  '      Vector Graphic'
2817 20 20 36 65
2818 63 74 6F 72
281F 20 47 72 61
2823 70 68 69 63
2827 00 0A        DD  CRLF          ;print CRLF
2829 20 20 50 72  DT  ' Prom Programming System'
2820 6F 60 20 50
2831 72 6F 67 72
2835 61 60 60 69
2839 6E 67 20 53
283D 79 73 74 65
2841 60

```

PROM/ROM III Board Users Manual

Addr	B1	B2	B3	B4	E	Label	Opcd	Operand
2B42	0D	0A					DD	CRLF
2B44						*		
2B44	0A						DB	LF ;down a line
2B45	20	20	50	72			DT	' Program prom'
2B49	6F	67	72	61				
2B4D	6D	20	70	72				
2B51	6F	60						
2B53	0D	8A					DD	CRLF+MSB ;end of message
2B55						*		
2B55	0D	43	20			STARTADRS	CALL	PRINT ;send message
2B58	20	20	53	74			DTH	' Starting from :'
2B5C	61	72	74	69				
2B60	6E	67	20	66				
2B64	72	6F	5D	2D				
2B68	8A							
2B69	0D	4F	2D				CALL	ADRS ;get start address
2B6C	0A	55	2B				JC	STARTADRS ;if invalid try again
2B6F	0D	43	2D				CALL	PRINT
2B72	0D	8A					DD	CRLF+MSB ;print CRLF
2B74	0D	24	2D				CALL	RANGERR ;check for error
2B77	0A	55	2B				JC	STARTADRS ;try again if error
2B7A	0D	96	2D				CALL	MOD ;check boundary
2B7D	0A	55	2B				JC	STARTADRS ;no good
2B80	EB						XCHG	;DE=start adrs
2B81						*		
2B81	0D	43	2D			ENDADRS	CALL	PRINT ;send message
2B84	20	20	54	65			DTH	' Terminating at:'
2B88	72	6D	69	6E				
2B8C	61	74	69	6E				
2B90	67	20	61	74				
2B94	8A							
2B95	0D	4F	2D				CALL	ADRS ;get end address
2B98	0A	81	2B				JC	ENDADRS ;if invalid try again
2B9B	0D	43	2D				CALL	PRINT
2B9E	0D	8A					DD	CRLF+MSB ;carriage return linefeed
2BA0	0D	24	2D				CALL	RANGERR ;check for range error
2BA3	0A	81	2B				JC	ENDADRS ;try again if error
2BA6	23						INX	H ;compensate
2BA7	0D	96	2D				CALL	MOD ;check boundary
2BAA	0A	81	2B				JC	ENDADRS ;no good
2BAD	44						MOV	B,H ;save end address
2BAE	4D						MOV	C,L ; in register pair BC
2BAF						*		
2BAF	62						MOV	H,D ;save start address
2BB0	5B						MOV	L,E ; in register pair HL
2BB1	1A					TFFS	LDAX	D ;get byte from prom
2BB2	FE	FF					CPI	BLANK ;is it clear
2BB4	02	B7	2D				JNZ	BADPROM ;print "bad prom"
2BB7	13						INX	D ;check next location
2BBB	0D	F6	2C				CALL	TEST ;end of area
2BBE	02	B1	2B				JNZ	TFFS ;more to come
2BBE	EB					RESTORE	XCHG	;restore registers
2BBF						*		
2BBF	0D	43	2D			SOURCEADRS	CALL	PRINT
2BC2	20	20	53	6F			DTH	' Source address:'
2BC6	75	72	63	65				

Addr	B1	B2	B3	B4	E	Label	Opcd	Operand
2B0A	20	61	64	64				
2B0E	72	65	73	73				
2B02	BA							
2B03	0D	4F	2D				CALL	ADRS ;get source address
2B06	0A	BF	2B				JC	SOURCEADRS ;if not valid try again
2B09						*		
2B09	0D	43	2D				CALL	PRINT ;send message
2B0C	0D	0A					DD	CRLF
2B0E						*		
2B0E	0D	0A					DD	LF ;format output
2B0D	20	20	54	75			DT	' Turn on the programming enable switch'
2B04	72	6E	20	6F				
2B08	6E	20	74	68				
2B0C	65	20	70	72				
2BF0	6F	67	72	61				
2BF4	6D	6D	69	6E				
2BF8	67	2D	65	6E				
2BFC	61	62	6C	65				
2C00	20	73	77	69				
2C04	74	63	68					
2C07	0D	0A					DD	CRLF
2C09	20	20	48	69			DT	' Hit return to continue?'
