The Standard of Excellence in Microcomputer Systems.

IMSAI

PCS 80/30

REFERENCE MANUAL

PRELIMINARY

IMSAI Manufacturing Corporation
San Leandro, CA

IMSAI

PCS - 80/30

REFERENCE MANUAL

PRELIMINARY

October, 1977

TABLE OF CONTENTS

I. INTRODUCTION

1. INTRODUCTION

This volume, the IMSAI PCS-80/30 Reference Manual, outlines the theory of operation of the system PC boards and contains detailed information regarding the use of the module features. Section II of the manual details the MPU-B Microprocessor Board, Section III is a description of the Video Board (VIO-C), and Section IV relates to the Intelligent Keyboard (IKB-1). Section V contains technical information on the Power Supply (PS-28U).

II. THE 8085 MICROPROCESSOR BOARD (MPU-B)

II. THE 8085 MICROPROCESOR BOARD (MPU-B)

A. THEORY OF OPERATION

1. INTRODUCTION

The MPU-B is a central processing board for the S-100 bus based on the 8085 microprocessor. In addition to the processor, it contains:

- 1K program PROM/ROM
- 1/4 K RAM
- Power on jump
- Serial I/O
- Parallel I/O
- Timers
- 5-level interrupts

The 1K byte PROM/ROM socket is provided with jumpers to configure it for 1K or 2K byte PROMs or 1K, 2K, or 4K byte ROMs. A 1K ROM is included with basic system support firmware. Under software control, the ROM can be located at 0000, or at D800. Power on clear initializes the ROM at 0000 to serve as a power on jump. The processor can write to the RAM at 0000 even when executing code in the ROM at 0000. A custom programmed PROM in this location can initialize the system, load programs in the entire 65K memory space (or more with the IMM installed), and then disable the PROM while jumping to the program start location. The PROM can be re-enabled at any time to use code stored in it, either at location 0000 or at location D800. The PROM enable is delayed to permit a jump or call to a routine even from a program executing directly underneath the PROM location. A system reset automatically re-enables the PROM at 0000.

There are 256 bytes of RAM provided on the board for minimum stand-alone systems, and for variable & data storage at a fixed location for the loaders and monitor in ROM or custom programs in PROM. It can be software enabled/disabled, so that loader and monitor variables do not take up user memory space. It is located at D000, and is enabled/disabled at the same time as the PROM at D800. It is not enabled by the PROM being at 0000, however the PROM may be enabled at both locations simultaneously if the RAM is desired with the PROM at 0000.

The ROM included provides power on system initialization, automatic baud rate select if the serial interface is used, and complete monitor facilities. The monitor includes loaders for floppy disk, casette tape, and paper tape, as well as extensive machine language entry and debugging features. The monitor will automatically run from either the serial interface or a keyboard and VIO. The loaders and other routines can be called from user programs.

The serial I/O provides complete serial interfacing capability. True RS-232 levels are driven for data and handshaking lines both in computer end and terminal end configurations. No jumpering is needed. A 20 mil current loop interface is included for driving ASR-33 type devices. Baud rates are software selectible at any standard or non-standard rate up to 9600 baud in asynchronous mode, or 56K baud in synchronous mode. The input data ready, output

buffer empty, and sync detect status flags can be switched to interrupt the processor, and/or used in a polled status mode. For software compatibility, serial port addressing is the same as an MIO at 00 or 10, or both ports of an SIO at 00, or the first port of an SIO at 10.

The parallel I/O provides a complete 8-bit parallel port, with handshaking and latching available in both in and out directions. The input data ready and output buffer empty status flags can be switched to interrupt the processor, and/or used in a polled status mode. For compatibility, the connector has the same pin definitions used on the MIO or PIO, and the addressing permits a keyboard to look like the serial input of an MIO at 00, or the second port of an SIO at 10. While the parallel port is used by the monitor for keyboard input, the monitor can also use the serial port. The parallel port is otherwise free to be used for any other parallel interfaced device such as the AP-44 mini-printer.

Three 16 bit timers are provided. All three are completely software programmable and readable. They are memory mapped at D100-D103, and are software enabled/disabled along with the RAM and PROM in D000. One timer is normally dedicated to provide the baud rate clock for the serial interface, but can be used differently when serial I/O is not needed. All three timers have clock inputs of 2 MHz, with one timer's clock switch selectible to be either 2 MHz or the output of another timer. This permits a programmable interrupt or periodic interrupt period from microseconds to more than 30 minutes. Both real time clocks and time of day clocks are easily implemented. The outputs of the two timers not used for serial clock can be switch selected to interrupt the processor at count termination, or periodically, depending on the software programmed mode of operation. The timer, once programmed and started, does not have to remain enabled (with the RAM & PROM) to operate. Interrupts will occur regardless of enabled status, or it can be momentarily enabled to read the current counts. The baud rate clock continues to run with the timer disabled from memory space. The two free timers can be used to provide software controlled baud rate clocks for serial interfaces on the MIO or SIO.

