VIDEO OPERATION

PolyMorphic Systems

460 Ward Drive Santa Barbara California 93111 (805) 967-2351

NOTE

The status port feature is not available on factory-assembled video terminal interface cards. If you have a factory-assembled card, please disregard the discussion of the status port in this manual.

The PolyMorphic Systems video terminal interface card is guaranteed to work only with these video monitors:

Hitachi WM909V
Hitachi WM972V
Hitachi P-04 with Pickles and Trout conversion kit.
Hitachi P-05 with Pickles and Trout conversion kit.

Sanyo VM4092
Sanyo VM4155

Javelin VM9A

This manual is PolyMorphic Systems part number 810115.

Revision F

VIDEO TERMINAL INTERFACE THEORY OF OPERATION

TABLE OF CONTENTS

The PolyMorphic Systems Video Terminal Interface (VTI) provides a complete interface between a microcomputer main unit such as the PolyMorphic Systems POLY 88 or System 88 and keyboard and video monitor. It produces a full range of characters, letters, numbers, and graphics, on a video screen.

This manual describes the operation of the VTI.

1. VTI theory of operation and block diagram.

The principal functional blocks which form the video terminal interface are shown in the figure below.

The on-board memory is connected in parallel with the keyboard input port to an array of I/O buffers driving the S-100 data bus. This allows the transfer of information between the memory and the data bus or between the keyboard and the data bus.

These data transfers are controlled by logic driven from the address and control lines. For example, the processor can read or write a location in memory just as it would with any main memory -- it outputs the memory address (16 bits) while signaling a read or a write by the state of the control bus. The six most significant address bits are compared to the jumper-selected bits. If these bits match, then the remaining 10 address bits are gated through to select the memory location.

At this time the appropriate bus drivers are enabled to read from or write into memory, according to the control bus command. If the control bus signals neither a memory read nor a memory write, but rather an input instruction, then the keyboard buffer is enabled instead of the memory. Note that the input port address (8 bits) is the same as the most significant bits of the 16 bit memory address.

When the processor is not accessing the video terminal, i.e. not accessing video memory , then the video refresh circuitry takes control of the memory. The memory locations are scanned by the control and sync generator, with the memory data being fed into a character ROM. This read-only memory stores the video dot pattern of each ASCII character.

The character font is a 7 X 9 matrix, so that each ASCII character has 9 memory blocks 7 bits wide in the ROM. Thus, each line of characters on the screen results from many sequential scans through a line of memory locations. Each scan increments a counter, so that the ROM reads off the next line of the dot matrix. Each clock of 7 bits read from the character ROM is loaded in parallel into a shift register and shifted out serially. The signal is then mixed with the video sync signals to form the composite video output.

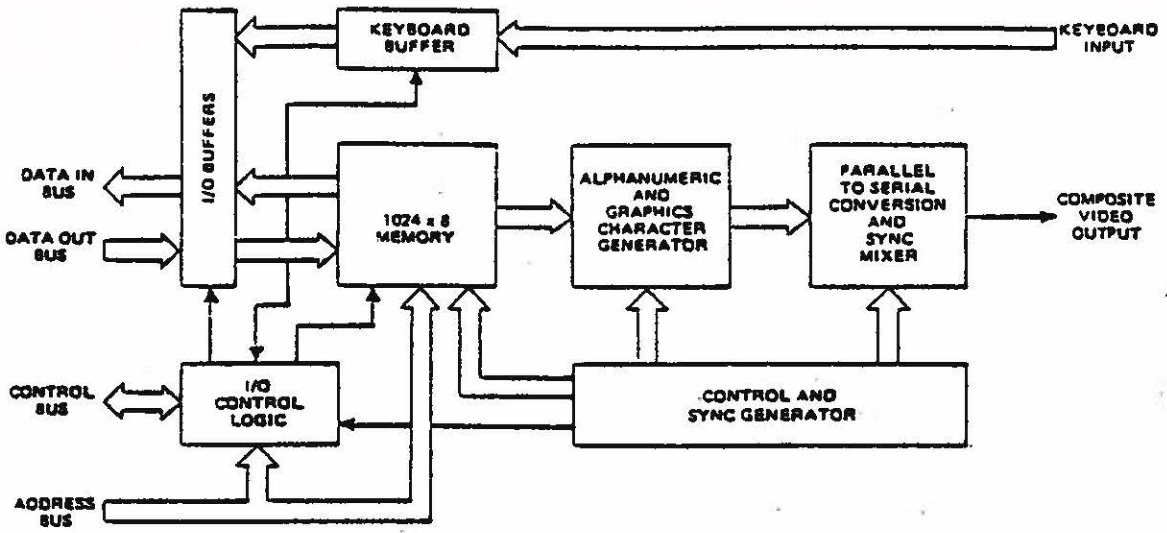


Figure 1. Block diagram.

A more detailed view of the card circuitry is shown in the schematic diagram at the end of this volume. We are now going to examine the board in some detail to see how it performs its various functions. The level of complexity is fairly high; not all readers will find it useful.

Look at the schematic and note that all the on-board memory, data latches, and bus drivers are connected to a common on-board data bus. We will be referring to the video terminal interface (VTI) data bus as the on-board bus, and the S-100 bus as the external bus.

Another point of terminology is sweep vs line. Each character on the monitor screen consists of a selection of dots in a dot matrix seven dots wide by nine high, embedded in a field of ten by fifteen dots (to provide space between characters). So the monitor picture tube must sweep fifteen times to produce one line of characters.

1.2 Symbol generation.

With a high on the BS- (bus strobe) line from IC8 pin 7 to the MUX strobe lines, ICs 17, 18, and 19 pins 1, the addressed portion of the RAM is continuously sent to the internal data bus in the refresh mode. Eight-bit display data on the internal data bus is sampled and held in the latch IC40 whenever there is coincidence (in IC30) of a dot pulse from the dot clock IC29 and an "end of character" (EOC) signal (tenth dot carry) from the "dot counter" IC13. In the absence of one in the MSB (most significant bit) from the latch, MUXs (multiplexors) IC33 and IC36 pass the seven-dot conversion pattern of this display data from the character-generating ROM (read-only memory) IC37 to the least significant bits of the output shift register IC35. When the eighth bit specifies that graphics are being generated, these MUXs switch to select all ten bits of the data for the shift register from IC38 and IC39. ICs 38 and 39 are, in effect, the graphics generation ROM.

In the case of non-graphics characters, the first three dots of every character space are always low to create spaces between letters. Note that, while the latched data for the nth character position of the sweep is identical for fifteen consecutive sweeps, the ROM output may vary in each sweep, according to the additional addressing from the sweep counter half of IC15. The sweep counter is self-resetting after every fifteenth sweep, and this resetting action is accumulated in the line counter half of IC15.

In similar fashion, the dot counter IC13 is self-resetting every tenth dot, and its output is accumulated in the symbol counter IC16. The combination of line and symbol counter outputs determine the address of each individual character stored in the memory (ICs 20 through 28). Since all of these counters (dot and character, sweep, and line) are reset by appropriate relationships to the horizontal and vertical sync (respectively) of the video raster, the lowest memory address will always contain the record for the top left corner of the display. Corresponding relationships are similarly maintained between other addresses in memory and positions in the display field.

1.3 Raster and timing.

Horizontal sync, vertical sync, and vertical blanking are timed by subcounting the absolute frequency system clock. Horizontal blanking is initiated at the end of sweep by subcounting the variable frequency dot clock IC29, and blanking is maintained by a variable-duration one-shot IC34. Varying the "pos" pot changes the one-shot delay and thus the position in the next sweep where the display is again unblanked. Varying the dot clock frequency ("width" pot) changes the rapidity with which the full line character count will accumulate to initiate horizontal blanking and therefore the distance across the screen that is used for display.

A crystal-controlled clock is generated by IC45. The clock is divided by sixteen in IC1 and again by thirteen in IC2. A carry on exit from the highest (16th) state (all four output bits = 1, or binary 15) is used to preload a binary 3 into the same IC2 so that it may again divide by 13. This binary 3 at the IC2 outputs will therefore last for one-thirteenth of the period between carries and is passed through IC3a for horizontal sync.

The same carry triggers the horizontal blanking one-shot. The carry is also used to clock the 4-bit binary sweep counter (IC15a) which is used both to address the character generation ROM and to signal the line counter (IC15b) every fifteen sweeps that a new display line is being addressed.

When 16 line counts (16 X 15 = 240 sweeps) have accumulated in IC15b, the carry resulting from the transition from its binary 15 state to its binary zero state is inverted by IC5 to set the vertical blanking flip-flop IC4. In addition to blanking the screen, IC4 also enables the 1 of 8 decoder IC12. Pin 15 of IC12 will go low, producing a vertical sync pulse.

This vertical sync lasts for eight blanked sweeps until IC15a resets itself and advances the line counter. IC3 ANDs this vertical sync with the horizontal sync carry, so that interruptions in the wide vertical sync pulse will maintain horizontal sync.

Further subcounts of the sweep and advances of the line counter accumulate in IC15 until IC12 decodes the 23rd blanked sweep to trigger the pulse stretcher IC34. (Line counter = 1 and sweep counter = 8.) IC34 is a very short duration one-shot which terminates the vertical blanking (disabling IC12) and also resets the sweep and line counters for top of the page addressing. The subsequent termination of horizontal blanking has the character counter IC16 reset to prepare all addressing from the top left of page as described below.

For 50Hz operation, JMP3, 5, and 6 are jumpered differently. The vertical sync pulse begins on the 23rd blank line and lasts for seven lines. The pulse stretcher (IC34) is triggered when IC12 pin 13 is low and IC15 pin 9 is high. This occurs on the 83rd blanked line. Each frame contains 323 lines total.

1.4 Symbol and raster synchronization.

Termination of the horizontal blanking one-shot IC34a re-enables the dot clock oscillator IC29a but does not unblank the screen. At this time, symbol count addresses are set to zero, but the data latch IC40 contains unrelated data sampled with some previous address. Similarly, the shift register IC35 contains old data. The screen has been darkened by the dot blank flip-flops of IC32 which have been held set by the horizontal blanking. The symbol counter IC16 MSB is zero and IC4 pin 6 is typically high; therefore, IC30-3 is low, presenting a zero to the D- input of IC32. IC4 pin 6 goes low whenever BS- goes low to blank the screen during memory access. It is reset by the next EOC.

After the first ten dots from the dot clock, the shift register (which is shift-clocked by dots) is emptied and the EOC (end-of-character) signal from the dot counter IC13 sends load signals gated through IC30 to both the data latch and the shift register. Since propagation time through the ROMs and MUXs is not zero, the latch now contains beginning-of-line data, but the register is loaded with different but still useless data.

The same end-of-character pulses, however, have advanced the symbol address in IC16 by 1 and have also propagated the zero at the input of the first DBLK (dot blank) flip-flop to the second flip-flop. The ROM and MUX paths present valid first symbol data to the shift register, so that the second EOC pulse loads first symbol dots into the shift register and second symbol data into the latch. They also propagate the zero through the second dot blank flip-flop so that the screen is unblanked for the first symbol data shifted out of the register by the subsequent ten dots.

When the 64th end-of-character pulse accumulates in the character counter, it loads the data latch with the 64th character and the register with the next-to-last character. Simultaneously, the MSB of the symbol counter presents a 1 to the dot blank flip-flops through IC7a and IC30a, and the next 20 dots shift the last two symbols out to the video, and the 1 through the flip-flops to blank the screen in the 65th character position. The dot clock runs, and the dot and symbol counters keep accumulating, but the MSB of the character counter maintains its 1 input to the dot blank flip-flops until either double the number of symbols is counted or, as normally, horizontal sync and horizontal blanking occur to stop the dot clock, reset the symbol counter, and reaffirm the dot blank.

Clocked by the sweep counter reset, the line counter will increment every fifteen sweeps until the vertical blanking process described above resets the MSBs of the addressing system.

1.5 External bus and keyboard interfacing.

The comparator (IC6) compares the 6 MSBs of the external address bus with the switch pattern selected for display memory addressing.

In the switched condition, RAM address is determined by the ten LSBs on the external address bus instead of by the combination of the line and symbol counters used in the display refresh mode.

The BS- strobe enables the line drivers that put internal data bus information onto the external data bus. If INP+ (pin 46) is also true, keyboard data latched in IC41 will be sent to the CPU via the line drivers. The MEMR+ signal, if true, similarly enables the memory output to the on-board bus. (INP+ and MEMR+ cannot both be true simultaneously.) If MWR+ (pin 68) is high with BS- low, the line receivers are enabled by IC7s to transfer the external data bus to the internal data bus and write it into the on-board RAM. Thus, CPU data can be written into or read from display memory, and keyboard data can be input to the CPU.

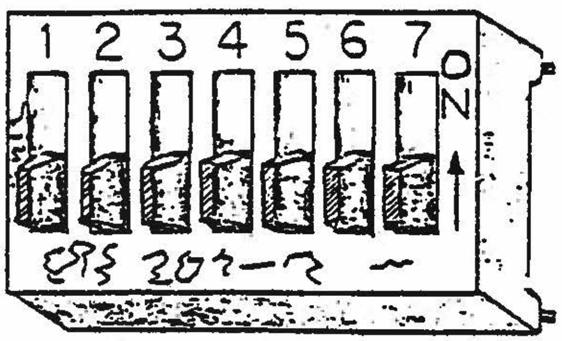
Keyboard data can be latched into IC41 in response to "key pressed" strobes of jumper selected polarity. The key depressed condition is shown by a low signal on INT- (IC41 pin 23). The INT- signal then passes through a buffer (IC31) to the vectored interrupt jumper section; the resulting signal then passes on to one of the interrupt pins on the S-100 bus.

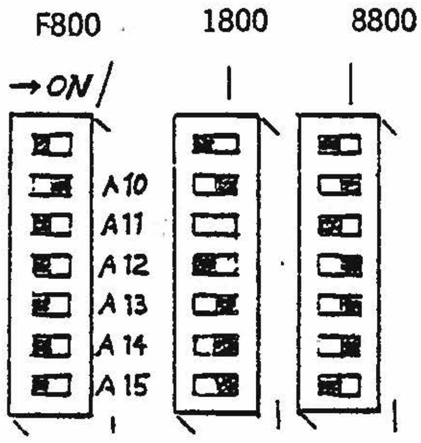
2. Option Selection.