2C0D	74	2D	72	65				
2C11	74	75	72	6E				
2C15	20	74	6F	2D				
2C19	63	6F	6E	74				
2C1D	69	6E	75	65				
2C21	BF							
2C22						*		
2C22	0D	03	0D			STAT	CALL	INPUT ;check keyboard
2C25	0A	22	2C				JZ	STAT ;no character
2C28	FE	0D					CPI	CR ;is it a return
2C2A	02	22	2C				JNZ	STAT ;no try again
2C2D						*		
2C2D	0D	43	2D				CALL	PRINT
2C30	0D	0A					DD	CRLF
2C32	0A						DB	LF
2C33	20	20	50	72			DT	' Programming in progress'
2C37	6F	67	72	61				
2C3B	6D	6D	69	6E				
2C3F	67	2D	69	6E				
2C43	2D	70	72	6F				
2C47	67	72	65	73				
2C4B	73							
2C4C	0D	0A					DD	CRLF
2C4E	BA						DB	LF+MSB ;stop sending with linefeed
2C4F						*		
2C4F	AF						XRA	A ;zero
2C50	32	02	2E				STA	PASS ;pass counter
2C53						*		
2C53	E5					SAVE	PUSH	H ;save source address
2C54	05						PUSH	D ;save it
2C55						*		
2C55	7E					LOOP	MOV	A,M ;get byte from source
2C56	12						STAX	D ;program it to destination
2C57						*		

PROM/RAM III Board Users Manual

Addr	B1	B2	B3	B4	E	Label	OpCod	Operand	
2C57	3E	64					MVI	A,100	;delay for dynamic memory
2C59	3D					DELAY	DCR	A	;time up
2C5A	C2	59	2C				JNZ	DELAY	;keep stalling
2C5D						*			
2C5D	23						INX	H	
2C5E	13						INX	D	;advance pointers
2C5F	CD	F6	2C				CALL	TEST	;end of block
2C62	C2	55	2C				JNZ	LOOP	;no keep going
2C65						*			
2C65	21	02	2E				LXI	H,PASS	;point to pass counter
2C68	34						INR	M	;256 passes
2C69	F5						PUSH	PSW	;save Z flag
2C6A	C5						PUSH	B	;save end pointer
2C6B						*			
2C6B	CD	43	2D				CALL	PRINT	;send message
2C6E	0D						DB	CR	
2C6F	2D	2D	5D	61			DTH	' Pass '	
2C73	73	73	A0						
2C76	7E						MOV	A,M	;get pass number
2C77	0E	00					MVI	C,0	;clear number of digits
2C79	06	FF				LDIV	MVI	B,-1	;compensate for increment
2C7B	04					DIV	INR	B	;increment quotient
2C7C	D6	DA					SUI	10	;subtract 10 from dividend
2C7E	D2	7B	2C				JNC	DIV	;can more be subtracted
2C81	C6	3A					ADI	10+'0'	;adjust remainder 0 to 9 ASCII
2C83	F5						PUSH	PSW	;add to list of remainders
2C84	0C						INR	C	;one more digit
2C85	78						MOV	A,B	;prepare for next division
2C86	B7						ORA	A	;was quotient zero
2C87	C2	79	2C				JNZ	LDIV	;more to come
2C8A	F1					LOUT	POP	PSW	;get a remainder
2C8B	CD	38	CD				CALL	OUT	;print it
2C8E	0D						DCR	C	;out of digits
2C8F	C2	3A	2C				JNZ	LOUT	;no then keep printing
2C92						*			
2C92	C1						POP	B	;restore end
2C93	F1						POP	PSW	;restore Z flag
2C94	D1						POP	D	;restore start address
2C95	E1						POP	H	;restore HL
2C96	C2	53	2C				JNZ	SAVE	;more passes to come
2C99						*			
2C99	1A					VERIFY	LDAX	D	;get byte from prom
2C9A	BE						CMP	M	;is it the same
2C9B	C2	FC	2C				JNZ	VERIFYERR	;print error
2C9E	23						INX	H	
2C9F	13						INX	D	;advance pointers
2CA0	CD	F6	2C				CALL	TEST	;end of block
2CA3	C2	99	2C				JNZ	VERIFY	;still more to test
2CA6						*			
2CA6	CD	43	2D				CALL	PRINT	
2CA9	0D						DB	CR	
2CAA	2D	2D	4E	6F			DT	' No errors detected'	
2CAE	2D	65	72	72					
2CB2	6F	72	73	2D					
2CB5	64	65	74	65					
2CBA	63	74	65	64					

Addr	B1	B2	B3	B4	E	Label	Opnd	Operand
2CB8	00	3A					DD	CRLF+MSB
2CC0						*		
2CC0	CD	43	2D			END	CALL	PRINT
2CC3	20	20	54	75			DT	' Turn off the programming enable switch'
2CC7	72	6E	20	5F				
2CCB	66	66	20	74				
2CCF	68	65	20	70				
2CD3	72	6F	67	72				
2CD7	61	6D	6D	69				
2CD8	6E	67	20	65				
2CDF	6E	51	62	6C				
2CE3	65	20	73	77				
2CE7	69	74	63	68				
2CE8	00	3A					DD	CRLF+MSB
2CED						*		
2CED	2A	03	2E				LHLD	STACK ;retrieve SP
2CF0	F9						SPHL	;move it back
2CF1	F1						POP	PSW ;restore registers
2CF2	C1						POP	B
2CF3	D1						POP	D
2CF4	E1						POP	H
2CF5	C9						RET	;bye-bye
2CF6						*		
2CF6	78					TEST	MOV	A,B ;get end byte
2CF7	BA						CMP	D ;same as start
2CF8	CD						RNZ	;no then return