The MPU-B can be used either with or without a CP-A front panel. The power on jump and monitor eliminate the need for a front panel for any program operations. A CP-A may be used for hardware development or maintenance.

All address decoding on the MPU-B is done in a fusible link ROM, so that all the specific addresses mentioned may be changed for special applications, including memory mapping the serial or parallel I/O or I/O mapping the timers or any portion of the RAM. Special monitors can be installed in the program ROM socket to support OEM applications and special applications packages. Switching the RAM, PROM, and timers out of the address space facilitates implementing larger, more sophisticated systems by removing all address conflicts.

or

By adding power supplies and a terminal, the MPU-B forms a complete computer system with everything necessary to use it as an evaluation system or develop small machine language games or applications.

2. MPU-B CIRCUIT DESCRIPTION

The IMSAI MPU-B processor board interfaces an 8085 microprocessor chip to the S-100 bus. In addition to the processor and S-100 bus interface, the MPU-B has parallel and serial I/O, PROM/ROM, RAM, timer, and necessary address decoding. Power is regulated on the board. The processor runs at 3.0 MHz, and runs 8080 machine code along with a couple added instructions. Thus all existing 8080 code is 100% software compatible with the 8085 and MPU-B, paying due attention to the specifics of I/O device attachment. The I/O ports included on the MPU-B have been addressed such that IMSAI software will run without any changes, at the same time allowing switch selection of either the serial port (for a terminal) or the parallel port (for a keyboard & VIO) as the system I/O device. Both serial and parallel I/O are always available for systems which use both.

Even though the processor runs 1 1/2 times as fast as a standard 8080, the S-100 bus signals have looser timing requirements due to the 8085's more efficient use of the bus. The clock lines phase 1 and 2 run at the higher 3 MHz rate, however the CLK line is run at 2 MHz for use by existing S-100 boards that depend on a 2 MHz timing source. Most other S-100 bus lines retain their old definitions. Although the timing of some of the signals is slightly different from the 8080, the chip was designed to replace 8080's in existing systems. Any properly designed S-100 boards should experience no problems running with the MPU-B.

Most of the circuitry interfaces to a local bidirectional bus, which is connected to and defined by the 8085 pins. The circuitry in each of the blocks will be described.

3. 8085 PROCESSOR

The 8085 processor section of the board contains only a little circuitry beyond the 8085 chip itself. A 6 MHz crystal is connectec to the on-chip clock generator. All other system timing is generated from the 3 MHz clock out pin on the 8085. A power-on reset time constant of 20 milliseconds is generated by resistor PN2 and capacitor PN3 connected to the schmitt trigger reset input. The delay allows ample time for power supply voltages to stabilize before the 8085 is permitted to start running. Diode PN4 insures that if power is lost even for a short time, the 8085 will come back up running properly. When the +5 volt line falls below the schmitt's lower threshold, capacitor PN3 will be discharged quickly through diode PN4. When power is re-applied, the system begins to run only after PN3 charges to above the schmitt's upper threshold. Any manual system reset also discharges PN3, and the 8085 resumes running only after PN3 charges back to the upper threshold through PN2.

To insure that nothing connected to the bus is falsly activated during a hold or halt operation (during which the 8085 bus is three-stated), the three strobe lines (/WR, /RD, & /INTR) are held inactive by pullup resistors. The IO/M line is also pulled up to indicate an I/O status during hold or halt states, to prevent any of the on-board memory mapped circuitry being enabled, since in one mode of operation they would respond to a write strobe generated external to the MPU-B.

The 8085 is powered only by +5 volts.

The four RST lines into the 8085 have been arranged to permit use of interrupts for basic system I/O without the use of a priority interrupt card or any jumpers. One line is reserved for interrupts from a mass storage device (TRAP, @RST4.5) such as a floppy disk. The other three can be connected to the on-board I/O by activating DIP switches. The interrupts have

been segregated. RST7.5 is used for the extra timer sections, RST6.5 is used for output data buffer empty on both the serial and parallel interfaces, and RST5.5 is used for input data ready on both parallel and serial interfaces, and for sync. character detect on the serial interface. Open collector logic was used throughout the interrupt section, and jumper pads provided to connect these lines to the VIx lines on the S-100 bus. This permits connecting other interrupt devices to the same lines without a priority interrupt board. If a priority interrupt board is added, it can drive the 8 standard RST locations, and the MPU-B I/O can continue to use the RST lines unique to the 8085.

4. S-100 BUS INTERFACE

Data Out:

The local address/data bus (AD0-AD7) from the 8085 always appears on the data out bus unless an external board pulls data out disable low (/DODSBL, pin 23). It is driven by 8 sections of 8T97 three-state bus drivers.

Data In:

The three-state bus drivers connecting the data in bus to the local address/data bus are always held disabled unless one or the other of the two input strobes are active (/RD or /INTA). This condition is part of the definition of /IDIDSBL. These input data drivers are also held disabled by either an external board pulling data in disable low (/DIDSBL, pin 21), or by other conditions causing an active signal on internal data in disable (/IDIDSBL). /IDIDSBL is active whenever any of the on-board memory or I/O is enabled, or whenever an input strobe is not present.