2.1 Address location.

The VTI interacts through the S-100 bus as a block of memory and input port for the keyboard. The memory block can be located at any address from 0 through 63 K in 1 K increments. Software written for this product will usually locate it at hexadecimal address 8800 in systems other than the POLY 88, in which it is at F800, or the System 88 disk system, in which it is at 1800.

Set the address as required by matching the appropriate figure below.





black is switch

2.2 Connect keyboard

Near the upper right hand corner of the video terminal interface card is the keyboard input port. This port provides a latched 8 bit parallel input capability which interfaces with any ASCII keyboard. Keyboards usually indicate a keystrike to the computer via a strobe line, in addition to the eight parallel input lines.

The signal on this line changes state-- from high to low or from low to high-- to indicate a keystrike. Hookup varies according to whether the strobe on your keyboard is "positive going" (rising to indicate keystrike) or "negative going" (dropping to indicate keystrike).

2.2.1 Connector configuration.

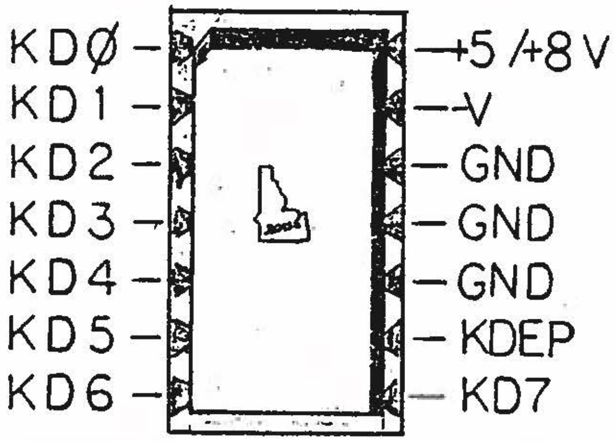
The parallel input from the keyboard is designed to come in over a ribbon cable terminated by a male DIP connector. This plugs into the 14 pin DIP socket near the upper right hand corner of the card. The 8 parallel input lines are connected to pins 1 through 8 of this socket (J-1), with 1 being the least significant bit. Pin 9 carries the "positive going" or "negative going" strobe. Pins 10, 11, and 12 are grounded. Pin 13 is the output from the optional negative voltage regulator used when the keyboard requires a negative supply. Pin 14 carries +5 volts as the primary supply for most keyboards. JMP8 allows +8 volts unregulated power at Pin 14 if desired.

( ) PolyMorphic Systems keyboard #009010 (keyboard using ESC key and U, D, R, and L keys to move cursor) requires +5 volts. No modification is required for this keyboard.

( ) PolyMorphic Systems keyboard #009012 (keyboard with arrow keys for cursor movement) requires +8 volts. Cut the trace from the center pad to the pad furthest from the regulator on the BACK of the card, and insert a jumper from the middle pad of JMP8 to the pad nearest the regulator within the area designated JMP8.

WARNING

FAILURE TO CUT THE TRACE SUPPLYING +5 VOLTS WHEN JUMPERING IN +8 VOLTS WILL DAMAGE COMPONENTS AND VOID THE WARRANTY.



2.2.2 Keypress strobe.

When the processor accesses the video terminal interface with an input instruction, the state of the keyboard input latch is transferred to the accumulator. Proper use of the keyboard requires that the processor must verify two conditions before using the input data. It must determine that

1) a key has been pressed, and

2) this particular key depression has not been previously serviced.

These functions are accomplished by making the keypress strobe information available to the processor.

The keypress strobe line is an additional keyboard output line in parallel with the data lines. This line signals each depression by a pulse. This test-function informs the processor that the necessary input conditions have been met. The keypress strobe signal is used in one of two ways:

1) The pulse interrupts the processor by setting an interrupt service latch contained on the input buffer, or

2) the interrupt request latch is made available on data bit 0 of the status port; the keypress strobe

is made available on data bit 7.

NOTE: The status port should be accessed no more often than 1000 times per second. More frequent access may cause noticeable interference to character generation.

2.2.3 Keystrobe Selection.

The key depressed strobe may be one of four types. Attach a strobe line to a logic probe to determine the type. PolyMorphic Systems keyboards 009010 and 009012 are type 1.

1. It may be normally low (below 0.8V), go high (above 2V) when a key is depressed, and return low when it is released.
2. The keystrobe may be normally high, go low on a key depresssion, and return high on release.
3. The keystrobe may be normally low, generate a positive pulse on key depression, and immediately return low.
4. It may be high and generate a negative-going pulse on key depression.

If your keyboard is type 2 or 3, the jumper is already configured correctly.

If it is a type 1 or 4, cut the minus trace from the center pad of JMP7 and jumper from the center pad to the + labeled pad.

2.3 Optional voltage regulator.

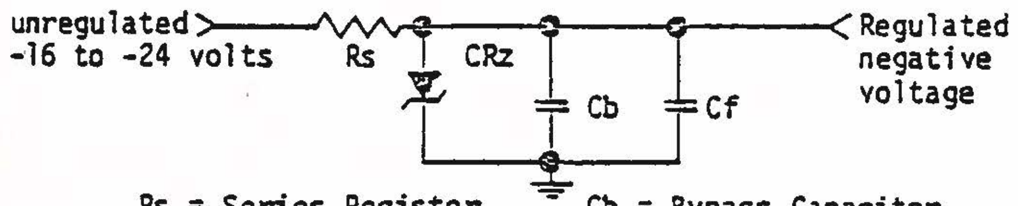
Provision has been made for the optional negative voltage regulator required by a number of keyboards. The pads and traces for this voltage supply are located just above ICs 22 and 23. This circuit regulates the -16V supply by means of a resistor and zener diode stabilized by two capacitors. The four components are R14, C29, C28, and D2. The choice of resistor and zener values depends on the voltage and current requirements of the keyboard. PolyMorphics Systems keyboards do not require the negative voltage regulator.

2.3.1 Installing Optional Voltage Regulator.

The component values of the customer-provided zener keyboard supply must be calculated. The values depend not only on the required voltage, but also on the required current.

Determine the required voltage and current values by consulting the keyboard manufacturer or distributor.

The supply circuit is represented by the following schematic (the component labels have been generalized to avoid conflicts between different card revisions; see schematic for actual part designations):



Rs = Series Registor
CRz = Zener Diode
Cb = Bypass Capacitor
Cf = Filter Capacitor

The bypass capacitor (Cb) should be a 0.1mF or 0.01mF ceramic disk; the value is not critical. The filter capacitor (Cf) should be a 10mF 25-35 volt tantalum with the positive lead to ground (ground is positive with respect to the negative regulated voltage).

The series resistor (Rs) and zener diode (CRz) are more difficult to calculate. Two values must be calculated for each part: resistance and wattage for Rz; voltage and wattage for CRz.

1. CRz voltage. Voltage should equal the required regulated voltage.

2. Rs resistance. To determine the resistance of Rs, use the specified unregulated voltage value closest to zero. This is -16 volts according to bus specifications. Take the difference between this value and the regulated value.

EXAMPLE: For regulated -12 volts, -12-(-16) = 4 volts.

Divide the remainder by the maximum required current in amps.

EXAMPLE: for 10mA current = 0.010 amps, 4 volts/0.010 amps = 400 ohms.

Use a convenient standard resistance approximately 20 percent lower than the value calculated above.

EXAMPLE: 400 ohms minus 20 percent = 400-80=320. 320 ohms is not a standard value; use 330 ohms or 270 ohms.

3. CRz wattage: To determine the wattage rating for CRz, use the worst-case current. Assume all the current passes through the zener (this can happen if the keyboard is disconnected and the -16 supply is unloaded).

EXAMPLE: Using Rs = 330 ohms, Iwc = 12/330 ohms - 0.03636 amps.

Now calculate the wattage for CRz.

EXAMPLE: 12 volts x 0.03636 amps = 0.436 watts. Use a higher wattage than calculated, like 1/2 watt or higher for the given example.

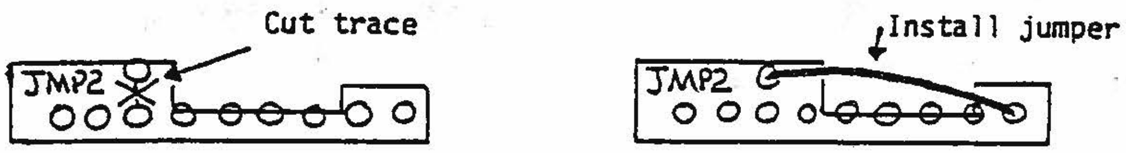
Install the components. note the capacitors Cf and Cb can be in either capacitor position-- they are in parallel-- as long as the tantalum polarity is correct.

2.3.2 Interrupt wiring.

The VTI card as designed is compatible with the PolyMorphic Systems product line, which uses vectored interrupts. If you use the VTI in another product, you may need to make a modification:

( ) If you use the card in a system that does not use interrupts at all, cut the trace in the JMP2 area.

( ) Many systems use non-vectored interrupts. If you use it in a system with non-vectored interrupts, cut the trace in the JMP2 area and jumper the top pad to pad PINT- in that area.



( ) If you use it in a system that is not a PolyMorphic Systems product, but that does have vectored interrupts, cut the trace and jumper to one of the VI pads 0 through 7 in the JMP2 area as required.

2.4 Interfacing card to main unit.

PolyMorphic Systems provides a cable set to interface the VTI to the rear panel of the POLY 88 (separate order; part number 100010). One of the two cables picks up the video output signal from the two-pin header in the VIDEO OUT and conveys it to the location of the coaxial connector on the rear panel. The other interfaces the keyboard port with the D connector on the rear panel; it includes a parallel mini-card that mounts at the D connector location. The keyboard cable is terminated with a male DIP connector that plugs into the video card keyboard port socket near the voltage regulator.

3. Software.

This software is for use in systems without the PolyMorphic Systems monitor ROM. The monitor ROM includes a video driver routine.

3.1 Video Typewriter.

Both the input to and the output from a computer is ordinarily a string of characters, whether it be characters typed in from a typewriter-like keyboard or output from the computer to a printer. Not all of these "characters," however, strictly correspond to a printed symbol, like a letter. Consider the output to a printer. Some "characters" will cause the printer to perform some function other than a keystrike-- such as carriage return or backspace.

The VTI is essentially a block of memory, and at the hardware level does not distinguish between characters and other functions. Without an intervening program, the VTI would send a "carriage return" on to the screen as a symbol, rather than returning the cursor to the beginning of the line.

We append to this manual three versions of a program that accepts a string of ASCII characters and causes them to appear on the screen exactly as the characters would be printed by a printer. (The first version is a very complete one for non-POLY 88 users; the others are shortened versions for POLY 88 users and for non-POLY 88 users.) "Carriage return" causes the cursor to return to the beginning of the line, "line feed" causes it to move down one line, and so forth.

The program includes a keyboard input routine, which puts the characters you type on the keyboard directly onto the screen, with proper carriage return, line feed, and other functions. Load the program as written. To use the computer as a "TV typewriter," connect the keyboard to the parallel input port provided on the video board.

The program can be used to interpret the output of another program which would ordinarily be sent on to a printer, so as to put the appropriate visual display on the screen.

The first program assumes the user has a defined stack area. If you have no preassigned stack location, execute an LXI SP, 0FFFH.

Programs ordinarily send a character from the accumulator to a serial output port in response to the instruction OUT. The program includes a subroutine called OUT (located at address 1D00H in the first version). When called, this subroutine interprets the character in the accumulator as required to put it on the screen. In converting a program to run with the VTI, substitute CALL OUT.
Note that the shortened versions of the program do not include all of the commands below.

VIDEO TERMINAL SOFTWARE - COMMAND SUMMARY

	Control Character	Function
	H	Home cursor
	R	cursor Right
	L	cursor Left
Cursor	U	cursor Up
Controls	D	cursor Down
	E	Erase screen
	X	delete character

	I	Insert/delete mode set
	T	Text(reset I/D mode)
Mode Commands	F	auto line Feed mode set
	N	Normal TTY(reset ALF mode)
	S	Scroll mode set
	P	Page (reset scroll mode)

Striking the LINE FEED key advances the cursor one line, except when the cursor is on the bottom screen line in scroll mode; then the cursor remains fixed, and the page scrolls.

CARRIAGE RETURN retreats the cursor to the beginning of the line, blanking the line from the end unless the I/D mode is set.

3.2 Graphics.

The PolyMorphic Systems VTI includes full graphics capability. Any or all character locations on the screen can be used in a graphics display.

When a screen location is part of a graphics display, it is sub-divided into six parts, thus:

5	2
4	1
3	Ø

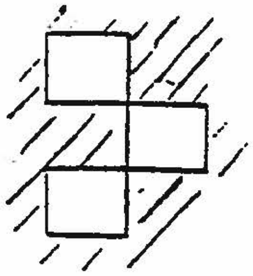
(NOTE: Graphics display uses the entire screen location, including the border area that is kept dark to provide space around other characters.) Each of the six "cells" of the screen location corresponds to one bit in the byte stored in the screen location. The "zero bit" corresponds to cell 0, etc.:

	D7	D6	D5	D4	D3	D2	D1	DØ
Graphics	Ø	X	These select character					

X can be a Ø or 1 without affecting the character.

	D7	D6	D5	D4	D3	D2	D1	DØ
ASCII	1	These select character						

0 is "on" or "bright," 1 "off" or "dark." Thus, storing 01101010B (6AH) at a screen location produces this graphic at that location:



Thus 00 or 40 hex (00000000 or 01000000 binary) produce an all-bright graphic character. 3F or 7F hex (00111111 or 01111111 binary) produce an all-dark graphic character.