2CF9	79						MOV	A,C
2CFA	8B						CMP	E ;low half same
2CFB	C9						RET	;return with Z flag
2CFC						*		
2CFC	CD	43	2D			VERIFYERR	CALL	PRINT
2CFF	00						DB	CR
2D00	3F	20	76	65			DTH	'? verification error at '
2D04	72	69	66	69				
2D08	63	61	74	69				
2D0C	6F	6E	20	65				
2D10	72	72	6F	72				
2D14	20	61	74	A0				
2D18	EB						XCHG	
2D19	CD	E8	2D				CALL	HEX ;print hex address
2D1C	CD	43	2D				CALL	PRINT
2D1F	00	3A					DD	CRLF+MSB
2D21	C3	CD	2C				JMP	END
2D24						*		
2D24	7C					RANGERR	MOV	A,H ;get high address
2D25	FE	CC					CPI	PROM/256 ;valid address
2D27	0A	2E	2D				JC	RANGEMES ;no print message
2D2A	FE	00					CPI	PROM/256+4 ;valid address
2D2C	3F						CMC	;compensate
2D2D	00						RNC	;return with C in question
2D2E	CD	43	2D			RANGEMES	CALL	PRINT
2D31	3F	20	6F	75			DT	'? out of range'
2D35	74	20	6F	66				
2D39	20	72	61	6E				
2D3D	67	65						
2D3F	00	3A					DD	CRLF+MSB

Addr	B1	B2	B3	B4	E	Label	Opcc	Operand
2041	37						STC	
2042	C9						RET	;set error flag
2043						*		
2043	E3					PRINT	XTHL	;save HL get SP
2044	7E					LPRINT	MOV	A,M ;get character
2045	0D	08	0D				CALL	OUT ;print it
2048	23						INX	H ;advance pointer
2049	B7						ORA	A ;is MSB set
204A	F2	44	2D				JP	LPRINT ;keep sending
204D	E3						XTHL	;restore HL and adjusted SP
204E	C9						RET	
204F						*		
204F	21	00	00			ADRS	LXI	H,0 ;zero value
2052	0D	03	0D			LADRS	CALL	INPUT ;get character
2055	CA	52	2D				JZ	LADRS ;is it there
2058	0D	08	0D				CALL	OUT ;print it
205B	FE	0D					CPI	CR ;was it a return
205D	08						RZ	;thats it
205E	06	30					SUI	'0' ;reduce to hex
2060	0A	7B	2D				JC	INVAL ;invalid entry
2063	FE	0A					CPI	10 ;alpha character
2065	0A	72	2D				JC	SAB
2068	06	07					SUI	7 ;alpha bias
206A	0A	7B	2D				JC	INVAL ;bad character
206D	FE	10					CPI	16 ;number out of range
206F	02	7B	2D				JNC	INVAL
2072	29					SAB	DAD	H ;multiply address by 16
2073	29						DAD	H
2074	29						DAD	H
2075	29						DAD	H
2076	35						ADD	L ;combine new value
2077	6F						MOV	L,A
2078	03	52	2D				JMP	LADRS ;keep going
207B						*		
207B	0D	43	2D			INVAL	CALL	PRINT
207E	0D	0A					DD	CRLF
2080	3F	20	69	6E			DT	'? invalid response'
2084	76	61	6C	69				
2088	64	20	72	65				
208C	73	70	6F	6E				
2090	73	65						
2092	0D	8A					DD	CRLF+MSB
2094	37						STC	
2095	C9						RET	;set error flag
2096						*		
2096	7D					MOD	MOV	A,L ;get low byte
2097	E6	0F					ANI	0FH ;mask low nibble
2099	08						RZ	;if zero fine
209A	0D	43	2D				CALL	PRINT
209D	3F	20	62	61			DT	'? bad boundary address'
20A1	64	20	62	6F				
20A5	75	6E	64	65				
20A9	72	79	20	61				
20AD	64	64	72	65				
20B1	73	73						
20B3	0D	8A					DD	CRLF+MSB

Addr	B1	B2	B3	B4	E	Label	Opcd	Operand
20B5	37						STC	;set error flag
20B6	C9						RET	
20B7						*		
20B7	CD	43	2D			BADPROM	CALL	PRINT
20B8	3F	20	73	70			DT	'? specified portion of prom is not erased'
20BE	63	63	69	66				
20C2	69	65	64	20				
20C6	70	6F	72	74				
20CA	69	6F	6E	20				
20CE	6F	66	20	70				
20D2	72	6F	6D	20				
20D6	69	73	20	6E				
20DA	6F	74	20	65				
20DE	72	61	73	65				
20E2	64							
20E3	0D	8A					DD	CRLF+MSB
20E5	C3	8E	2B				JMP	RESTORE ;continue and restore registers
20E8						*		
20E8	7C					HEX	MOV	A,H ;first the high byte
20E9	CD	8D	2D				CALL	BYTE ;print hex byte
20EC	7D						MOV	A,L ;now the low byte
20ED						*		
20ED	CD	FD	2D			BYTE	CALL	NIBBLE ;print nibble
20F0						*		
20F0	0F					NIBBLE	RRC	;swap nibbles
20F1	0F						RRC	
20F2	0F						RRC	
20F3	0F						RRC	
20F4	F5						PUSH	PSW ;save A
20F5	E6	0F					ANI	0FH ;mask high nibble
20F7	C6	90					ADI	90H ;super short-cut
20F9	27						DAA	;technique for converting
20FA	CE	40					ACI	40H ;binary to ASCII
20FC	27						DAA	;ala NB
20FD	CD	08	CD				CALL	OUT ;print it