Address 0-7:

The local address/data bus is latched into PN5 at the trailing edge of ALE (address latch enable). At this time address bits 0 through 7 are on the A/D bus. The three-state driver outputs of the 8212 (PN5) drive the lower half of the address bus directly.

Address 8-15:

The upper half of the address bus (A8 through A15) is driven by 8T97's directly from the internal A8-A15 address bus. Both the 8T97 drivers and the 8212 drivers on the address bus can be disabled by an external board pulling address disable low (/ADDRDSBL, pin 22).

Data and Address Disables:

/DODSBL and /ADDRDSBL are normally driven only during a DMA transfer, when the MPU-B must be removed from the bus so that the DMA system may access the memories. Note that the address and data drivers on the MPU-B are not bidirectional; thus DMA operations to/from the on-board memory and I/O are not enabled.

Address timing:

Both the high and low half of the address bus appear early in T1, and since the 8212 is a gated latch, there is ample setup time to latch the full address at the trailing edge of ALE (PSYNC, pin 76) on an external card

5. STATUS

Status on the 8085 is encoded differently than on the 8080. In order to generate the lines needed to run the S-100 bus, the 8085 status is used as address inputs to a ROM. The pattern in the ROM translates the 8085-type status at the inputs to 8080-type status at the outputs. The 74S288 ROM used (PN6) has three-state outputs with sufficient drive to directly run the S-100 bus. An external board can disable the status drivers by pulling status disable low (/STSDSBL, pin 18), which will remove chip select from PN6.

Status timing:

The status appears at the same time as the address in T1, rather than having to be latched at a later time, as in the 8080. It remains valid until the corresponding time in the proceeding T1. There is ample setup time to permit latching the status with the trailing edge of ALE (PSYNC, pin 76) on an external card.

Status differences from the 8080:

Most of the status looks identical to that of the 8080 on the MPU-A. Two status lines, however, are not among the states presented by the 8085.

One of these is interrupt acknowledge (SINTA, pin 96). The 8085 does not uniquely identify an interrupt acknowledge cycle until the time that the /INTA input strobe appears. Until that time, the status is the same as an instruction fetch status. As a result, the MPU-B interrupt acknowledge status appears on the S-100 bus as an instruction fetch from the beginning of the cycle to the start of the input strobe. At that time it changes to an interrupt acknowledge status, enabling the RST or CALL op-code to be presented by the interrupting board (PIC-8, IMM, etc.). This reduces the access time available to the interrupting board to approximately 280 nsec, which is sufficient since no memory chip access time is involved.

The other status line is not decoded at all by the 8085. The stack operation status line (SSTACK, pin 98) is held not true (low) permanently (the signal can be opened should it be desired to drive it with another board). Since this line has not been used, its absence will not be a problem.

6. CONTROL STROBES

There are two input strobes on the 8085, which are OR'ed together and inverted to form a single positive input strobe (PDBIN) for the S-100 bus.

The 8085 signals /WR, ALE, and HLDA are used directly to drive S-100 signals (respectively) /PWR, PSYNC, and PHLDA. For systems without a front panel, the S-100 signal MWRITE is generated from /PWR and SOUT. The driver for MWRITE can be disabled by moving the dipswitch FP-/FP to the FP position (this permits the front panel to drive this line instead of the MPU-B). Provisions are also made for permanently disabling the MWRITE line for systems in which it is not used.

Wait signal generation:

The S-100 signal PWAIT is not generated by the 8085. This signal has been used on many S-100 bus boards to generate a single wait state for memories between 500 nsec and 1 usec, so the MPU-B includes a circuit to synthesize the required timing for PWAIT. PN7 is a latch which puts the inverse of the RDY line onto PWAIT; when the 8085 is receiving a not ready signal, an active (high) PWAIT signal is put on the S-100 bus. In order to match the timing of the 8080 WAIT, PWAIT should not become true (high) until the end of T2 (beginning of TWAIT). Since RDY will normally be driven low during T1, another signal is needed to prevent PWAIT from going high too soon. PN8 is used to delay ALE at the beginning of each cycle to provide a signal to delay the start of PWAIT.

During T4, T5, and the first part of the next T1, PWAIT can return high if the memory board still returns PRDY = PWAIT. This is of no consequence since the 8085 does not test RDY during this part of a cycle.

Control signal disable:

The five control lines will be disabled if an external board pulls control disable low (/CCDSBL, pin 19). This is normally done only during a DMA operation. If the DMA board waits for PHLDA before disabling the control bus, the 8085 will have just completed returning the strobes (PDBIN & PWR) to the inactive state. If the DMA board waits longer, or during a halt state, the MPU-B will continue to hold the strobes inactive. It is up to the DMA card to insure that the strobes cannot reach an active state during the transition of bus control.