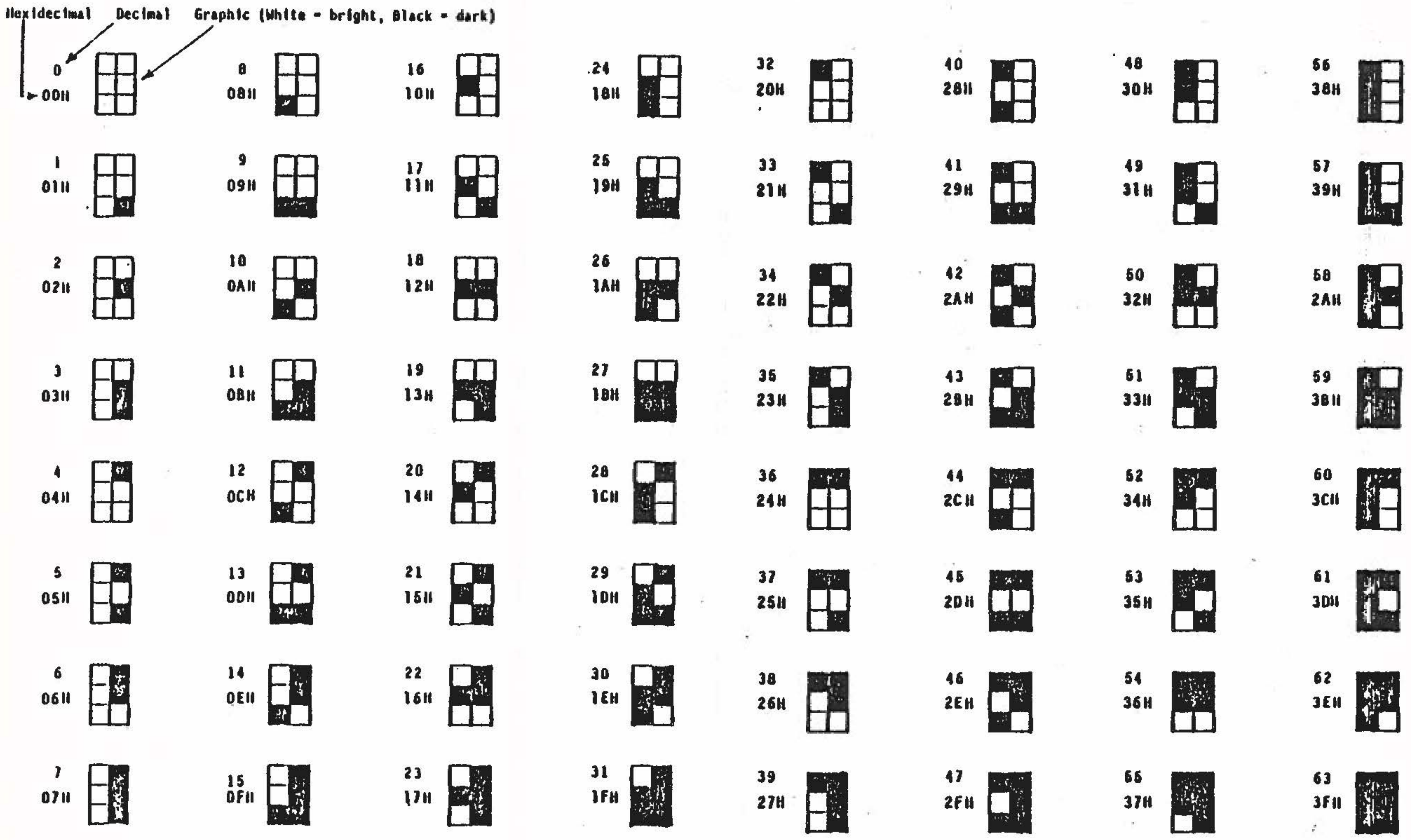
Appended below is a chart of all 64 possible graphics characters, with their associated hex values. Also shown is the ASCII character set produced by the PolyMorphic Systems video character ROM.

We also include a "game" program, LIFE, originally invented by John Conway and popularized by Martin Gardiner in his "Mathematical Games" Section of Scientific American in 1970. It illustrates the power of the graphics capability.

LIFE depicts the birth, growth, and death of a culture of cells. When a cell has one neighbor or no neighbors in the eight cells adjacent to it, it dies of loneliness. When it has four or more neighbors in the eight adjacent cells, it dies of overcrowding. It survives into the next generation whenever it has two or three neighbors. So a cell may live for just one generation, or may live for as long as the culture lives (or anything in between). A cell is born whenever an empty cell location has exactly three neighbors. (Cells are trisexual.)

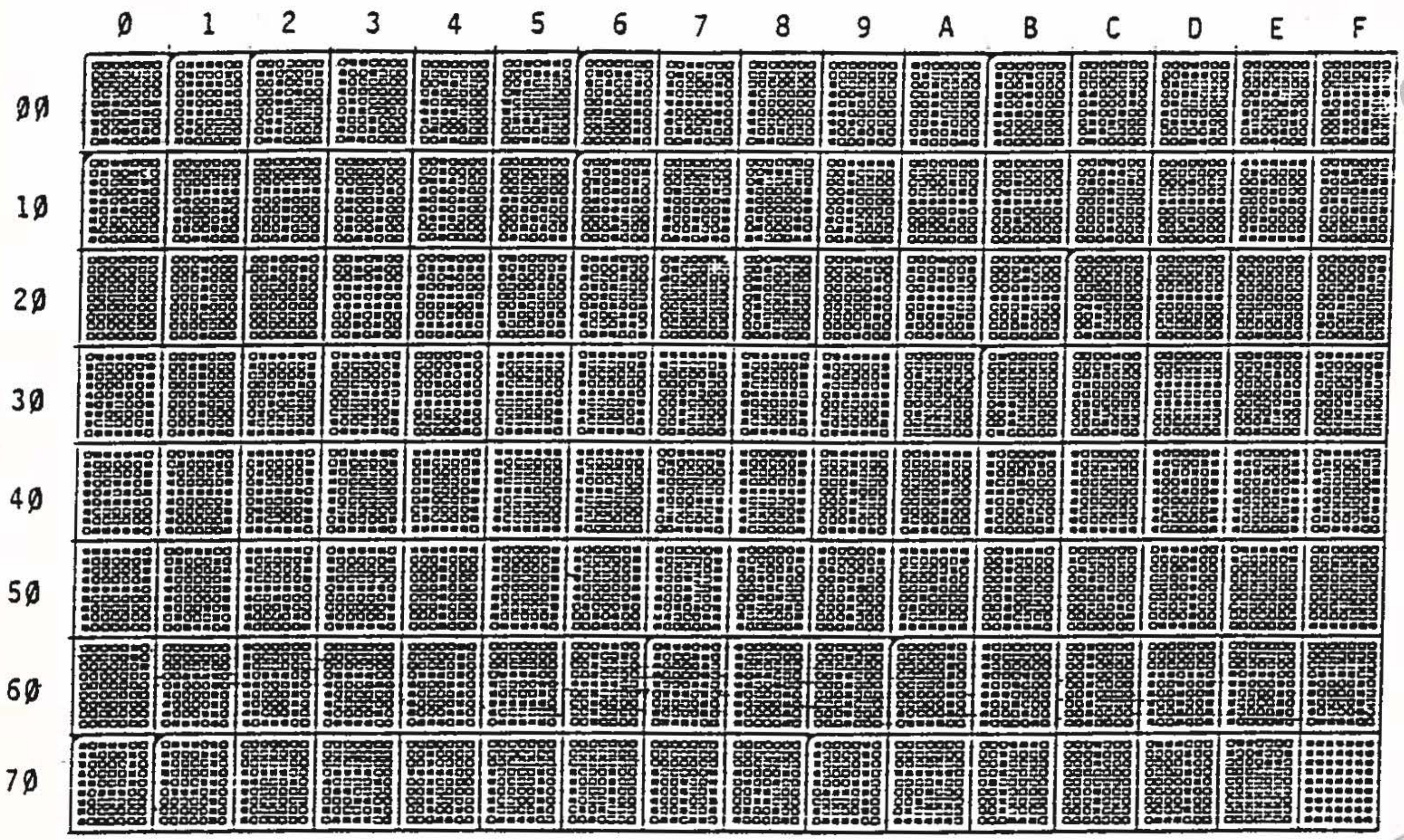
The game begins with an initial entry, or Divine Creation,of a seed organism (group of cells). The initial entry can be as simple or complex as you like. The life cycle of the resulting culture arises entirely from the nature of the initial entry given the rules of LIFE.

The following program executes the rules of LIFE on the video screen in graphics. Load both programs at the addresses indicated. Execute the screen clearing routine at 0F00. If your system has a stack is not already initialized, set it with an LXI SP, 0FFFH. Then you are ready to load an initial generation (by using the hex-to-graphic table in appendix D) into memory locations in the middle of the screen (such as 8A10H). When you are satisfied with your initial organism, execute the LIFE routine at address zero.