2E00	F1						POP	PSW ;restore A
2E01	C9						RET	
2E02						*		
2E02						PASS	DS	1
2E03						STACK	DS	1

III. THEORY OF OPERATION

3.1 ADDRESSING

Address input lines A0 to A9 are buffered in line receivers U13 and U14. The outputs of U13 and U14 are then connected to both the PROM and RAM memory address pins. Address input lines A10 to A15 are buffered in U12 before use on the board. Lines A10 to A12 are inverted by the buffers and used as inputs to decoders U8 and U9. These three lines enable one of eight outputs on U8 or U9, depending on which decoder is enabled. Note that since A10 to A12 are inverted, the decoding sequence is reversed. When A10 to A12 are all "0", the number 7 output of the enabled decoder is selected. Each of the eight outputs from each decoder is used to enable a specific 2708 PROM or the 1K block of on-board RAM, or one of the three 1K segments which are not used on this board.

Address input lines A13 to A15 are used to enable one or the other decoder. Jumper Areas E and F determine which specific 8K block of memory corresponds to each decoder. The decoders are enabled by the output of U18-13 and U10-6. (They are enabled when their D input is a logic low "0".) Which decoder is enabled by which line depends on the jumpering in Area G. Jumper Area G can be used to switch the memory blocks thus assigned to each decoder.

Inversion of the on-board PROM and scratchpad memory address within block B may be accomplished by changing the jumper in Area J. This jumper determines whether or not the A12 address line is inverted by U11-4 before being used by decoder U9.

Selection of which 1K segment of the memory space will be assigned to the on-board RAM and which three 1K segments will be returned for use by other boards is handled by U9 outputs pins 1, 2, 3, 4, gate U10-12 and jumpers in Areas I and H. Any time an input to gate U10-12 goes low, this board is inhibited from putting data on the DI bus by forcing the DI line drivers to the high impedance state. Therefore, the three outputs of U9 which are connected to the inputs to U10-12 cause output from this board to be inhibited when one of the corresponding addresses appear on the address bus. Likewise, whichever U9 output is tied to the CE input to the RAM will enable the on-board RAM when that address appears.

3.2 DATA INPUT/OUTPUT

The DO lines from the S-100 bus contain data from the CPU to the memory. RAM is contained in two 2114 chips (U1 and U2). U1 contains the low four data bits in each location and U2 the high four bits. Thus DO0 to DO3 are tied to the data pins of U1 and DO4 to DO7 to the data pins of U2. These data bus lines are also tied in parallel to the eight data lines of each 1K byte PROM chip.

Data outputs from the RAM and PROM are connected to the input of a tri-state line driver U16 or U17. This parallel bussing of outputs from the memory chips is possible since all data outputs on the chips are tri-state.

3.3 CONTROL SIGNALS

U15 buffers the data lines inputting to the board. This buffer is enabled so long as U5-10 is low, which is true if U4-11 is high, which is true if either the on-board RAM is being written to or if PROM socket 11 is being written to. This logic is accomplished as follows. U4-6 is the NAND of MWRITE and the inverted (active high at U5-4) chip select for PROM socket 11, so that U4-6 is low if both PROM socket 11 is selected and MWRITE is active. U20-6 is the NAND of MWRITE and the inverted RAM chip select (active high at U5-13) so that U20-6 is low if both RAM is selected and MWRITE is active. Since U4-11 is the NAND of U4-6 and U20-6, U4-11 will be high if either U4-6 or U20-6 is low.