7. CLOCKS

Clock out (COUT) from the 8085 provides a 3 MHz clock which is put on the S-100 phase 2 line (pin 24). This signal is inverted and used to provide phase 1 (pin 25). PN9, PN10 and PN11 form a divide by 3/2 circuit to produce a 2 MHz signal from the 3 MHz clock. The output waveform is a 2 MHz, 33% duty cycle (high) pulse train which is put on the CLK line (pin 49) for use by S-100 boards which need a 2 MHz timing source. Some S-100 boards may have originally used phase 1 or 2 for a timing source and these will need a "blue wire" change. Do not confuse the use of phase 1 or 2 as a strobe with their use as a timing source.

8. CLEAR

Positive going reset out (RESOUT) from the 8085 is inverted and used to drive power on clear (/POC, pin 99). The output of the driver is also used to drive external clear on the S-100 bus through a germanium diode. The diode permits a front panel, if present, to lower the external clear line (/EXTCLR, pin 54) without affecting the /POC line. With no front panel, there is no differentiation between the two clear lines.

The clock and the clear lines are driven all the time.

9. CONTROL INPUTS

Reset:

The /PRESET signal (pin 75) connects directly to the power on reset network described in the 8085 processor section. PN2 and PN3 provide enough time constant that a mechanical switch may be connected directly between /PRESET and ground, as long as the contact bounce is

shorter than approximately 10 msec.

Hold:

/PHOLD is held inactive by a 1K pullup, and inverted to drive HOLD on the 8085. HOLD is sampled on the trailing edge of phase 2, but does not have to be externally synchronized. The hold line is used primarily by DMA interfaces to cause the 8085 to cease executing at the end of the current instruction, and release the bus so that a DMA interface can gain control of the bus.

Ready:

RDY is high when both S-100 signals PRDY (pin 72) and XRDY2 (pin 12) are high. The S-100 signals are pulled inactive wth 1K resistors so that they need not be driven. If they are not driven, a constant ready is presented to the 8085 and it proceeds executing at full speed.

PRDY is used by memories, I/O cards, or interrupt controllers which need to delay the 8085's sampling of input data or cause it to hold output data longer. XRDY2 is used by the front panel (if present) to hold the 8085 from continueing or to allow it to single step only. Either line pulled low will stop the 8085 between the next T2 and T3.

10. INTERRUPTS

The S-100 interrupts enabled line (INTE, pin 28) has no equivalent in the 8085. It is pulled true with a 1K resistor, for any interrupt boards which look at it. There should be no effect on the operation of those boards.

The interrupt request line is pulled inactive with 1K, and drives INTR on the 8085 through an inverter. There are 4 interrupt lines on the 8085 which do not exist on the 8080. Jumper pads are provided so that these may be jumpered to any of the 8 S-100 priority interrupt request lines. This eliminates the need of a priority interrupt board for up to 5 levels of interrupt; the four 8085 restart lines and the standard interrupt request. (If the interrupt request is used without an interrupt board, no one will be driving the bus at interrupt acknowledge time, and the processor will read an FF opcode, which is an RST7.

The four 8085 restart lines are pulled inactive with 1K resistors, so they need not be jumpered. Of the four, only the TRAP line (non-maskable RST 4.5) comes jumpered (to V17, pin 11). The other three can be connected to the on-board I/O via DIP switches.

11. S-100 BUS INTERFACE TABLE

Following is a summary of the standard S-100 bus signals including:

S-100 pin number
High true (H) or low true (L)
Driven/received by the MPU-B (*) jumperable (X)
S-100 signal name

1	- *	+8 VOLTS POWER	51	- *	+8 VOLTS POWER
2	- *	+16 VOLTS POWER	52	- *	-16 VOLTS POWER
3	H	XRDY	53	L	/SSW DSB
4	L	VI0	54	L *	/EXTCLR
5	L	VI1	55		
6	L	VI2	56		
7	L	VI3	57		
8	L X	VI4	58		
9	L X	VI5	59		
10	L X	VI6	60		
11	L *	VI7	61		
12	H *	XRDY2	62		
13	H	AA15	63		
14	H	AA14	64		
15	H	A18	65		
16	H	A16	66		
17	H	A17	67		
18	L *	/STSDSBL	68	H *	MWRITE
19	L *	/CCDSBL	69	?	PROTECT STATUS
20	?	MEM.UNPROTECT	70	?	MEM.PROTECT
21	L *	/DIDSBL	71	H *	RUN
22	L *	/ADDR DSBL	72	H *	PRDY
23	L *	DODSBL	73	L *	/PINT
24	H *	02 (PHASE 2)	74	L *	/PHOLD
25	H *	01 (PHASE 1)	75	L *	/PRESET
26	H *	PHLDA	76	H *	PSYNC
27	H *	PWAIT	77	L *	/PWR
28	H *	PINTE	78	H *	PDBIN
29	H *	A5	79	H *	A0
30	H *	A4	80	H *	A1
31	H *	A3	81	H *	A2
32	H *	A15	82	H *	A6
33	H *	A12	83	H *	A7
34	H *	A9	84	H *	A8
35	H *	DO1	85	H *	A13
36	H *	DO0	86	H *	A14
37	H *	A10	87	H *	A11
38	H *	DO4	88	H *	DO2
39	H *	DO5	89	H *	DO3
40	H *	DO6	90	H *	DO7
41	H *	DI2	91	H *	DI4
42	H *	DI3	92	H *	DI5
43	H *	DI7	93	H *	D6
44	H *	SM1	94	H *	DI1
45	H *	SOUT	95	H *	DI0
46	H *	SINP	96	H *	SINTA
47	H *	SMEMR	97	L *	/SWO
48	H *	SHLTA	98	H *	SSTACK
49	H *	CLK (2MHZ)	99	L *	/POC
50	- *	GROUND	100	- *	GROUND