Graphic Character Set

6571A Character Fonts



Video Typewriter Routine-- long version for non-users of PMS monitor ROM.

```
Hex decimal
Address     Op          Mnemonic
            Code        Instruction                Comments
0000                    0100 SCRN EQU 8800H      *VIDEO SCREEN ADDRESS
0000                    0110 STR EQU 1CFFH       *STORAGE FOR SYMBOL UNDER CUR
0000                    0120 STS EQU 1CFEH       *STORE OUTPUT MODE
0000                    0130 CURS EQU 1CFCH      *STORE RELATIVE CURSOR LOCATI
0000                    0140 SEND EQU 8CH        *1ST BYTE OF SCREEN END
0000                    0150 LINE EQU 64         *LINE LENGTH
0000                    0160 CS EQU 0FFH         *CURSOR SYMBOL (RUB OUT)
0000                    0170 LT EQU 3FH          *LINE TERMINATION CHARACTER
0000                    0180 KBD EQU 88H         *KEYBOARD PORT ON VTI
0000                    0190  ORG 0000
0000 21 00 00           0200  LXI H,0
0003 22 FC 1C           0210  SHLD CURS
0006 7D                 0220  MOV A,L
0007 32 FE 1C           0230  STA STS            *SET UP WITH CLEAR SCREEN
000A 21 11 00           0240  LXI H,LOOP         *AND CURSOR AT UPPER RIGHT
000D E5                 0250  PUSH H             *USER MUST DEFINE OWN STACK AR
000E C3 65 1D           0260  JMP FF
0011 FB                 0310 LOOP EI
0012 C3 11 00           0320  JMP LOOP
0015                    0330  ORG 38H            *RESTART 7
0038 DB 88              0340 IN IN KBD           *INTERRUPT DRIVEN KEYBOARD
003A F6 80              0345  ORI 80H
003C F6 80              0350  ORI 80H
003E 47                 0360  MOV C,A
003F CD 00 1D           0370  CALL OUT
0042 78                 0380  MOV A,C
0043 C9                 0400  RET
0044                    0500  ORG 1D00H
1D00 2A FC 1C           1000 OUT LHLD CURS
1D03 EB                 1010  XCHG               *PUT RELATIVE CURSOR IN D
1D04 21 00 88           1020  LXI H,SCRN         *PUT SCREEN BLOCK ADDRESS IN H
1D07 19                 1030  DAD D              *GET ABS CURSOR LOCATION
1D08 47                 1040  MOV B,A
1D09 3A FF 1C           1050  LDA STR
1D0C 77                 1060  MOV M,A            *PUT BACK CHAR UNDER CURSOR
1D0D 78                 1070  MOV A,B                  *CHECK*
1D0E FE 88              1100  CPI 88H            *CTL H FOR HOME
1D10 CA 5C 1D           1110  JZ HOME
1D13 FE 85              1120  CPI 85H            *CTL E FOR ERASE
1D15 CA 65 1D           1130  JZ FF
1D18 FE 92              1140  CPI 92H            *CTL R FOR RIGHT
1D1A CA 74 1D           1150  JZ HT
1D1D FE 95              1160  CPI 95H            *CTL U FOR UP
1D1F CA 7C 1D           1170  JZ VT
1D22 FE 8C              1180  CPI 8CH            *CTL L FOR LEFT
1D24 CA 91 1D           1190  JZ BS
1D27 FE 84              1192  CPI 84H            *CTL D FOR DOWN
1D29 CA E8 1D           1194  JZ LF
1D2C FE 98              1200  CPI 98H            *CTL X (DELETE CHAR)
1D2E CA 99 1D           1210  JZ RO
```

```
1D31 FE 89            CPI 89H         *CTL I FOR INSERT (SET I/D)
1D33 CA 86 1D         JZ SID
1D36 FE 94            CPI 94H         *CTL T FOR TEXT (X I/D)
1D39 CA B1 1D         JZ RID
1D3B FE 86            CPI 86H         *CTL F FOR FEED (SET ALF)
1D3D CA BC 1D         JZ SALF
1D40 FE 8E            CPI 8EH         *CTL N FOR NORMAL TTY (X ALF
1D42 CA C7 1D         JZ RALF
1D45 FE 93            CPI 93H         *CTL S FOR SCROLL (SET SCRL)
1D47 CA D2 1D         JZ SSC
1D4A FE 90            CPI 90H         *CTL P FOR PAGE (X SCRL)
1D4C CA DD 1D         JZ RSC
1D4F FE 8A            CPI 8AH         *LINE FEED
1D51 CA E8 1D         JZ LF
1D54 FE 8D            CPI 8DH         *CARRIAGE RETURN
1D56 CA 21 1E         JZ CR
1D59 C3 45 1E         JMP DEF         *ANY OTHER CHARACTER
1D5C 21 00 00        HOME LXI H,0     *HOME CURSOR
1D5F 22 FC 1C         SHLD CURS
1D62 C3 6F 1E         JMP OUT1
1D65 21 00 88        FF LXI H,SCRN    *FORM FEED
1D68 36 3F           WIPE MVI M,LT    *LINE TERMINATION CHAR 7FH
1D6A 23               INX H
1D6B 7C               MOV A,H
1D6C FE 8C            CPI SEND        *SCREEN END?
1D6E C2 68 1D         JNZ WIPE
1D71 C3 5C 1D         JMP HOME        *CLEAR, GO HOME
1D74 13              HT INX D         *CURSOR RIGHT
1D75 EB               XCHG
1D76 22 FC 1C         SHLD CURS
1D79 C3 6F 1E         JMP OUT1
1D7C 21 C0 FF        VT LXI H,0-LINE  *CURSOR UP
1D7F 19               DAD D
1D80 22 FC 1C         SHLD CURS
1D83 C3 6F 1E         JMP OUT1
1D86 3A FE 1C        SID LDA STS      *SET I/D MODE
1D89 F6 01            ORI 01H         *RIGHT BIT =1
1D8B 32 FE 1C         STA STS
1D8E C3 6F 1E         JMP OUT1
1D91 1B              BS DCX D         *CURSOR LEFT
1D92 EB               XCHG
1D93 22 FC 1C         SHLD CURS
1D96 C3 6F 1E         JMP OUT1
1D99 3A FE 1C        RO LDA STS       *RUB OUT IF I/D SET
1D9C 1F               RAR
1D9D D2 91 1D         JNC BS
1DA0 23              SWAP INX H       *DEL CHAR, SWAP LINE IN
1DA1 7E               MOV A,M
1DA2 2B               DCX H
1DA3 77               MOV M,A
1DA4 23               INX H
1DA5 7D               MOV A,L
1DA6 E6 3F            ANI 3FH
```

```
1DA8 C2 A0 1D    2370  JNZ SWAP
1DAB 2B          2380  DCX H
1DAC 36 7F       2390  MVI M,7FH
1DAE C3 6F 1E    2400  JMP OUT1
1DB1 3A FE 1C    2410 RID LDA STS       *RESET I/D MODE
1DB4 E6 FE       2420  ANI 0FEH         *RIGHT BIT =0
1DB6 32 FE 1C    2430  STA STS
1DB9 C3 6F 1E    2440  JMP OUT1
1DBC 3A FE 1C    2450 SALF LDA STS      *SET ALF MODE
1DBF F6 40       2460  ORI 40H          *2ND BIT LEFT =1
1DC1 32 FE 1C    2470  STA STS
1DC4 C3 6F 1E    2480  JMP OUT1
1DC7 3A FE 1C    2482 RALF LDA STS      *RESET ALF MODE
1DCA E6 BF       2484  ANI 0BFH         *2ND BIT LEFT =0
1DCC 32 FE 1C    2486  STA STS
1DCF C3 6F 1E    2488  JMP OUT1
1DD2 3A FE 1C    2490 SSC LDA STS       *SET SCROLL MODE
1DD5 F6 80       2500  ORI 80H          *LEFT BIT =1
1DD7 32 FE 1C    2510  STA STS
1DDA C3 6F 1E    2520  JMP OUT1
1DDD 3A FE 1C    2530 RSC LDA STS       *RESET SCROLL MODE
1DE0 E6 7F       2540  ANI 7FH          *LEFT BIT =0
1DE2 32 FE 1C    2550  STA STS
1DE5 C3 6F 1E    2560  JMP OUT1
1DE8 21 40 00    2570 LF LXI H,64       *LINE FEED
1DEB 19          2580  DAD D            *ADD 64 TO REL CURSOR
1DEC 3A FE 1C    2590  LDA STS
1DEF 17          2600  RAL
1DF0 DC F9 1D    2610  CC SCRL          *CHECK SCROLL
1DF3 22 FC 1C    2620  SHLD CURS        *UPDATE CURSOR LOCATION
1DF6 C3 6F 1E    2630  JMP OUT1
1DF9 7C          2640 SCRL MOV A,H      *SCROLL ROUTINE
1DFA FE 04       2650  CPI 4            *OFF PAGE?
1DFC D8          2660  RC               *IF NOT, DO NOTHING
1DFD E5          2670  PUSH H
1DFE 11 00 88    2680  LXI D,SCRN       *TAKE IT FROM THE TOP
1E01 21 40 88    2700  LXI H,SCRN+LINE
1E04 7E          2710 SWP MOV A,M       *GRAB CHARACTER
1E05 23          2720  INX H
1E06 EB          2730  XCHG             *GET ADDRESS ONE LINE UP
1E07 77          2740  MOV M,A          *PUT CHARACTER THERE
1E08 23          2760  INX H
1E09 EB          2770  XCHG
1E0A 7C          2780  MOV A,H
1E0B FE 8C       2800  CPI SEND         *SCREEN FINISHED?
1E0D C2 04 1E    2810  JNZ SWP          *TAKE NEXT CHAR IF NOT
1E10 EB          2812  XCHG
1E11 06 3F       2814  MVI B,LT         *BLANK LAST LINE
1E13 70          2816 LAST MOV M,B
1E14 23          2820  INX H
1E15 7D          2830  MOV A,L
1E16 FE 00       2840  CPI 0
1E18 C2 13 1E    2850  JNZ LAST
```

```
1E1B E1          2860  POP H             *GET BACK REL CURSOR
1E1C 11 C0 FF    2862  LXI D,0-LINE
1E1F 19          2864  DAD D             *MOVE UP ONE LINE
1E20 C9          2870  RET
1E21 3A FE 1C    2890 CR LDA STS         *CARRIAGE RETURN
1E24 1F          2900  RAR
1E25 DA 32 1E    2910  JC BACK           *INSERT/DELETE? IF SO, DON'
1E28 36 3F       2920 SLOP MVI M,LT      *SCRATCH END OF LINE
1E2A 23          2930  INX H
1E2B 3E 3F       2940  MVI A,3FH         *MAKE 1FH FOR 32 CHAR LINE
1E2D A5          2950  ANA L
1E2E C2 28 1E    2960  JNZ SLOP
1E31 2B          2970  DCX H
1E32 3E C0       2980 BACK MVI A,0C0H    *GO TO BEGINNING OF LINE
1E34 A3          2990  ANA E
1E35 5F          3000  MOV E,A
1E36 3A FE 1C    3020  LDA STS
1E39 17          3030  RAL
1E3A 17          3040  RAL
1E3B DA E8 1D    3050  JC LF             *CHECK AUTO LINE FEED
1E3E EB          3052  XCHG
1E3F 22 FC 1C    3055  SHLD CURS
1E42 C3 6F 1E    3060  JMP OUT1
1E45 3A FE 1C    4000 DEF LDA STS        *DEFAULT ROUTINE, CHECK /I
1E48 1F          4010  RAR
1E49 DC 5C 1E    4020  CC INSR           *INSERT IF NOTED
1E4C 70          4030  MOV M,B           *STUFF CHARACTER
1E4D 13          4040  INX D             *INCREMENT CURSOR
1E4E EB          4050  XCHG
1E4F 3A FE 1C    4060  LDA STS
1E52 17          4070  RAL
1E53 DC F9 1D    4080  CC SCRL           *CHECK SCROLL
1E56 22 FC 1C    4090  SHLD CURS         *UPDATE CURSOR
1E59 C3 6F 1E    4100  JMP OUT1
1E5C E5          4200 INSR PUSH H        *MAKE SPACE FOR INSERT
1E5D 7E          4210  MOV A,M
1E5E 3A FF 1C    4220  LDA STR
1E61 77          4230  MOV M,A           *REPLACE CHAR UNDER CURSOR
1E62 23          4240 SHFT INX H         *MOVE LINE OUT
1E63 4E          4250  MOV C,M
1E64 77          4260  MOV M,A
1E65 3E 3F       4270  MVI A,3FH
1E67 A5          4280  ANA L
1E68 79          4290  MOV A,C
1E69 C2 62 1E    4300  JNZ SHFT
1E6C 77          4310  MOV M,A
1E6D E1          4320  POP H
1E6E C9          4330  RET
1E6F 2A FC 1C    8000 OUT1 LHLD CURS     *KEEP CURSOR ON SCREEN
1E72 7C          8010  MOV A,H
1E73 E6 03       8020  ANI 3
```

```
1E75 67           8030  MOV H,A
1E76 22 FC 1C     8040  SHLD CURS
1E79 11 00 88     8060  LXI D,SCRN         *INDEX BY SCREEN ADDRESS
1E7C 19           8070  DAD D
1E7D 7E           8080  MOV A,M            *STORE CHAR UNDER CURSOR
1E7E 32 FF 1C     8090  STA STR
1E81 36 FF        8100  MVI M,CS           *STUFF NEW CURSOR SYMBOL
1E83 C9           8110  RET
```

For users of PolyMorphic monitor ROM.

```
                ; *****VIDEO TYPEWRITER ROUTINE*****
                ; Note:  Poly 88 version, using wormhole
                ; The video typewriter routine allows the user to
                ; put ASCII characters from the keyboard onto the
                ; monitor screen.
                ;
0C20            CO      EQU     0C20H
F800            SCRN    EQU     0F800H  ; first address of screen memory
00FC            SEND    EQU     0FCH    ; end addrs of screen memory
0040            LINE    EQU     64D     ; length of line on screen
00FF            CS      EQU     0FFH    ; cursor symbol (rubout)
003F            LT      EQU     3FH     ; blank
00F8            KBD     EQU     0F8H    ; keyboard port location
2000                    ORG     2000H
2000 000000             NOP
2003 00                 NOP
2004 000000             NOP
2007 00                 NOP
2008 CD5120             CALL    FF      ; clear screen
200B CD200C     GET:    CALL    CO
200E 00                 NOP
200F F680               ORI     80H     ; to make char ASCII, not grphic
2011 CD1920             CALL    PUT     ; char--then put on screen
2014 00                 NOP
2015 78                 MOV     A,B
2016 C30B20             JMP     GET
2019 2AE020     PUT:    LHLD    CURS    ; put rel cursor position in D
201C EB                 XCHG
201D 2100F8             LXI     H,SCRN  ; add to first addrs of screen
2020 19                 DAD     D       ; memory
2021 00                 NOP
2022 47                 MOV     B,A
2023 00                 NOP
2024 3ADF20             LDA     STR     ; put old char that was under
2027 77                 MOV     M,A     ; cursor up on screen
2028 78                 MOV     A,B     ; now process new char--
2029 FE88               CPI     88H     ; Control-H char? If yes,
202B 00                 NOP
202C CA6020             JZ      HOME    ; home up screen
202F FE85               CPI     85H     ; Control-E? If yes, form feed
2031 00                 NOP
2032 CA5120             JZ      FF      ; to erase screen
2035 FE8C               CPI     8CH     ; Control-L? If yes, backspace
2037 00                 NOP
2038 CA6920             JZ      BS
203B FE8D               CPI     8DH     ; carriage return? If yes, go to
203D 00                 NOP
203E CAA620             JZ      CR      ; CR routine.
2041 00                 NOP
2042 00                 NOP
2043 70         DEF:    MOV     M,B     ; increment cursor position
2044 13                 INX     D
2045 EB                 XCHG
2046 CD7220             CALL    SCRL    ; scroll screen
2049 22E020             SHLD    CURS
204C 00                 NOP
204D C3C520             JMP     OUT1
2050 00                 NOP
2051 2100F8     FF:     LXI     H.SCRN  ; form feed to clear screen
```

```
2054 363F     WIPE:   MVI     M,LT
2056 23               INX     H
2057 7C               MOV     A,H
2058 00               NOP
2059 FEFC             CPI     SEND      ; end of screen?
205B 00               NOP
205C C25420           JNZ     WIPE      ; if not, keep going w/ blanks
205F 00               NOP
2060 210000   HOME:   LXI     H,0
2063 22E020           SHLD    CURS
2066 C3C520           JMP     OUT1
2069 1B       BS:     DCX     D
206A EB               XCHG
206B 22E020           SHLD    CURS
206E C3C520           JMP     OUT1
2071 00               NOP
2072 7C       SCRL:   MOV     A,H
2073 FE04             CPI     4         ; if top half of rel curs pos<4,
2075 00               NOP
2076 D8               RC                ; dont scroll, 'cause not at end
2077 E5               PUSH    H         ; of screen
2078 1100F8           LXI     D,SCRN    ; to scroll, move down a line.
207B 2140F8           LXI     H,SCRN+LINE
207E 00               NOP
207F 7E       SWP:    MOV     A,M       ; save char at that point
2080 23               INX     H
2081 EB               XCHG
2082 77               MOV     M,A       ; put char on screen
2083 23               INX     H
2084 7C               MOV     A,H
2085 EB               XCHG
2086 00               NOP
2087 FEFB             CPI     SEND-1    ; at last quadrant of screen?
2089 00               NOP
208A C27F20           JNZ     SWP
208D 7B               MOV     A,E       ; see if at screen end-64
208E 00               NOP
208F FEC0             CPI     0C0H      ; (beginning of last line)
2091 EB               XCHG
2092 C27F20           JNZ     SWP
2095 063F             MVI     B,LT
2097 00               NOP
2098 70       LAST:   MOV     M,B       ; put blank at end of screen
2099 23               INX     H
209A 7D               MOV     A,L
209B B7               ORA     A
209C C29820           JNZ     LAST
209F E1               POP     H         ; get back rel cursor position
20A0 11C0FF           LXI     D,0-LINE
20A3 19               DAD     D         ; move up one line
20A4 C9               RET
20A5 00               NOP
20A6 363F     CR:     MVI     M,LT
20A8 23               INX     H
20A9 3E3F             MVI     A,3FH
20AB A5               ANA     L
20AC 00               NOP
20AD C2A620           JNZ     CR
20B0 2B               DCX     H
20B1 00               NOP
```

```
20B2 3EC0      BACK:   MVI     A,0C0H    ; go to beginning of line
20B4 A3                ANA     E
20B5 5F                MOV     E,A
20B6 00                NOP
20B7 214000    LF:     LXI     H,64
20BA 19                DAD     D
20BB CD7220            CALL    SCRL
20BE 22E020            SHLD    CURS
20C1 00                NOP
20C2 C3C520            JMP     OUT1
20C5 2AE020    OUT1:   LHLD    CURS
20C8 00                NOP
20C9 7C                MOV     A,H
20CA E603              ANI     3         ; keep cursor on screen
20CC 00                NOP
20CD 67                MOV     H,A
20CE 22E020            SHLD    CURS
20D1 00                NOP
20D2 1100F8            LXI     D,SCRN
20D5 19                DAD     D
20D6 7E                MOV     A,M       ; save char where going to put
20D7 00                NOP
20D8 32DF20            STA     STR       ; cursor--store char in STR
20DB 36FF              MVI     M,CS      ; put cursor symbol on screen
20DD 00                NOP
20DE C9                RET
20DF           STR:    DS      1         ; char under cursor
20E0           CURS:   DS      2         ; rel position of cursor
0000                   END
```

```
                 ;*****************************************************
                 ;*                                                   *
                 ;*            VIDEO TYPEWRITER ROUTINE               *
                 ;*                                                   *
                 ;*    The video typewriter routine allows you to     *
                 ;*    put ASCII-character input from the keyboard    *
                 ;*    onto the monitor screen.                       *
                 ;*                                                   *
                 ;*****************************************************
                 ;
     F800        SCRN    EQU     0F800H   ; first addrs of screen memory
     00FC        SEND    EQU     0FCH     ; end addrs of screen memory
     0040        LINE    EQU     64D      ; length of line on screen
     00FF        CS      EQU     0FFH     ; cursor symbol (rubout)
     003F        LT      EQU     3FH      ; blank
     00F8        KBD     EQU     0F8H     ; keyboard port location
                         IDNT    0,0
     0000                ORG     000
0000 C3001D              JMP     START
                 ;
                 ; when keyboard interrupt received, location 38H jumped to.
                 ;
                 ;       ORG      038H    ; when you get interrupt, get char from
0038 DBF8        GET:    IN      KBD      ; KBD and put in A. ORI w/ 80H
003A F680                ORI     80H      ; to make char ASCII, not graphics
003C CD0B1D              CALL    PUT      ; char--then put on screen
003F C3061D              JMP     LOOP     ; get another char
                 ;
     1D00                ORG     1D00H
1D00 31001D      START:  LXI     SP,1D00H
                 ;
                 ; initialize stack pointer.  NOTE: Some operating systems,
                 ; like the POLY 4.0 monitor, initialize stack pointer.  If
                 ; this is so with your system, eliminate this instruction by
                 ; changing all 3 bytes to zeros.
                 ;
1D03 CD391D              CALL    FF
                 ;
                 ; Enable interrupts--when keyboard interrupt received,
                 ; location 38H jumped to.
                 ;
1D06 FB          LOOP:   EI
1D07 76                  HLT
1D08 C3061D              JMP     LOOP
1D0B 2AB41D      PUT:    LHLD    CURS     ; put rel cursor position in D
1D0E EB                  XCHG
1D0F 2100F8              LXI     H,SCRN   ; add to first addrs of screen
1D12 19                  DAD     D        ; memory
1D13 47                  MOV     B,A
1D14 3AB31D              LDA     STR      ; put old char that was under
1D17 77                  MOV     M,A      ; cursor up on screen
1D18 78                  MOV     A,B      ; now process new char--
1D19 FE88                CPI     88H      ; Control-H char? If yes,
1D1B CA451D              JZ      HOME     ; home up screen
1D1E FE85                CPI     85H      ; Control-E? If yes, form feed
1D20 CA391D              JZ      FF       ; to erase screen
1D23 FE8C                CPI     8CH      ; Control-L? If yes, backspace
1D25 CA4E1D              JZ      BS       ; (move cursor left)
1D28 FE8D                CPI     8DH      ; carriage return? If yes, go to
```

```
1D2A CA831D            JZ      CR       ; CR routine.
1D2D 70       DEF:     MOV     M,B      ; increment cursor position
1D2E 13                INX     D
1D2F EB                XCHG
1D30 CD561D            CALL    SCRL     ; scroll screen
1D33 22B41D            SHLD    CURS
1D36 C39E1D            JMP     OUT1
1D39 2100F8   FF:      LXI     H,SCRN   ; form feed to clear screen
1D3C 363F     WIPE:    MVI     M,LT
1D3E 23                INX     H
1D3F 7C                MOV     A,H
1D40 FEFC              CPI     SEND     ; end of screen?
1D42 C23C1D            JNZ     WIPE     ; if not, keep going w/ blanks
1D45 210000   HOME:    LXI     H,0      ; home up screen by
1D48 22B41D            SHLD    CURS     ; reinitializing rel curs pos to
1D4B C39E1D            JMP     OUT1     ; zero. Then put on screen
1D4E 1B       BS:      DCX     D        ; back space
1D4F EB                XCHG
1D50 22B41D            SHLD    CURS
1D53 C39E1D            JMP     OUT1
1D56 7C       SCRL:    MOV     A,H
1D57 FE04              CPI     4        ; if top half of rel curs pos<4,
1D59 D8                RC               ; not end of screen: dont scroll
1D5A E5                PUSH    H
1D5B 1100F8            LXI     D,SCRN   ; to scroll, move down a line.
1D5E 2140F8            LXI     H,SCRN+LINE
1D61 7E       SWP:     MOV     A,M      ; save char at that point
1D62 23                INX     H
1D63 EB                XCHG
1D64 77                MOV     M,A      ; put char on screen
1D65 23                INX     H        ; one line up
1D66 7C                MOV     A,H
1D67 EB                XCHG
1D68 FEFB              CPI     SEND-1   ; last quadrant of screen?
1D6A C2611D            JNZ     SWP      ; if not, keep going
1D6D 7B                MOV     A,E      ; see if at screen end-64
1D6E FEC0              CPI     0C0H     ; (beginning of last line)
1D70 C2611D            JNZ     SWP
1D73 EB                XCHG
1D74 063F              MVI     B,LT
1D76 70       LAST:    MOV     M,B
1D77 23                INX     H
1D78 7D                MOV     A,L
1D79 B7                ORA     A
1D7A C2761D            JNZ     LAST
1D7D E1                POP     H        ; get back rel cursor position
1D7E 11C0FF            LXI     D,0-LINE
1D81 19                DAD     D        ; move up one line
1D82 C9                RET
1D83 363F     CR:      MVI     M,LT
1D85 23                INX     H
1D86 3E3F              MVI     A,3FH
1D88 A5                ANA     L
1D89 C2831D            JNZ     CR
1D8C 2B                DCX     H
1D8D 3EC0     BACK:    MVI     A,0C0H   ; go to beginning of line
1D8F A3                ANA     E
1D90 5F                MOV     E,A
1D91 214000   LF:      LXI     H,64
1D94 19                DAD     D
```

```
1D95 CD561D              CALL    SCRL
1D98 22B41D              SHLD    CURS
1D9B C39E1D              JMP     OUT1
1D9E 2AB41D    OUT1:     LHLD    CURS
1DA1 7C                  MOV     A,H
1DA2 E6Ø3                ANI     3          ; keep cursor on screen by
1DA4 67                  MOV     H,A        ; keeping least 2 significant
1DA5 22B41D              SHLD    CURS       ; bits of most significant byte
1DA8 11ØØF8              LXI     D,SCRN
1DAB 19                  DAD     D
1DAC 7E                  MOV     A,M
1DAD 32B31D              STA     STR
1DBØ 36FF                MVI     M,CS
1DB2 C9                  RET
               ;
1DB3           STR:      DS      1          ; char under cursor
1DB4           CURS:     DS      2          ; rel position of cursor
```