Writing of data into the RAM is controlled by MWRITE. Depending on the jumper in Area B, MWRITE can be taken from the bus (if a front panel is used or if there is another source of MWRITE in the system), or it can be generated from SOUT and $\overline{\text{PWR}}$ on this board. To generate MWRITE on the board, when SOUT and $\overline{\text{PWR}}$ are both low, U18-10 is high. This signal is buffered at U14-9 and is available both to the bus and the board as MWRITE. MWRITE is Nanded with the RAM chip select (inverted to active high at U5-13), giving the RD/ $\overline{\text{WR}}$ signal for RAM. Why is this necessary, since the signals are combined within the 2114? It is not necessary in order to generate RD/ $\overline{\text{WR}}$, but to enable the data bus input driver U15, as explained above, we needed external active low signals specifically for writing to RAM and to PROM. Rather than putting another inverter on the board, the same signal is used for RD/ $\overline{\text{WR}}$ to RAM. A low on RD/ $\overline{\text{WR}}$ puts the chip in the write mode. Data on lines DO0 to DO7 will be written

into the RAMs, assuming the board has been addressed and the RAM selected by the chip enable from Area I.

When it is desired to read data from this board, the U19-6 must be low at the appropriate time, enabling the DI bus drivers U16 and U17. This is accomplished by generating the logic NAND function of numerous signals. When either block A or block B is selected, the output of U20-3 is high which is used as one input to U19-6. Another input to U19-6 is generated by SMEMR which indicates that a memory read is to be executed. SMEMR is inverted at U11-2, then gated through U18-1, before being connected to U19. To allow selective disabling of this board's data outputs for any of the three unused 1K memory blocks, the chosen chip select lines are connected to U10 pins 1, 2 and 13. So long as they are high (not active), then U10-12 is low. In combination with a low from U11-2 (inverted SMEMR), a high appears on U18-1, which goes to U19-1. Another input to U19-6 is from U18-4 which senses that both SOUT and SINP are low. The last input to U19-6 is PDBIN. When this signal is high it indicates that the DI lines are in the input mode. Therefore, when all four inputs are high, indicating on board memory can be read, U19-6 will go low, thus enabling the data output buffers U16 and U17.

The power on/reset jump feature is initiated by the $\overline{\text{POC}}$ or $\overline{\text{PRESET}}$ input (jumper option in Area D). Disabling of other system memory boards during the power on/reset jump is accomplished by the PHANTOM output from this board, assuming the other boards are so wired. The power on/reset feature is provided by an RS flip-flop in U20, with the $\overline{\text{POC}}$ or $\overline{\text{PRESET}}$ line from the bus connected to the set input (U20-9) of the flip-flop. The PHANTOM signal is generated by the U20-11 active low output, and the U20-8 active high output is used to set U18-13 low, thus enabling U8 or U9, depending on the jumper in Area G. Since the address on the bus will be 0000, this causes the processor to execute the first instruction in the enabled 8K block. If this instruction is a jump to the next instruction in the same block, then when that instruction is decoded causing a low at U10-8 and hence at U20-13, the flip-flop will reset and cancel the PHANTOM signal.

The PRDY signal can be tied to the WAIT input by jumpering Area K. If so, the PRDY driver is enabled whenever this board is addressed and the processor is not doing I/O (determined by U19 pins 9, 10, 12 and 13.) WAIT is low at this time, thus PRDY goes low, putting the processor in a wait state. This makes WAIT go high, so that when the next clock cycle occurs, PRDY goes high again. The result is a one-cycle WAIT state each time the board is addressed. Note there is an error in this logic: a wait state will be generated (if jumpered in Area K) so long as any part of blocks A or B are addressed, INCLUDING the 3K which are used by other boards. This other 3K may be a function such as video or disk controller, which should not have a wait state.

3.4 PROM PROGRAMMING

PROM socket 11 is used to program an EPROM. EPROMs are programmed as follows: With the desired data on the data inputs to the PROM and the desired low order address byte on the address lines to the PROM, chip select must be raised to 12V (rather than the usual 0 for reading and 5 for not-select.) Then after a delay of 10 micro-seconds, a 26V pulse on the chip's programming pin (pin 18) must occur for 400 micro-seconds. The CPU must be held in a wait state during this time, as well as an additional 1/2 micro-second. This will program one byte ONCE. Proper programming of 2708 EPROMs require that each byte be programmed 256 times, with a delay after each time. This is handled in software, which should program all the locations on the PROM once, and then repeat the cycle 256 times. Software does not have to send any special signal for programming a PROM, since hardware will interpret any memory write to the PROM as an intent to program it. Unintentional writing to the PROM will thus cause programming if the 26V supply is accidentally left on.