12. ADDRESS DECODE

The address decoder on the MPU-B uses a 512x4 bit ROM (PN12), a dual 2 to 4 decoder (PN13), and several small gates. It produces chip enable signals for all the on-board memory and I/O circuits (PROM, RAM, timer, serial I/O, parallel I/O, and control port).

Address decode ROM:

The 9 inputs to the 512x4 rom (PN12) are the 8 high order address bits and the IO/M status line. The address decode portion of the MPU-B does not decode the low order address bits. Thus on board memory circuits either have low order address decoding at the circuit (PROM, RAM) or appear at multiple locations due to incomplete address decoding (timer). The state of address bits 8 through 15 and the I/O operation / memory operation status line (IO/M) is decoded by the ROM pattern. Since greater than 4 circuit enables are needed, the output of the ROM is encoded. The enables are divided into two groups: PROM, RAM, timer, and parallel I/O, serial I/O, control port. Output bit 4 low indicates that one of the enables in group 1 is active (PROM, RAM, or timer). Output bit 3 low indicates that one of the enables in group 2 is active (parallel I/O, serial I/O, or control port). Both bits 3 and 4 high indicate that the address and IO/M presented to the ROM are not in the address space of any of the on-board circuits. The ROM is encoded so that output bits 3 and 4 are never both low. Each of these bits is used as an enable to one half of the dual 2 to 4 decoder.

There is one time when an on-board enable may be indicated by the address and IO/M incorrectly. That is during an interrupt acknowledge. The 8085 appears as though it were going to execute the next instruction fetch at interrupt acknowledge time, except that a /INTA input strobe appears instead of a /RD strobe. In order to avoid a bus conflict and permit the interrupt board to send the interrupt op-code to the 8085, all memory and I/O circuits must recognize this condition and stay off the bus. To allow for this situation, INTA (the inverse of the /INTA strobe) is connected to the ROM chip enable input. Normally this is low, enabling the ROM; but during an interrupt acknowledge cycle, it goes high at data input time. ROM output bits 3 and 4 are pulled high by resistors when the ROM is disabled, and neither half of PN13 is enabled.

Output bits 1 and 2 are encoded to indicate which circuit enable is present, according to the following table:

Group 1:

21	
11	PROM enabled if POJM is true
10	PROM normal location
01	RAM
00	timer

Group 2:

21	
11	Parallel port
10	Serial port
01	System port
00	Control port

In the standard ROM, group 2 is all I/O ports and group 1 is all memory mapped. The board is not restricted to this scheme, but if a different ROM is made for a custom application, some restrictions need to be considered.

Any circuit mapped as I/O cannot use the /MWRT write strobe because it is not active during I/O writes. A jumper is provided so that the two devices using this strobe (timer and RAM) can be switched to /IWR, the internal bus write strobe. If this is done, the circuits will run either memory mapped or I/O mapped, but the front panel (if present) cannot write into the memory mapped circuit since it uses /MWRT. The processor has no restrictions on writing.

The Serial I/O, Parallel I/O, and control port run from the internal strobes and can be memory or I/O mapped or both. They cannot be jumpered to respond to memory mapped writes from CP-A's. The program PROM/ROM is a read only device, and can be memory or I/O mapped. Writes to these addresses affect whatever system memory or I/O has been installed off the MPU-B. The RAM and timer can be memory mapped or I/O mapped, but if either is I/O mapped, the jumper provided must be changed to /IWR instead of /MWRT. This will prevent front panel writes to either as noted above.

When a device is memory mapped on the MPU-B, it takes up a minimum of 256 bytes of memory space. On board I/O device decoding can be complete since the complete I/O address appears on A8 - A15. Note that some of the I/O circuits use address bits from A0 - A7 for further decoding of port address. In these cases the corresponding address bits in A8 - A15 are made don't care bits in the address decode ROM.

Dual 2 to 4 decoder:

The decoder translates ROM output bits 1 and 2 into individual enable lines according to the table above. Section one decodes group 2 (I/O and control) and is enabled only by ROM output bit 3. This group of crcuits is a permanent feature of the address space of the 8085, it cannot be disabled.