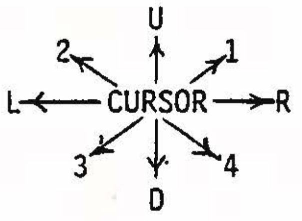
For Users of the PolyMorphic Monitor ROM:

VIDEO GRAPHICS CURSOR WITH LIFE

Requires 4K of RAM Beginning at 2ØØØ Hex

To use this software, execute 2000 hex. You are now in the "move" mode. Use the cursor control commands to place the cursor at a desirable starting location. Use the "W" printing control once to allow the deposit of white squares. Use the cursor control commands to "draw" on the screen. If you want to erase a square or squares, depress the "B" key once and manipulate the cursor across the square you wish to erase. The "B" command actually prints black squares, so don't cross white squares you want to keep. If you want to move the cursor without affecting the display, press the "M" key once.

KEYBOARD COMMANDS
FOR CURSOR CONTROL



KEYBOARD COMMANDS
FOR PRINTING CONTROL

M = Move Cursor

W = Print White Squares

B = Print Black Squares
(or erase white squares)

The cursor control commands can be used in succession -- for example, pressing the "U" key five times will "draw" a white line upward from the original cursor location.

When you wish to execute LIFE on the present graphics display, press "X". The number of generations will be displayed in the upper-right corner of the screen.

If you wish to exit LIFE and modify its present state, press "E". You are now back to the original graphics cursor mode with the last LIFE generation intact. You can modify the existing pattern and execute LIFE with the resulting pattern.

LIFE WItH GRAPHICS CURSOR FOR 4.0 MONITOR ROM

3 sectors from 0 to 3000

Life - with changes

life3 - original values

```
 2000                  ORG     32000H
 2E00        adr TADD1  EQU    32E00H
 0C20          WH0      EQU     0C20H
 002E          TAD1     EQU 1800H 2EH
 F800          VADD EQU 0F800H ;VIDEO BLOCK ADDRESS
 0C0C          KBUF EQU 0C0CH  ;KEYBOARD INPUT BUFFER ADDRESS
 2400      Adr MADD EQU 32400H      ;MASTER COPY ADDRESS
 2900      Adr SADD EQU 32900H      ;SLAVE COPY ADDRESS
 0024          MAD EQU 24H          ;1ST BYTE OF MADD
 0029          SAD EQU 29H          ;1ST BYTE OF SADD
 0040          LINE EQU 64          ;LINE LENGTH
 2308      adr TADD EQU 32308H      ;TABLE (MASK & SCRATCH)
 0023          TAD EQU 23H          ;1ST BYTE OF TADD
 2280      adr CADD EQU 32280H      ;COUNT ADDRESS (GENERATIONS)
 2000 C31E21    JMP SETUP adr            ;START OF GRAPHICS CURSOR
 2003 210823    LXI H,TADD d16      ;SET UP MASK TABLE
 2006 3E20      MVI A,20H d8        ;FIRST MASK FOR TABLE
 2008 0E08     MASK:MVI C,08H d8    ;GETS EIGHT SPOTS
 200A 77       TABLE:MOV M,A
 200B 23        INX H
 200C 0D        DCR C
 200D C20A20    JNZ TABLE adr       ;IN TABLE.
 2010 0F        RRC                 ;THEN MASK FOR NEXT LOWER BIT
 2011 D20820    JNC MASK adr        ;GETS THE NEXT EIGHT.
 2014 210029    LXI H,SADD d16      ;SAVE SLAVE ADDRESS
 2017 E5        PUSH H              ;FOR USE IN LOOP.
 2018 218022    LXI H,CADD d16      ;LOAD CADD WITH OWN
 201B 3680      MVI M,80H d8        ;SECOND BYTE TO START COUNT.
 201D 21C0F7    LXI H,VADD-40H d16  ;SET UP FOR SWAP FROM
 2020 11C028    LXI D,SADD-40H d16  ;SCREEN TO SLAVE WITH SLOP.
 2023 C3DC20   LOOP:    JMP     KBDTST adr
 2026 7E       CONTIN:  MOV     A,M
 2027 2F        CMA                 ;COMPLEMENT FOR TRUE LIFE
 2028 12        STAX D              ;STORE ON OTHER COPY.
 2029 23        INX H               ;NEXT
 202A 13        INX D               ;SPOT.
 202B 7C        MOV A,H             ;CHECK
 202C E607      ANI 7 d8            ;LAST THREE BITS OF 1ST BYTE
 202E FE05      CPI 5 d8            ;FOR END
 2030 C22320    JNZ LOOP adr        ;OF COPY PLUS SLOP.
 2033 21C028    LXI H,SADD-40H d16  ;SWAP SLAVE
 2036 11C023    LXI D,MADD-40H d16  ;TO MASTER
 2039 7E       SWAP:MOV A,M
 203A 12        STAX D
 203B 23        INX H
 203C 13        INX D
 203D 7C        MOV A,H
 203E FE2E      CPI SAD+5 d8        ;WITH SLOP
 2040 C23920    JNZ SWAP adr        ;UP TO HERE.
 2043 118022    LXI D,CADD d16      ;SET UP FOR COUNT
 2046 0128F8    LXI B,VADD+28H d16  ;IN UPPER RIGHT OF SCREEN
 2049 218022    LXI H,CADD d16      ;WATCH THE ZERO AND CARRY!!
 204C 6B          MOV L,E
 204D 23         COUNT:INX H              ;NEXT SIGNIFICANT DIGIT
 204E 0B          DCX B                   ;NEXT DOWN ON SCREEN
 204F C25320      JNZ NOINC adr X U       ;ZERO FLAG TO INCREMENT
 2052 34           INR M
 2053 1A          NOINC:LDAX D  X O        ;ARE WE TO END
 2054 BD           CMP L                   ;OF COUNT (STORED AT CADD)?
 2055 DA5E20       JC REP    Adr X U       ;YES
 2058 3EBA         MVI A,0BAH  d8          ;NO, CHECK FOR
 205A BE           CMP M                   ;DECIMAL CARRY IN ASCII.
 205B C26120       JNZ HERE   adr X U      ;NO
 205E 3EB0         REP:MVI A,0B0H d8 X O   ;YES, ZERO THAT DIGIT
 2060 77           MOV M,A                 ;AND REPLACE MEMORY.
 2061 7E          HERE:MOV A,M   X O       ;GET MEMORY
 2062 02           STAX B                  ;AND VIEW IT
 2063 D24D20       JNC COUNT               ;UNTIL ALL DIGITS ARE VIEWED.
 2066 2B           DCX H                   ;CHECK MOST SIGNIFICANT DIGIT
 2067 BE           CMP M                   ;AGAINST NEXT MOST.
 2068 CA6D20       JZ THERE      X U       ;BOTH ZERO? EXIT.
 206B EB           XCHG                    ;NO, INCREASE
 206C 34           INR M            →      ;END OF COUNT.
 206D 21BF23      THERE:LXI H,MADD-LINE-1 X O
 2070 1623         MVI D,TAD               ;GET IN POSITION FOR TABLE.
 2072 01ED20      BYTE:LXI B,INST  X O     ;PSEUDO OP LIST
 2075 0A          BIT:LDAX B      X O      ;LOAD PSEUDO OP.
 2076 0F           RRC                     ;CHECK RIGHT BIT FOR
 2077 D28D20       JNC ROT       X U       ;CELL CHECK FROM SAME BYTE
 207A 0F           RRC                     ;NO. NEXT BYTE?
 207B D28920       JNC ONE       X U       ;YES
 207E FEF0         CPI 0F0H                ;NO. ALL NGHBRS DONE THIS BYTE?
 2080 D29F20       JNC DONE      X U       ;YES.
 2083 113D00       LXI D,LINE-3            ;NEXT LINE ON 3X3 MATRIX
 2086 19           DAD D                   ;INCREMENT BY LINE-2
 2087 1623         MVI D,TAD               ;BY LINE-3+1, SINCE WE NEED
 2089 23          ONE:INX H      X O       ;A +1 ANYWAY
 208A E63F         ANI 3FH                 ;GET RID OF 2 MSB'S.
 208C 07           RLC                     ;ZERO CARRY BIT AND
 208D 1F          ROT:RAR       X O        ;GET IN POSITION
 208E 03           INX B                   ;FOR THIS AND NEXT PSEUDO OP
 208F 5F           MOV E,A                 ;2ND BYTE FEEDS MASK TABLE
 2090 1A           LDAX D                  ;LOAD MASK FOR BIT
 2091 A6           ANA M                   ;AND CHECK IT ON THE MASTER
 2092 CA7520       JZ BIT        X U       ;NO LIFE, NEXT BIT
 2095 EB           XCHG                    ;BRING DOWN SCRATCH
 2096 3E07         MVI A,07H               ;ADDRESS TO STORE NEIGHBOR
 2098 A5           ANA L                   ;COUNT CODED BY BIT #
 2099 6F           MOV L,A
 209A 34           INR M                   ;COUNT ONE NEIGHBOR
 209B EB           XCHG                    ;GET MASTER COPY
 209C C37520       JMP BIT       X U       ;AND GET NEXT BIT IN BYTE
 209F 01BFFF      DONE:LXI B,0-LINE-1 X O ;GO BACK TO BYTE
 20A2 09           DAD B                   ;THAT WE'RE WORKING ON
 20A3 1E00         MVI E,0                 ;ZERO SCRATCHPAD BYTE #2
 20A5 E3           XTHL                    ;MOVING ON TO SLAVE COPY.
 20A6 97          LOAD:SUB A               ;ZERO A SO WE CAN
 20A7 12           STAX D                  ;ZERO NEIGHBOR COUNT
 20A8 79           MOV A,C                 ;GET INVERTED BIT MASK
```