U3 contains two one-shots which are used to generate the timing for the programming pulse. Each of these one shots has different R and C values connected to it, creating different length pulses. A 10 micro-second active low pulse is generated at U3-4 and a 410 micro-second active high pulse is generated at U3-5. When these two are NEEDED together at U4-3, the result is a 400 micro-second active low pulse following a 10 micro-second delay, as desired. This pulse begins when PSYNC (bus line 76) and clock-1 (bus line 25) are NEEDED at U4-8 and put into U3-1 and U3-9, and at the same time the PROM socket 11 chip select arrives at U3-2 and U3-10. They will only fire if it is not a memory read cycle, because U11-2 keeps the one-shots reset (via reset pins U3-3 and U3-11) if SMEMR is active.

The low-high transition of the 410 micro-second pulse at U3-5 generates an active low on XRDY (bus line 3) by inverting it at U6-2, in order to put the CPU in a wait state. This stays low for 1/2 micro-second after the pulse is over because of an RC delay tied to U6-2.

The 400 micro-second pulse is converted to active open at U6-10 and U6-12. The program pulse of 26V is then generated by a 2N3643 transistor, using a supply voltage from U7 and related circuitry. U7 is turned on by the sliding programming switch. This switch must ONLY be on when programming a PROM, because erroneous writing to that PROM will otherwise alter it when not desired.

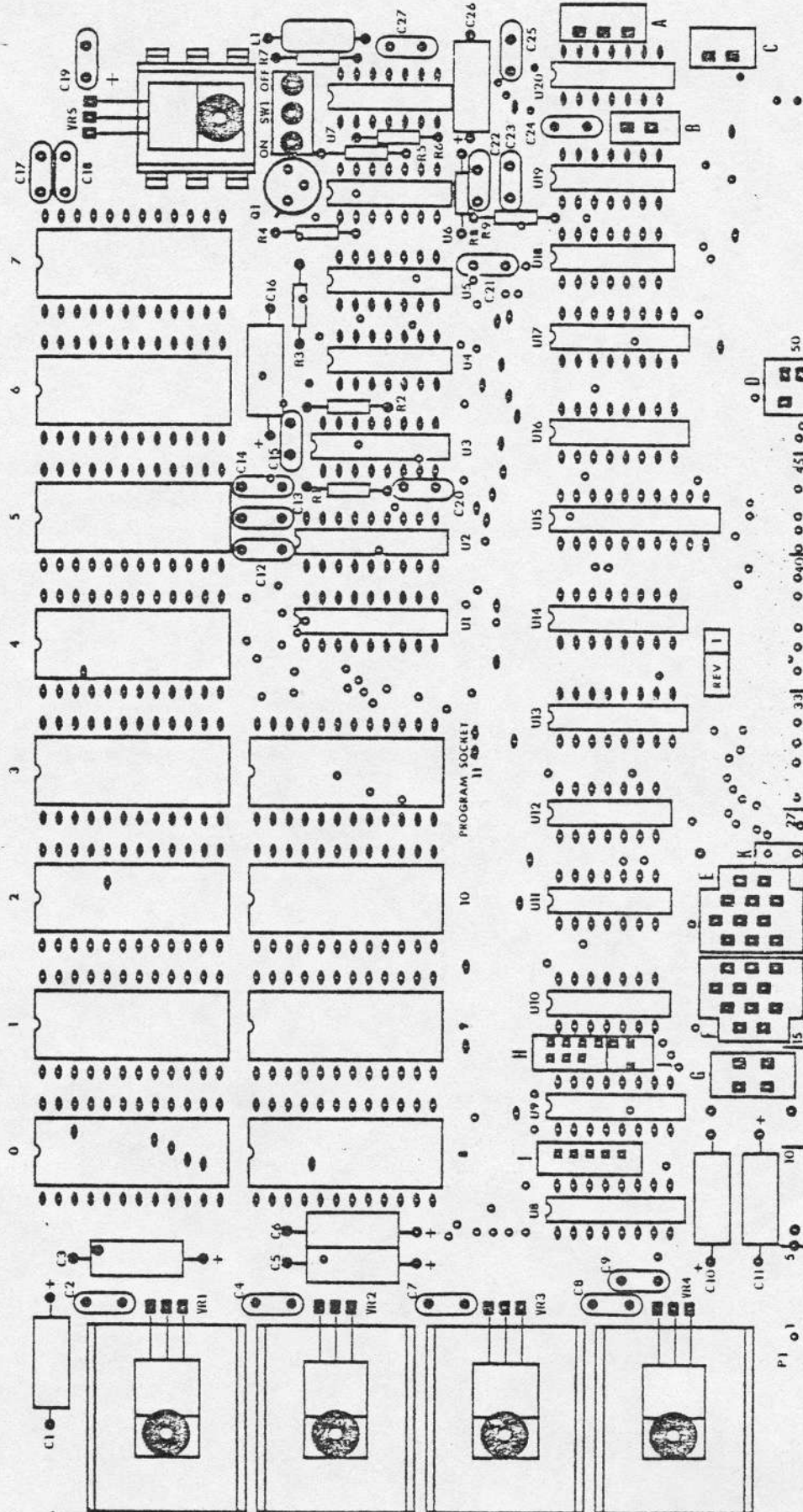
When the pulse is over and the wait line is released, the CPU is released to increment the address and program the next byte.