Group 1 (memory and timer) is decoded by section 2. ROM output bit 4 is one of the enables for this section; the other enable input is driven by the signal IMEM. One of the four group 1 enable outputs from PN13 is not used. Instead, the output is decoded separately and ANDed with a different enable. Instead of being enabled with IMEM, it is enabled with POJM (Power On Jump Mode), and is ORed with the normal program ROM/PROM enable to provide a separate control of enabling te program ROM/PROM at this location. Typically the POJM PROM address will be 0000H to provide for system start-up. The IMEM PROM address could be anywhere, though D800H is standard. Since POJM and IMEM are program controllable, group 1 circuits can be removed from or returned to the 8085's memory space.

Group 1:

Whenever IMEM is true, and the address is correct, the timer, RAM, or regular PROM is enabled. The timer and RAM are enabled for both read and write operations. The PROM enable is ANDed with read status in PN16, so that the PROM is enabled only during reads. During writes to the PROM's addresses, the PROM is disabled and the write occurs in whatever memory the system has installed at that address. If POJM is true, the PROM is enabled at its second location (typically 0000-07FF). Again, only PROM reads are decoded.

Group 2:

The serial, parallel, or control port is enabled directly whenever the appropriate address is decoded. The system port output is provided with two DIP switch positions to allow it to enable either the serial or the parallel port. Only one of these switch positions should be activated at a time to avoid bus conflicts of reads from these devices. The switches allow either I/O method to be run from the same "system" port addresses so that software need not be changed when the basic I/O device is changed.

Standard ROM addressing:

The standard addressing for the MPU-B circuits, as effected by the address decode ROM are:

Mapped as memory:

PROM @ POJM	0000H – 07FFH
PROM normal	D800H – DFFFH
RAM	D000H – D0FFH
timer	D100H – D103H

Mapped as I/O ports:

serial	4H,5H and 12H,13H
parallel	(14H,15H) EQUIVALENT
system port	(02H,03H) EQUIVALENT
control port	F3H

13. CONTROL PORT

The control port is used to enable or disable the group 1 circuits (PROM, RAM, and timer), and the second PROM location. The second PROM location is a separate enable at a second address of the same (single) PROM chip, so that it can appear at two places (standard is 0000 at power on jump time and D800 for normal operation).

Whether the second PROM location is enabled is controled by data bit 0 in the control port. The main PROM location, the RAM, and the timer are enabled by data bit 1 in the control port. In both cases, a data bit of "0" enables the circuits, a "1" disables them. These two data bits are each latched into their own 74LS195 shift register when the control port is enabled, by putting the shift registers into load mode. The data bit is latched only into bit A, at the trailing edge of the /WR strobe. After being latched, the data bit is shifted by successive /RD or /WR strobes, until it appears at the output (QD,/QD). This takes three cycles after the control write cycle. After the output takes on its new state, it remains static because the J and /K inputs are tied to load A with QA on each shift. Internal power on clear (/IPOC) is tied to both shift registers' clear inputs, and POJM and IMEM are both taken from /QD, so that at power on time (or after any reset) all the circuits controlled by these signals are enabled.

14. RAM

There are 256 bytes of RAM provided on the board, in two 8111's. The lower 8 bits of address are connected directly to the chip, along with the local bidirectional data bus. The interfacing involves no extra chips since the 8111's were designed to go directly onto a bidirectional bus. Chip select is driven by /REN from group 1 of the address decoder. The output disable pin on the memories is driven whenever either the 8085 read/write status indicates a write cycle, or the front panel (if present) drives data in disable (/DIDSBL) in preparation for a write from the front panel. The write strobe is driven by /MWRT, which is derived from MWRITE whether it is driven by the 8085 or the front panel.

If the address decode ROM addresses any of the RAM as I/O (the standard ROM does not) then the RAM write strobe should be jumpered to /IWR with the jumper provided since MWRT is not active during I/O cycles. This also prevents a front panel from writing directly into the RAM. Note: the same /MWRT - /IWR jumper affects both the RAM and the timer circuits.

15. PROM

Read only program memory on the MPU-B is provided in a single ROM/PROM socket. Three jumpers are provided so that the 24 pin socket can be configured to run a 1K or a 2K prom, or a 1K, 2K, or 4K ROM. The standard chip provided is a 2K ROM with a system monitor and bootstraps.

NOTE: Care must be exercised when installing a ROM/PROM other than the one provided, since it is possible to damage some types of ROM/PROM if the jumpers provided are in the wrong position. Refer to the user's guide.

All the ROM/PROM chips were designed to attach directly to a bidirectional bus, so there is no external logic onvolved. The chip is enabled with /PEN from the address decode logic when the high-order address is appropriate.