LIFE WItH GRAPHICS CURSOR FOR 4.0 MONITOR ROM

```
20A9 0F         RRC                     ;COMING IN BFH AND ROTATE
20AA D2C520     JNC NEXT                ;GOT ALL BITS?
20AD 4F         MOV C,A                 ;NO, REPLACE MASK
20AE 1C         INR E                   ;AND COUNT BIT NUMBER
20AF 1A         LDAX D                  ;GET # NEIGHBORS OF THAT BIT
20B0 FE02       CPI 02H                 ;IS IT TWO?
20B2 CAA620     JZ LOAD                 ;YES, CELL STAYS THE WAY IT IS
20B5 79         MOV A,C                 ;NO, SO
20B6 A6         ANA M                   ;KILL CELL ON
20B7 77         MOV M,A                 ;SLAVE COPY
20B8 1A         LDAX D                  ;HOW MANY NHBRS AGAIN?
20B9 FE03       CPI 03H                 ;ARE THERE THREE?
20BB C2A620     JNZ LOAD   Xu           ;YES, GOOD WE KILLED IT.
20BE 79         MOV A,C                 ;OOPS, GOT TO RESURRECT IT
20BF 2F         CMA                     ;BY INVERTING THE MASK
20C0 86         ADD M                   ;AND ADDING
20C1 77         MOV M,A                 ;REPLACE SLAVE
20C2 C3A620     JMP LOAD    Xu          ;UPDATE NEXT BIT IN BYTE
20C5 01C0FF    NEXT:LXI B,0-LINE  Xo    ;UP ONE, WHICH IS UPPER
20C8 23         INX H                   ;INCREMENT SLAVE ADDRESS
20C9 E3         XTHL                    ;FOR PROPER INITIALIZATION
20CA 3E28       MVI A,MAD+04H           ;END OF SCREEN?
20CC BC         CMP H
20CD 09         DAD B                   ;COMPLETE ONE UP
20CE C27220     JNZ BYTE   Xu           ;SCREEN NOT OVER, NEXT BYTE
20D1 E1         POP H                   ;LEAVE
20D2 210029     LXI H,SADD              ;SADD ON STACK
20D5 E5         PUSH H                  ;FOR NEXT TIME. SET UP TO
20D6 1100F8     LXI D,VADD              ;SWAP SLAVE TO SCREEN
20D9 C32320     JMP LOOP                ;ON EACH SUCCESSIVE LOOP.
                                ;KBDTST CHECKS THE KEYBOARD FOR "E"
                                ;(EXIT COMMAND) AND IF SO JUMPS INTO
                                ;WARMSTART OF GRAPHICS CURSOR
20DC F5        KBDTST:  PUSH    PSW
20DD 3A0D0C             LDA     KBUF+1
20E0 FE45               CPI     'E'
20E2 C2E920             JNZ     OVER2
20E5 F1                 POP     PSW
20E6 C32B21             JMP     MODIFY
20E9 F1        OVER2:   POP     PSW
20EA C32620             JMP     CONTIN
20ED C465      INST:DW 65C4H  Xo        ;PSEUDO OPS CODE 48
20EF C470       DW 70C4H                ;SPECIAL CASES:EIGHT
20F1 D071       DW 71D0H                ;NEIGHBORS FOR EACH OF
20F3 87A4       DW 0A487H               ;SIX CELLS PER BYTE
20F5 88A8       DW 0A888H               ;RIGHT TWO BITS OF
20F7 C8AC       DW 0ACC8H               ;EACH PSEUDO OP INDICATE
20F9 CC45       DW 45CCH                ;WHETHER NEXT NEIGHBOR IS
20FB 84A4       DW 0A484H               ;IN THE SAME BYTE AS
20FD 2868       DW 6828H                ;CURRENT NEIGHBOR, OR IN
20FF 88A8       DW 0A888H               ;NEXT BYTE, OR NEXT LINE
2101 C84C       DW 4CC8H                ;IN 3X3 MATRIX OF
2103 ACCC       DW 0CCACH               ;NEIGHBOR BYTES
2105 3050       DW 5030H                ;NEXT THREE BITS CODE
2107 B034       DW 34B0H                ;CELL WHOSE NEIGHBORS
2109 5474       DW 7454H                ;WE ARE COUNTING, IN
210B 94D4       DW 0D494H               ;REVERSE ORDER
```

LIFE WItH GRAPHICS CURSOR FOR 4.0 MONITOR ROM

```
 210D 5878        DW 7858H              ;REMAINING THREE BITS
 210F B831        DW 31B8H              ;CODE MASK FOR NEIGHBOR
 2111 5034        DW 3450H              ;IN SAME FORMAT
 2113 5474        DW 7454H
 2115 5878        DW 7858H
 2117 8F2D        DW 2D8FH
 2119 8C38        DW 388CH
 211B 9839        DW 3998H
 211D FF          DB 0FFH               ;WRITTEN BY BRIAN WILCOX
 211E 2100F8     SETUP: LXI H,VADD
 2121 3E0C        MVI A,0CH
 2123 84          ADD H
 2124 367F        WIPE: MVI M,7FH     XO
 2126 23           INX H
 2127 BC           CMP H
 2128 C22421       JNZ WIPE        XU
 212B 0E06       MODIFY: MVI     C,6
 212D 11002E      LXI     D,TADD1  ———— TADD1
 2130 3E01         MVI A,1
 2132 12         TABL1:  STAX    D
 2133 07           RLC
 2134 13           INX D
 2135 0D           DCR C
 2136 C23221             JNZ     TABL1
 2139 2128FA       LXI H,VADD+0228H
 213C 0600         MVI B,0
 213E 367E         MVI M,7EH                                    WHO
 2140 CD200C     LOOP1:  CALL    WH0      ;KEYBOARD INPUT WORMHOLE
 2143 F680               ORI     80H      ;GETS COMMAND CHARACTER
 2145 CD4B21             CALL    OUT1
 2148 C34021             JMP     LOOP1    ;NEXT COMMAND
 214B 4F         OUT1:   MOV     C,A
 214C 7E           MOV A,M
 214D E640         ANI 40H
 214F CA5721       JZ COMP    XU
 2152 CD1522      CALL MASK1  XU
 2155 AE           XRA M
 2156 77           MOV M,A
                                ;COMMAND INTERPRETER
 2157 79          COMP: MOV A,C  XO
 2158 FED8         CPI 0D8H
 215A CA2422       JZ CLNO
 215D FED5         CPI 0D5H
 215F CCBA21       CZ UP      XU
 2162 FEC4         CPI 0C4H
 2164 CCC921       CZ DOWN     XU
 2167 FED2         CPI 0D2H
 2169 CCD821       CZ RIGHT    XU
 216C FECC         CPI 0CCH
 216E CCE021       CZ LEFT     XU
 2171 FED7         CPI 0D7H
 2173 CCE821       CZ WHITE    XU
 2176 FEC2         CPI 0C2H
 2178 CCED21       CZ BLACK    XU
 217B FECD         CPI 0CDH
 217D CCF421       CZ MOVE     XU
 2180 FEB1         CPI 0B1H
```

IFE WItH GRAPHICS CURSOR FOR 4.0 MONITOR ROM

```
 2182 CCF921     CZ ONE1        Xu
 2185 FEB2        CPI 0B2H
 2187 CC0022      CZ TWO        Xu
 218A FEB3        CPI 0B3H
 218C CC0722      CZ THREE      Xu
 218F FEB4        CPI 0B4H
 2191 CC0E22      CZ FOUR       Xu
 2194 CD1522     CALL MASK1     Xu
 2197 A6          ANA M
 2198 CAA321      JZ LITE       Xu
 219B 2F          CMA
 219C A6          ANA M
 219D F640        ORI 40H
 219F 77          MOV M,A
 21A0 C3A721      JMP BACK        Xu
 21A3 3EBF       LITE:  MVI A,0BFH Xo  — R     (MOV) ←
 21A5 A6          ANA M
 21A6 77          MOV M,A
 21A7 78         BACK: MOV A,B    Xo
 21A8 E660        ANI 60H
 21AA C8          RZ
 21AB E640        ANI 40H
 21AD C2B521      JNZ BRITE     Xu
 21B0 3E40        MVI A,40H
 21B2 B6          ORA M
 21B3 77          MOV M,A
 21B4 C9          RET
 21B5 3EBF       BRITE: MVI A,0BFH  Xo
 21B7 A6          ANA M
 21B8 77          MOV M,A
 21B9 C9          RET
 21BA 78         UP: MOV A,B      Xo
 21BB E602        ANI 2
 21BD C2C221            JNZ     HERE1  Xu
 21C0 04          INR B
 21C1 C9          RET                       DCR
 21C2 05        HERE1:  DCR     D     Xo
 21C3 05          DCR B
 21C4 11C0FF      LXI D,0-LINE
 21C7 19          DAD D
 21C8 C9          RET
 21C9 78         DOWN: MOV A,B      Xo
 21CA E603        ANI 3
 21CC CAD121            JZ      THER1  Xu
 21CF 05          DCR B
 21D0 C9          RET
 21D1 04        THER1:  INR     B     Xo
 21D2 04          INR B
 21D3 114000      LXI D,LINE
 21D6 19          DAD D
 21D7 C9          RET
 21D8 78         RIGHT: MOV A,B     Xo
 21D9 07          RLC
 21DA 3F          CMC
 21DB 1F          RAR
 21DC 47          MOV B,A
 21DD D8          RC
```

LIFE WItH GRAPHICS CURSOR FOR 4.0 MONITOR ROM

```
 21DE 23          INX H
 21DF C9          RET
 21E0 78        LEFT:  MOV A,B        Xo
 21E1 07          RLC
 21E2 3F          CMC
 21E3 1F          RAR
 21E4 47          MOV B,A
 21E5 D0          RNC
 21E6 2B          DCX H
 21E7 C9          RET
 21E8 78        WHITE: MOV A,B       Xo
 21E9 F660        ORI 60H
 21EB 47          MOV B,A
 21EC C9          RET
 21ED 78        BLACK: MOV A,B       Xo
 21EE F620        ORI 20H
 21F0 E6BF        ANI 0BFH
 21F2 47          MOV B,A                 MOV
 21F3 C9          RET
 21F4 78        MOVE: MOV A,B        Xo
 21F5 E69F        ANI 9FH
 21F7 47          MOV B,A
 21F8 C9          RET
 21F9 CDBA21   ONE1:   CALL    UP    Xo
 21FC CDD821      CALL RIGHT         Xo
 21FF C9          RET
 2200 CDBA21    TWO: CALL UP         Xo
 2203 CDE021      CALL LEFT          Xo
 2206 C9          RET
 2207 CDC921    THREE: CALL DOWN     Xo
 220A CDE021      CALL LEFT          Xo
 220D C9          RET
 220E CDC921    FOUR: CALL DOWN      Xo
 2211 CDD821      CALL RIGHT         Xo
 2214 C9          RET
 2215 78       MASK1:  MOV     A,B   Xo
 2216 07          RLC
 2217 1F          RAR
 2218 D21D22      JNC CLEAR          Xo
 221B C603        ADI 3
 221D E61F      CLEAR: ANI 1FH       Xo
 221F 162E      MVI D,TAD1                TAD1
 2221 5F          MOV E,A
 2222 1A          LDAX D
 2223 C9          RET
                                ;THIS ROUTINE KILLS THE GENERATION
                                ;COUNTER WHEN JUMPING INTO A MODIFIED
                                ;LIFE SO THE COUNTER DOSN'T BECOME LIFE
 2224 2100F8   CLNO:    LXI     H,VADD
 2227 7D       CLNO1:   MOV     A,L
 2228 FE40              CPI     40H      ;END OF FIRST LINE
 222A CA0320            JZ   3-5 2003H   ;BEGINING OF LIFE PROGRAM
 222D 7E                MOV     A,M
 222E E680              ANI     80H      ;IS IT GRAPHICS? IF N0
 2230 C43722            CNZ     BLANK    ;PUT IN GRAPHICS BLANK
 2233 23                INX     H
 2234 C32722            JMP     CLNO1
```

LIFE WItH GRAPHICS CURSOR FOR 4.0 MONITOR ROM

```
 2237 363F     BLANK:  MVI      M,3FH
  239 C9               RET
 0000                  END
```

For Non-Users of PolyMorphic Monitor ROM:

LIFE

```
0000                VADD EQU 8800H      *VIDEO BLOCK ADDRESS
0000                MADD EQU 0300H      *MASTER COPY ADDRESS
0000                SADD EQU 0800H      *SLAVE COPY ADDRESS
0000                MAD EQU 03H         *1ST BYTE OF MADD.
0000                SAD EQU 08H         *1ST BYTE OF SADD
0000                LINE EQU 64         *LINE LENGTH
0000                TADD EQU 0208H      *TABLE (MASK & SCRATCH)
0000                TAD EQU 02H         *1ST BYTE OF TADD
0000                CADD EQU 0180H      *COUNT ADDRESS (GENERATIONS)
0000 21 08 02        LXI H,TADD         *SET UP MASK TABLE
0003 3E 20           MVI A,20H          *FIRST MASK FOR TABLE
0005 0E 08          MASK MVI C,08H      *GETS EIGHT SPOTS
0007 77             TABLE MOV M,A
0008 23              INX H
0009 0D              DCR C
000A C2 07 00        JNZ TABLE          *IN TABLE.
000D 0F              RRC                *THEN MASK FOR NEXT LOWER BI
000E D2 05 00        JNC MASK           *GETS THE NEXT EIGHT.
0011 21 00 08        LXI H,SADD         *SAVE SLAVE ADDRESS
0014 E5              PUSH H             *FOR USE IN LOOP
0015 21 80 01        LXI H,CADD         *LOAD CADD WITH OWN
0018 36 80           MVI M,80H          *SECOND BYTE TO START COUN
001A 21 C0 87        LXI H,VADD-40H     *SET UP FOR SWAP FROM
001D 11 C0 07        LXI D,SADD-40H     *SCREEN TO SLAVE WITH SLOP.
0020 7E             LOOP MOV A,M        *GRAB CHAR, BEGIN MAIN LOOP
0021 2F              CMA                *COMPLEMENT FOR TRUE LIFE
0022 12              STAX D             *STORE ON OTHER COPY.
0023 23              INX H              *NEXT
0024 13              INX D              *SPOT.
0025 7C              MOV A,H            *CHECK
0026 E6 07           ANI 7              *LAST THREE BITS OF 1ST BYTE
0028 FE 05           CPI 5              *FOR END
002A C2 20 00        JNZ LOOP           *OF COPY PLUS SLOP.
002D 21 C0 07        LXI H,SADD-40H     *SWAP SLAVE
0030 11 C0 02        LXI D,MADD-40H     *TO MASTER
0033 7E             SWAP MOV A,M
0034 12              STAX D
0035 23              INX H
0036 13              INX D
0037 7C              MOV A,H
0038 FE 0D           CPI SAD+5          *WITH SLOP
003A C2 33 00        JNZ SWAP           *UP TO HERE.
003D 11 80 01        LXI D,CADD         *SET UP FOR COUNT
0040 01 40 88        LXI B,VADD+40H     *IN UPPER RIGHT OF SCREEN
0043 21 80 01        LXI H,CADD         *WATCH THE ZERO AND CARRY
0046 6B              MOV L,E
```

```
0047 23          COUNT INX H        *NEXT SIGNIFICANT DIGIT
0048 0B           DCX B             *NEXT DOWN ON SCREEN
0049 C2 4D 00     JNZ NOINC         *ZERO FLAG TO INCREMENT
004C 34           INR M
004D 1A          NOINC LDAX D       *ARE WE TO END
004E BD           CMP L             *OF COUNT (STORED AT CADD)?
004F DA 58 00     JC OUT            *YES
0052 3E BA        MVI A,0BAH        *NO, CHECK FOR
0054 BE           CMP M             *DECIMAL CARRY IN ASCII.
0055 C2 5B 00     JNZ HERE          *NO
0058 3E B0       OUT MVI A,0B0H     *YES, ZERO THAT DIGIT
005A 77           MOV M,A           *AND REPLACE MEMORY.
005B 7E          HERE MOV A,M       *GET MEMORY
005C 02           STAX B            *AND VIEW IT
005D D2 47 00     JNC COUNT         *UNTIL ALL DIGITS ARE VIEWED.
0060 2B           DCX H             *CHECK MOST SIGNIFICANT DIGIT
0061 BE           CMP M             *AGAINST NEXT MOST.
0062 CA 67 00     JZ THERE          *BOTH ZERO? EXIT.
0065 EB           XCHG              *NO, INCREASE
0066 34           INR M             *END OF COUNT.
0067 21 BF 02    THERE LXI H,MADD-LINE-1
006A 16 02        MVI D,TAD         *GET IN POSITION FOR TABLE.
006C 01 D6 00    BYTE LXI B,INST    *PSEUDO OP LIST
006F 0A          BIT LDAX B         *LOAD PSEUDO OP.
0070 0F           RRC               *CHECK RIGHT BIT FOR
0071 D2 87 00     JNC ROT           *CELL CHECK FROM SAME BYTE
0074 0F           RRC               *NO. NEXT BYTE?
0075 D2 83 00     JNC ONE           *YES
0078 FE F0        CPI 0F0H          *NO. ALL NGHBRS DONE THIS BYT
007A D2 99 00     JNC DONE          *YES.
007D 11 3D 00     LXI D,LINE-3      *NEXT LINE ON 3X3 MATRIX
0080 19           DAD D             *INCREMENT BY LINE-2
0081 16 02        MVI D,TAD         *BY LINE-3+1, SINCE WE NEED
0083 23          ONE INX H          *A +1 ANYWAY
0084 E6 3F        ANI 3FH           *GET RID OF 2 MSB'S.
0086 07           RLC               *ZERO CARRY BIT AND
0087 1F          ROT RAR            *GET IN POSITION
0088 03           INX B             *FOR THIS AND NEXT PSEUDO OP
0089 5F           MOV E,A           *2ND BYTE FEEDS MASK TABLE
008A 1A           LDAX D            *LOAD MASK FOR BIT
008B A6           ANA M             *AND CHECK IT ON THE MASTER
008C CA 6F 00     JZ BIT            *NO LIFE, NEXT BIT
008F EB           XCHG              *BRING DOWN SCRATCH
0090 3E 07        MVI A,07H         *ADDRESS TO STORE NEIGHBOR
0092 A5           ANA L             *COUNT CODED BY BIT #
0093 6F           MOV L,A
0094 34           INR M             *COUNT ONE NEIGHBOR
0095 EB           XCHG              *GET MASTER COPY
```

```
0096 C3 6F 00    JMP BIT            *AND GET NEXT BIT IN BYTE
0099 01 BF FF   DONE LXI B,0-LINE-1 *GO BACK TO BYTE
009C 09          DAD B              *THAT WE'RE WORKING ON
009D 1E 00       MVI E,0            *ZERO SCRATCHPAD BYTE #2
009F E3          XTHL               *MOVING ON TO SLAVE COPY.
00A0 97         LOAD SUB A          *ZERO A SO WE CAN
00A1 12          STAX D             *ZERO NEIGHBOR COUNT
00A2 79          MOV A,C            *GET INVERTED BIT MASK
00A3 0F          RRC                *COMING IN BFH AND ROTATE
00A4 D2 BF 00    JNC NEXT           *GOT ALL BITS?
00A7 4F          MOV C,A            *NO, REPLACE MASK
00A8 1C          INR E              *AND COUNT BIT NUMBER
00A9 1A          LDAX D             *GET # NEIGHBORS OF THAT BIT
00AA FE 02       CPI 02H            *IS IT TWO?
00AC CA A0 00    JZ LOAD            *YES, CELL STAYS THE WAY IT IS
00AF 79          MOV A,C            *NO, SO
00B0 A6          ANA M              *KILL CELL ON
00B1 77          MOV M,A            *SLAVE COPY
00B2 1A          LDAX D             *HOW MANY NHBRS AGAIN?
00B3 FE 03       CPI 03H            *ARE THERE THREE?
00B5 C2 A0 00    JNZ LOAD           *YES, GOOD WE KILLED IT.
00B8 79          MOV A,C            *OOPS, GOT TO RESURRECT IT
00B9 2F          CMA                *BY INVERTING THE MASK
00BA 86          ADD M              *AND ADDING
00BB 77          MOV M,A            *REPLACE SLAVE
00BC C3 A0 00    JMP LOAD           *UPDATE NEXT BIT IN BYTE
00BF 01 C0 FF   NEXT LXI B,0-LINE   *UP ONE, WHICH IS UPPER
00C2 23          INX H              *INCREMENT SLAVE ADDRESS
00C3 E3          XTHL               *FOR PROPER INITIALIZATION
00C4 3E 07       MVI A,MAD+04H      *END OF SCREEN?
00C6 BC          CMP H
00C7 09          DAD B              *COMPLETE ONE UP
00C8 C2 6C 00    JNZ BYTE           *SCREEN NOT OVER, NEXT BYTE
00CB E1          POP H              *LEAVE
00CC 21 00 08    LXI H,SADD         *SADD ON STACK
00CF E5          PUSH H             *FOR NEXT TIME.  SET UP TO
00D0 11 00 88    LXI D,VADD         *SWAP SLAVE TO SCREEN
00D3 C3 20 00    JMP LOOP           *ON EACH SUCCESSIVE LOOP.
00D6 C4 65      INST DW 65C4H       *PSEUDO OPS CODE 48
00D8 C4 70       DW 70C4H           *SPECIAL CASES: EIGHT
00DA D0 71       DW 71D0H           *NEIGHBORS FOR EACH OF
00DC 87 A4       DW 0A487H          *SIX CELLS PER BYTE
00DE 88 A8       DW 0A888H          *RIGHT TWO BITS OF
00E0 C8 AC       DW 0ACC8H          *EACH PSEUDO OP INDICATE
00E2 CC 45       DW 45CCH           *WHETHER NEXT NEIGHBOR IS
00E4 84 A4       DW 0A484H          *IN THE SAME BYTE AS
00E6 28 68       DW 6828H           *CURRENT NEIGHBOR, OR IN
00E8 88 A8       DW 0A888H          *NEXT BYTE, OR NEXT LINE
```

```
00EA C8 4C      1940  DW 4CC8H          *IN 3X3 MATRIX OF
00EC AC CC      1950  DW 0CCACH         *NEIGHBOR BYTES
00EE 30 50      1960  DW 5030H          *NEXT THREE BITS CODE
00F0 B0 34      1970  DW 34B0H          *CELL WHOSE NEIGHBORS
00F2 54 74      1980  DW 7454H          *WE ARE COUNTING, IN
00F4 94 D4      1990  DW 0D494H         *REVERSE ORDER
00F6 58 78      2000  DW 7858H          *REMAINING THREE BITS
00F8 B8 31      2010  DW 31B8H          *CODE MASK FOR NEIGHBOR
00FA 50 34      2020  DW 3450H          *IN SAME FORMAT
00FC 54 74      2030  DW 7454H
00FE 58 78      2040  DW 7858H
0100 8F 2D      2050  DW 2D8FH
0102 8C 38      2060  DW 388CH
0104 98 39      2070  DW 3998H
0106 FF         2080  DB 0FFH
```

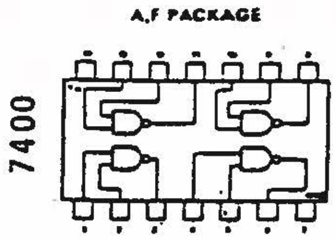
Screen clearing routine

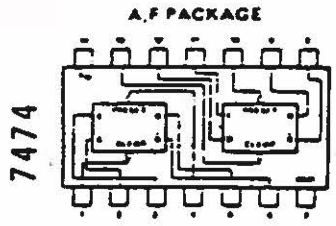
```
ASSM(CLEAR) 0F00

0F00 21 00 88           1000  LXI H,8800H
0F03 36 7F              1010 LOOP MVI M,7FH
0F05 23                 1020  INX H
0F06 7C                 1030  MOV A,H
0F07 FE 8C              1040  CPI 8CH
0F09 C2 03 0F           1050  JNZ LOOP
0F0C 76                 1060  HLT
```

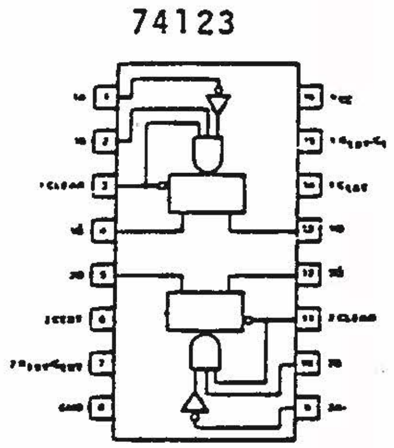
ASCII Character set

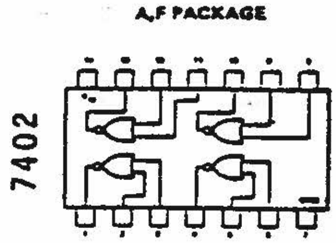
Bits				b7 b6 b5	0 0 0	0 0 1	0 1 0	0 1 1	1 0 0	1 0 1	1 1 0	1 1 1
b4	b3	b2	b1	COLUMN / ROW	0	1	2	3	4	5	6	7
0	0	0	0	0	NUL	DLE	SP	0	@	P	`	p
0	0	0	1	1	SOH	DC1	!	1	A	Q	a	q
0	0	1	0	2	STX	DC2	"	2	B	R	b	r
0	0	1	1	3	ETX	DC3	#	3	C	S	c	s
0	1	0	0	4	EOT	DC4	$	4	D	T	d	t
0	1	0	1	5	ENQ	NAK	%	5	E	U	e	u
0	1	1	0	6	ACK	SYN	&	6	F	V	f	v
0	1	1	1	7	BEL	ETB	'	7	G	W	g	w
1	0	0	0	8	BS	CAN	(	8	H	X	h	x
1	0	0	1	9	HT	EM	)	9	I	Y	i	y
1	0	1	0	10	LF	SUB	*	:	J	Z	j	z
1	0	1	1	11	VT	ESC	+	;	K	[	k	{
1	1	0	0	12	FF	FS	,	<	L	\	l	¦
1	1	0	1	13	CR	GS	−	=	M	]	m	}
1	1	1	0	14	SO	RS	.	>	N	^	n	~
1	1	1	1	15	SI	US	/	?	O	_	o	DEL

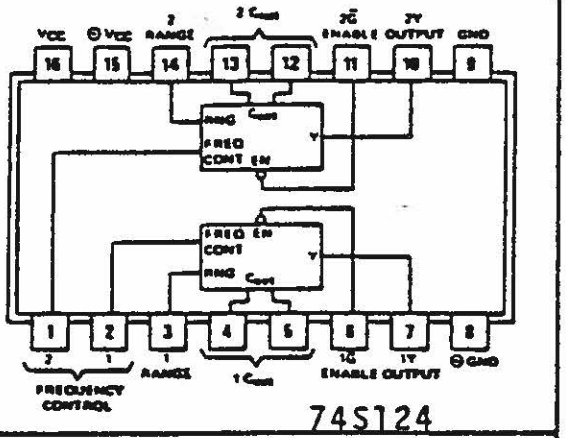
CHIP PINOUTS



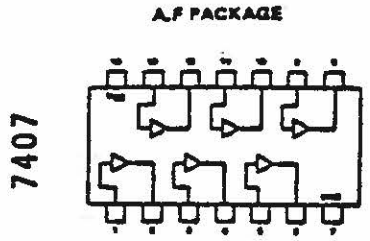


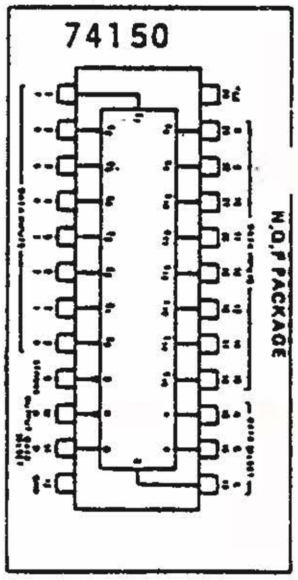


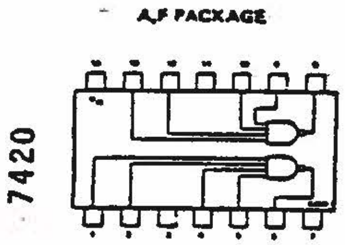


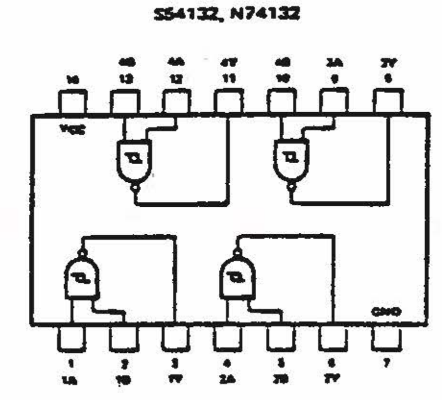


positive logic: While the enable input is low, the output is enabled. While the enable input is high, the output is high.