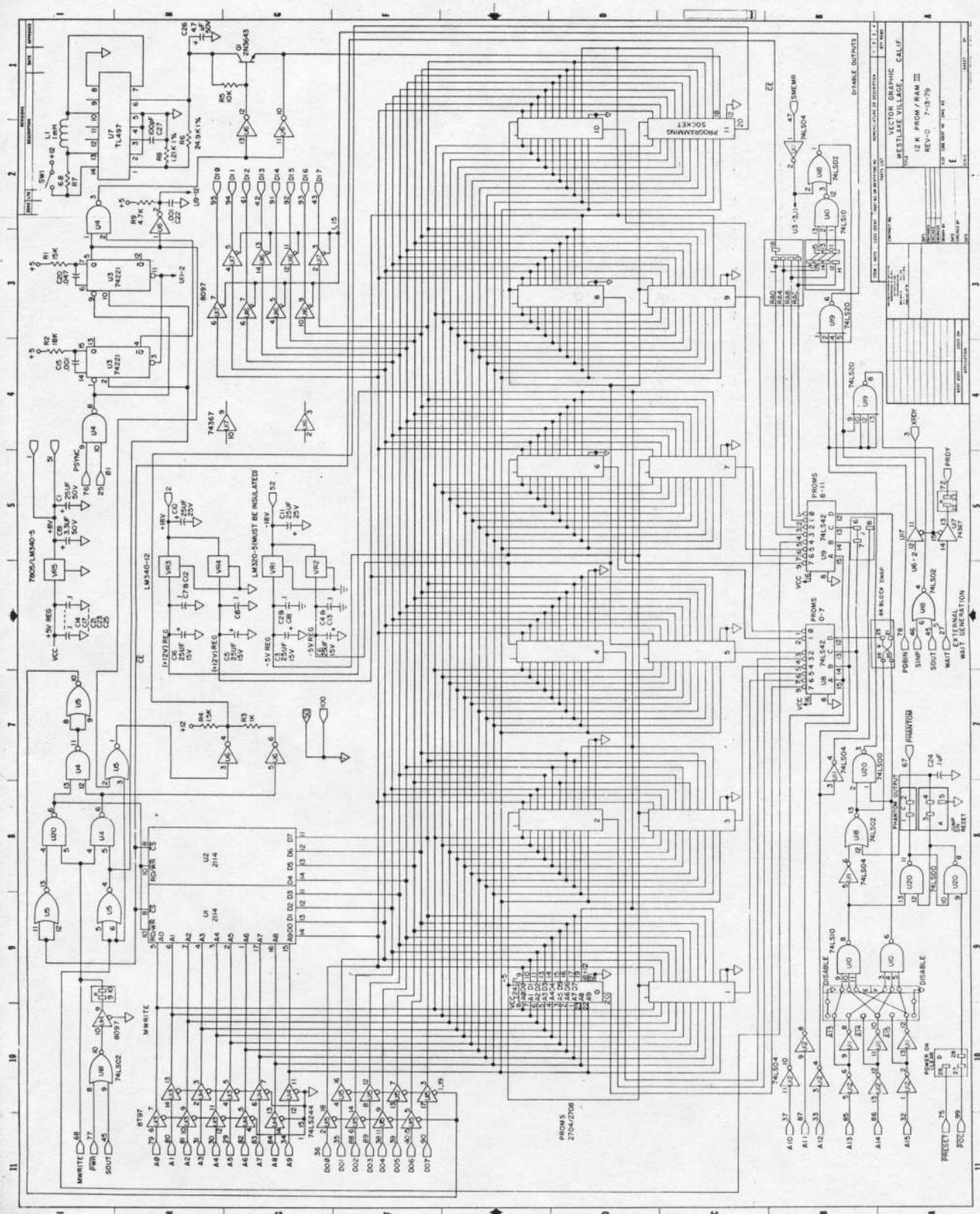
3.5 POWER SUPPLIES

Power for this board is obtained from the unregulated +8V and plus or minus 18V supplies in the system.

Regulation of the input voltage to the required -5V and +12V is obtained by the use of four three-terminal regulators. Dual regulators are used to insure ample supply current. The +5V supply is regulated by one regulator. Bypass filtering on all power lines is accomplished by multiple electrolytic capacitors for each supply voltage. This filtering insures stable noise free operation of the board. Capacitors are also used on each regulator input for high frequency bypassing and regulator stability.