16. TIMER

An 8253 chip is used as the timer circuit. No periferal gates are needed since it attaches directly to the bidirectional bus. Address lines A0 and A1 are decoded by the 8253 to select among its 4 control bytes. Since only A8 - A15 are decoded for the timer enable (/TEN), A2 - A7 are "don't care" bits in the address. As a result, the timer's 4 locations appear in multiple locations throughout the block of 256 decoded by the address decode logic. The low four locations only are called out throughout the specifications. The read strobe (/RDL) is a local read strobe generated by anding /DIDSBL with the 8085's /RD. This permits the front panel (if present) to remove the read strobe during a front panel write into the (memory mapped) timer. The write strobe (/MWRT) is derived from MWRITE, whether it is driven from the front panel or the 8085.

If the address decode ROM is changed to include accessing the timer as I/O ports, then the write strobe needs to be jumpered to /IWR at the location provided, since /MWRT does not occur during I/O writes. This will prevent the front panel from writing into the timer, even if the timer is also memory mapped at the same time. Note: The same /MWRT - /IWR jumper affects both the RAM and the timer.

The 8253 has three timers in it. Timer 0 is dedicated to providing a clock signal for the serial interface. The timer 0 out line is connected to the serial interface. Timer 0 gate is pulled high, and timer 0 clock is driven by the 2 MHz CLK2. This timer would normally be programmed to the auto-repeat square wave mode, although the system can use it for other timing functions at times when the serial interface is not needed.

Timers 1 and 2 are undedicated and available for system use. Timer 1 is clocked by the 2 MHz CLK2 line, while timer 2 can be switch selected to either run at 2 MHz, or cascaded from the output of timer 1. In the cascaded mode, the total count time of the two timers is over 30 minutes. The outputs of both timer 1 and 2 can be switch selected to cause an interrupt at terminal count (or periodically). These lines were run to RST7.5 (which is edge sensitive) to permit use of modes with short terminal output pulses. The gates on both timers 1 and 2 are pulled high.

The gate and output lines of all three timers are brought to solder-pads to permit easy modification of the board for custom applications where it is desired to drive these lines with external signals.The timers' function is not affected by whether its address decoding is enabled (IMEM). They will continue to count and produce interrupts as programmed until they are re-enabled and the mode changed. The interrupts are driven through open-collector gates so that other devices may also be connected to the same interrupt line if desired.

17. PARALLEL I/O

Data:

Two 8212's are used for the parallel I/O data path. One latches output data and drives the parallel output pins on J4 continuously. The other will buffer the input data and will also latch it if the strobe line is driven by the input device. Both the input and output data paths can use the handshaking available in the 8212, and the input data ready signal or the output buffer empty signal can both switched onto the interruppt pins for interrupt operation of the parallel port. The same two sgnals are available at the parallel I/O status port. The handshaking only runs when the respective external device drives the strobe lines. If the strobe lines are not driven, the input data ready and the output buffer empty will always indicate false. When handshaking is used, the failing edge of the strobe causes the respective status bit to go true, and latches the data on the input port. The processor's response (read or write data) causes the respective status bit to return false. The 8212 chips are selected by /KEN and A0=0, as well as the appropriate read or write strobe.

Status:

The parallel I/O status port is implemented with four sections of a 74LS367 to drive the internal data bus, and some gates to enable the 3-state buffers with /KEN=0 and A0=1 and /RDL=0, which is during the read strobe to the port address decoded at the address decoding. The output port output buffer empty signal is connected to data bit 0. The input port input data ready isconnected to data bit 1. These are the same bits as the respective status signals from the 8251 serial interface chip, to simplify software drivers which run both devices. Also driven during parallel status reads are two upper data bits. Bit 7 is driven with IMEM, and bit 6 is driven with POJM. These two bits are high when the respective on-board functions are enabled. This permits interrupt routines to determine the state of the board so they can return correctly.

Power:

Ground, +5 volts regulated, +18 volts unregulated, and -18 volts unregulated are provided on the parallel I/O connector. The +5 volt regulator was designed to be able to supply up to several hundred milliampers of power to the parallel connector, however it should be realized that any power drawn results in higher dissipation at the regulator. If too much current is drawn or to little ventilation provided, the +5 volt regulator may shut itself down.

Connector:

The parallel I/O connector is a 26-pin dual-row header, suitable for; connection to flat cable connectors. The pin definitions result in the same signal/wire connections as used in the MIO. Pin 26 is unused to permit the use of 25 wire cable and D type 25 pin connectors at the rear of the cabinet. The schematic shows the 26 pin connector pin-numbers in the arrow boxes, and the 25 pin connector pin-numbers in the circles. These pin numbers assume the standard pin numbering for D type connectors, standard flat cable pin numbering for the 26 pin connector (so that the flat cable wires counted from the edge correspond to the pin numbers), and a cable wired to connect pin 1's.

18. SERIAL I/O

The serial I/O port uses an 8251 serial interface chip.This USART will support asynchronous or synchronous serial communications to 9600 or 56000 baud respectively. It is interfaced to standard EIA RS-232 levels and pins for providing either the computer or terminal end of a serial line. It also will drive an ASR-33 Teletype type 20 mil. current loop interface. The output buffer empty and the input data ready or sync. detect status bits can be switched to drive the interrupt lines.