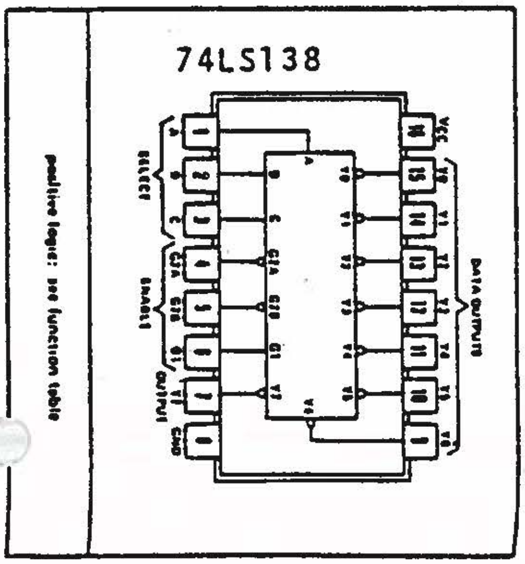


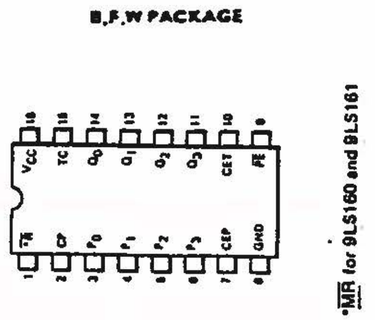




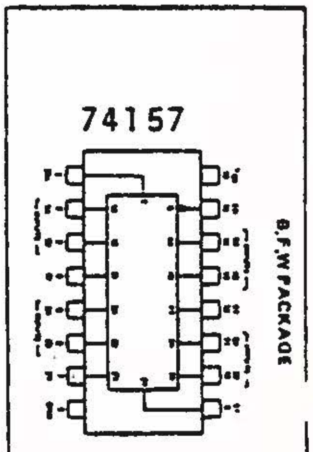


POSITIVE LOGIC: $Y=\overline{AB}$

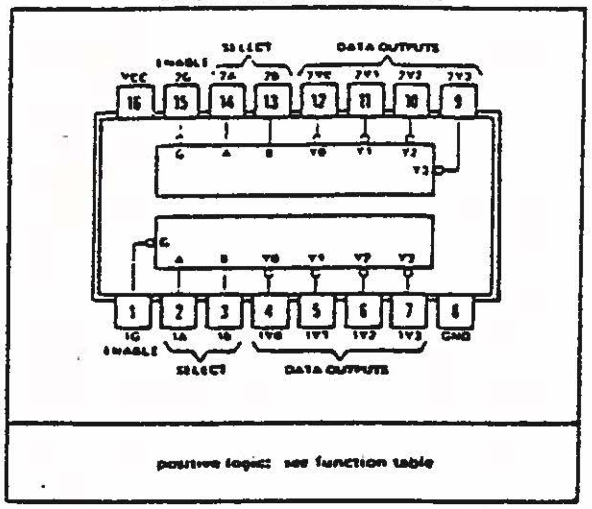




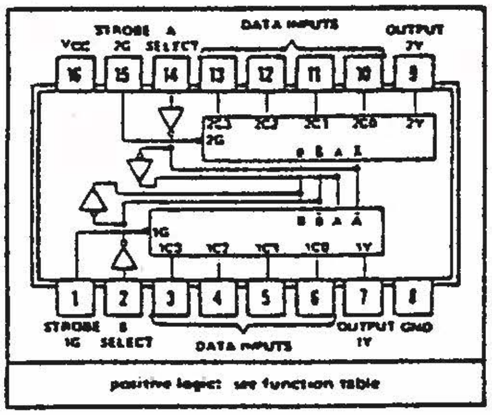
· 74161



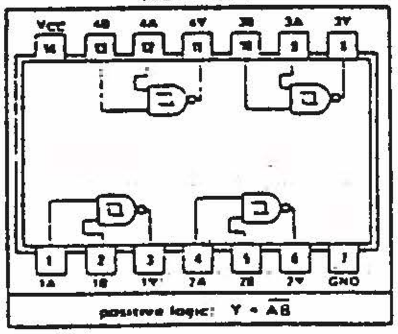
'LS139, 'S139
J OR N DUAL-IN-LINE OR
W FLAT PACKAGE (TOP VIEW)

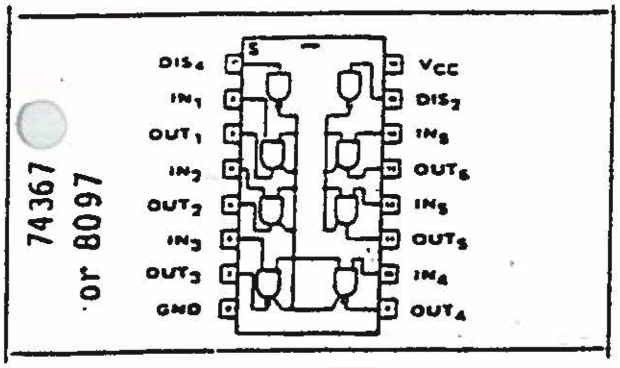


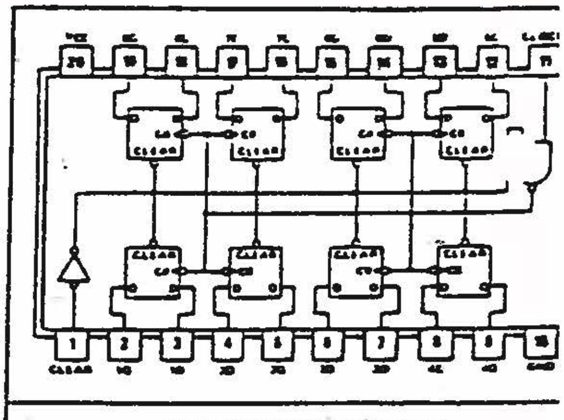
'153, 'LS153, 'S153 . . . J, N, OR W PACKAGE
'L153 . . . J OR N PACKAGE
(TOP VIEW)



SN54LS132 . . . J OR W PACKAGE
SN74LS132 . . . J OR N PACKAGE
(TOP VIEW)

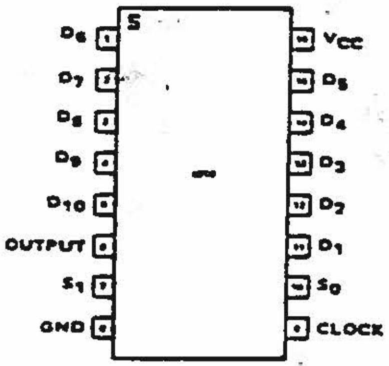


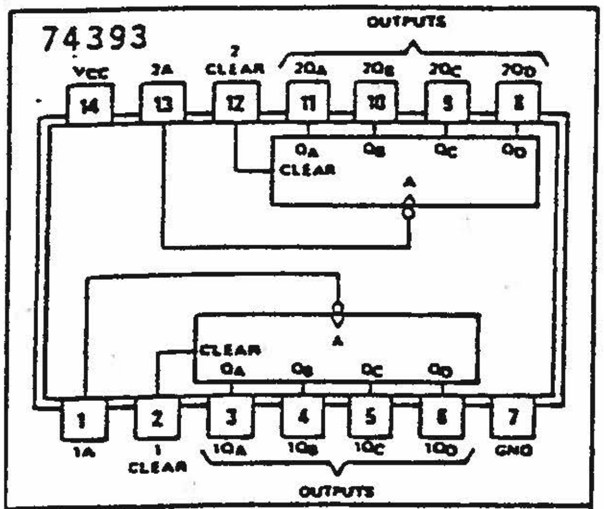




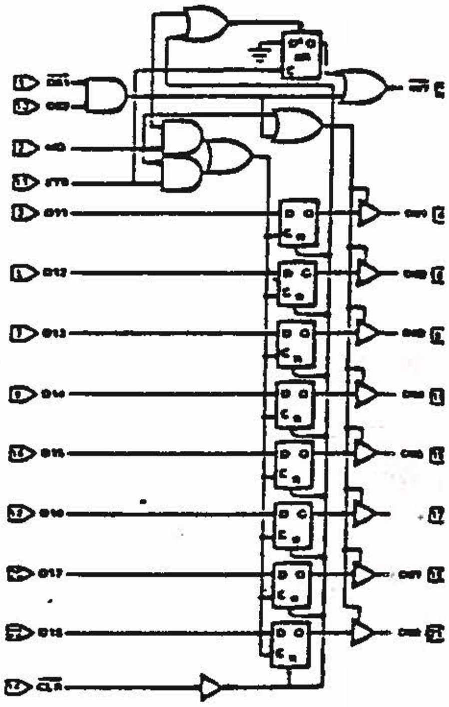
74273 positive logic: see function table

N8274B

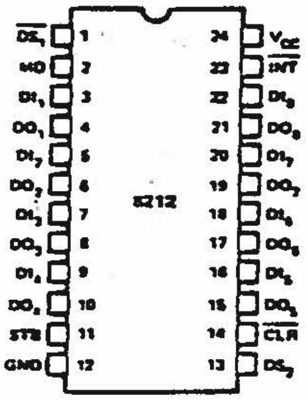


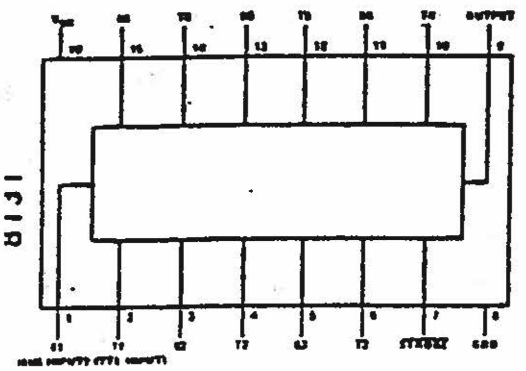


positive logic: High input to clear resets all four outputs low

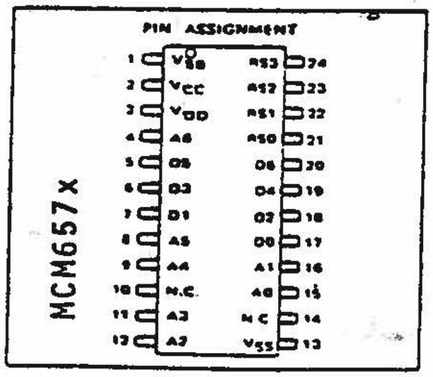


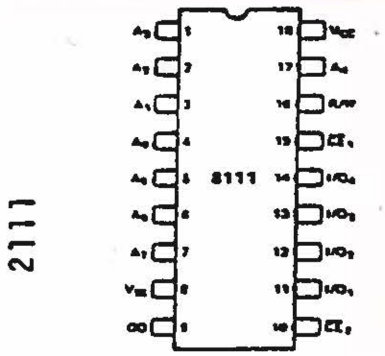
8212 or 74S412



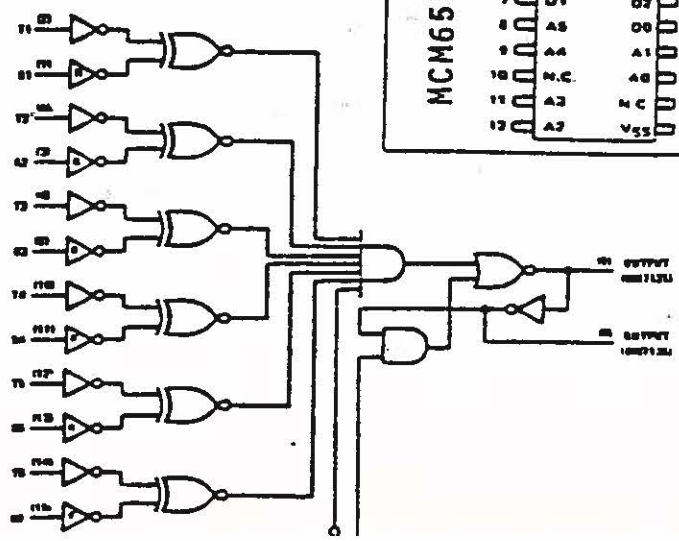


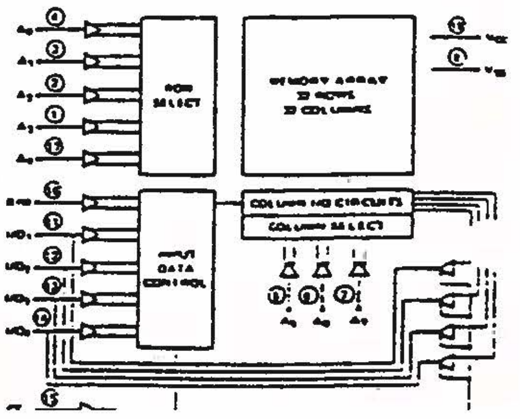
7131(J), (W); 8131(J), (N), (W);
7136(J), (W); 8136(J), (N), (W)





Logic Diagram


OUTPUT
OUTPUT


ROW SELECT
MEMORY ARRAY
INPUT DATA CONTROL
COLUMN I/O CIRCUITS
COLUMN SELECT