The +26V programming supply is produced from the +12V regulated supply by a TL497 switching voltage regulator in a low-power step-up configuration, using a 1 mH coil.





1
2
3
4
5
6
7
8
9
10
11

VECTOR GRAPHIC CALCULATOR
WESTLAKES VILLAGE, CALIF
REV. 0 7-13-79
E

PROMS
2704/2708

PROMS
74LS145 10-BIT BCD-TO-DECIMAL DECODER
74LS149 4-BIT BCD-TO-DECIMAL DECODER
74LS164 8-BIT SHIFT REGISTER
74LS165 8-BIT SHIFT REGISTER
74LS173 3-BIT D-TYPE FLIP-FLOP
74LS175 4-BIT D-TYPE FLIP-FLOP
74LS177 8-BIT D-TYPE FLIP-FLOP
74LS179 4-BIT D-TYPE FLIP-FLOP WITH CLEAR
74LS244 8-BIT BUS BUFFER/DRIVER
74LS245 8-BIT BUS TRANSCEIVER
74LS247 8-BIT BCD-TO-DECIMAL DECODER
74LS249 8-BIT BCD-TO-DECIMAL DECODER
74LS250 8-BIT BCD-TO-DECIMAL DECODER
74LS251 8-BIT BCD-TO-DECIMAL DECODER
74LS252 8-BIT BCD-TO-DECIMAL DECODER
74LS253 8-BIT BCD-TO-DECIMAL DECODER
74LS254 8-BIT BCD-TO-DECIMAL DECODER
74LS255 8-BIT BCD-TO-DECIMAL DECODER
74LS256 8-BIT BCD-TO-DECIMAL DECODER
74LS257 8-BIT BCD-TO-DECIMAL DECODER
74LS258 8-BIT BCD-TO-DECIMAL DECODER
74LS259 8-BIT BCD-TO-DECIMAL DECODER
74LS260 8-BIT BCD-TO-DECIMAL DECODER
74LS261 8-BIT BCD-TO-DECIMAL DECODER
74LS262 8-BIT BCD-TO-DECIMAL DECODER
74LS263 8-BIT BCD-TO-DECIMAL DECODER
74LS264 8-BIT BCD-TO-DECIMAL DECODER
74LS265 8-BIT BCD-TO-DECIMAL DECODER
74LS266 8-BIT BCD-TO-DECIMAL DECODER
74LS267 8-BIT BCD-TO-DECIMAL DECODER
74LS268 8-BIT BCD-TO-DECIMAL DECODER
74LS269 8-BIT BCD-TO-DECIMAL DECODER
74LS270 8-BIT BCD-TO-DECIMAL DECODER
74LS271 8-BIT BCD-TO-DECIMAL DECODER
74LS272 8-BIT BCD-TO-DECIMAL DECODER
74LS273 8-BIT BCD-TO-DECIMAL DECODER
74LS274 8-BIT BCD-TO-DECIMAL DECODER
74LS275 8-BIT BCD-TO-DECIMAL DECODER
74LS276 8-BIT BCD-TO-DECIMAL DECODER
74LS277 8-BIT BCD-TO-DECIMAL DECODER
74LS278 8-BIT BCD-TO-DECIMAL DECODER
74LS279 8-BIT BCD-TO-DECIMAL DECODER
74LS280 8-BIT BCD-TO-DECIMAL DECODER
74LS281 8-BIT BCD-TO-DECIMAL DECODER
74LS282 8-BIT BCD-TO-DECIMAL DECODER
74LS283 8-BIT BCD-TO-DECIMAL DECODER
74LS284 8-BIT BCD-TO-DECIMAL DECODER
74LS285 8-BIT BCD-TO-DECIMAL DECODER
74LS286 8-BIT BCD-TO-DECIMAL DECODER
74LS287 8-BIT BCD-TO-DECIMAL DECODER
74LS288 8-BIT BCD-TO-DECIMAL DECODER
74LS289 8-BIT BCD-TO-DECIMAL DECODER
74LS290 8-BIT BCD-TO-DECIMAL DECODER
74LS291 8-BIT BCD-TO-DECIMAL DECODER
74LS292 8-BIT BCD-TO-DECIMAL DECODER
74LS293 8-BIT BCD-TO-DECIMAL DECODER
74LS294 8-BIT BCD-TO-DECIMAL DECODER
74LS295 8-BIT BCD-TO-DECIMAL DECODER
74LS296 8-BIT BCD-TO-DECIMAL DECODER
74LS297 8-BIT BCD-TO-DECIMAL DECODER
74LS298 8-BIT BCD-TO-DECIMAL DECODER
74LS299 8-BIT BCD-TO-DECIMAL DECODER