The 8251 is connected directly to the on-board bidirectional bus. A0 drives the command/data input, the other 7 address bits are decoded at the address decoder to form /SEN. The clock input is driven with 2MHz, and power on clear and on-board read and write strobes drive the appropriate inputs.

The EIA RS-232 signals from the USART (including RTS, CTS, DTR, DSR, R and T Data, R and T synchronous clocks) are connected to the serial I/O connectors through RS-232 receivers and drivers. DIP switches permit selecting the on-board receive clock for asynchronous mode, or an off-board clock for synchronous data mode. DTR and CTS inputs are tied true with resistors to eliminate the need for jumpers when they are not driven. Carrier detect is tied true.

The current loop interface will switch and sense a 20 mil. loop compatible with the ASR-33 Teletype. The loop current source is provided, and any other current source should be removed. The current loop connections must not be referenced to ground. Provision for filtering the input data is made to permit reliable operation of the typically noisy ASR-33.

Serial connectors:

There are two serial I/O connectors on the board. They are both 26 pin dual row connectors suitable for flat cable connection to the EIA standard D type 25 pin connectors. Assuming a cable wired so that pin 1's on the two ends are connected, the 26 pin connector pin-numbers are indicated on the schematic in the arrow boxes, and the EIA standard RS-232 pin numbers

are indicated in the circles. The two connectors are arranged so that one is configured as a computer end of an RS-232 line, and the other is configured as a terminal end. Depending on the use of the system, the cable from the RS-232 line should be plugged into one connector or the other, with no jumpers being necessary. The board is not designed to use both connectors at once.

The current loop connections are made to the computer end connector only. If for some reason it is desired to operate in current loop mode as a terminal, a custom cable can be made to effect the proper connections.

Since the S-100 bus does not include a chassis ground, the chassis ground line from the RS-232 connectors is terminated in a .040" pin socket. If it is desired to connect this line at the board, a wire with a .040" pin can be attached to the chassis, and plugged into this socket.

19. POWER REGULATORS

Plus and minus 12 volts for the RS-232 drivers, pullups, and the PROM/ROM are regulated by TO-220 style three terminal regulators. They regulate only a small amount of current, so they are heat-sunk on the foil on the board. Minus 5 volts for the PROMs is regulated by another TO-220 regulator in series with the -12 volt regulator, to more evenly distribute the heat dissipation.

The plus 5 volt regulator is a TO-3 style three terminal regulator, mounted on a heatsink. It is required to supply the 5 volts for this board, plus any additional current taken by an external device connected to the parallel I/O port. The MPU-B 5 volt current drawn is typical 1.3 amps, maximum 1.6 amps.

Tantalum capacitors are provided at the inputs of each regulator for stability, and .1 uF ceramic bypass capacitors are provided throughout the board among the logic chips.

B. USER GUIDE

In this section we describe how to set up and use your MPU-B processor board. The level of detail is intended to supply the assembly language system programmer and OEM user with all the information needed to use the board. There are several option jumpers which must be changed if you are going to operate the MPU-B differently than set-up at the factory. You will probably not want to change these jumper options. There are a number of switch-options for functions which are often set differently in separate systems. These should be checked before operation to insure that the desired mode of operation is enabled. If your MPU-B has arrived as part of a factory-assembled system, it will be set-up to run. In that case you may wish to delay referring to the option set-up portion of this chapter until you wish to change the factory settings.

For normal operations, an operating system program, a high-level language, or an applications program is loaded into the computer from mass storage. The procedure for initializing the system and loading the desired program is described in the Sections IV and V of the PCS-80/30 User Manual. The option selection and machine-language operation of the MPU-B board only is described here. This section is primarily of use to those programming in Assembly or machine language, and during initial installation.

1. MPU-B OPTION JUMPERS

Some of the options provided on the MPU-B will be used only in special situations, such as when a system is being customized for a special configuration or special application. These options have been provided as jumper-pads on the printed-circuit board, and they have a small trace on the board connecting these pads in the standard factory configuration. Unless you have a special situation, you will not need to set or change these jumpers.

Three of these jumpers allow configuring the MPU-B program ROM socket to accept different size ROM's. Assembled boards are jumpered at the factory for the ROM installed. Other options are described in the Program ROM/PROM section. Note: INCORRECT JUMPERS CAN DAMAGE THE ROM, refer to the appendix on PROM installation.

There are jumpers to remove bus drive from the signals SSTACK, INTE, and MWRITE. These jumpers appear on the schematic but are not described since they are primarily for factory use. Should a knowledgeable user wish to change one of these and need information, he should contact the factory.

A .040" pin socket is provided for connecting RS-232 line AA (chassis gnd.). Solder pads for wires or pin sockets are provided for 8085 signals SID and SOD, as well as timer signals GATE and OUT for all three timers.

Jumper pads permit connecting any of the eight vectored interrupt lines on the bus to any of the four restart inputs or the interrupt request line on the 8085. The standard configuration is described in the USING INTERRUPTS section (II-B,8).

2. MPU-B OPTION SWITCHES

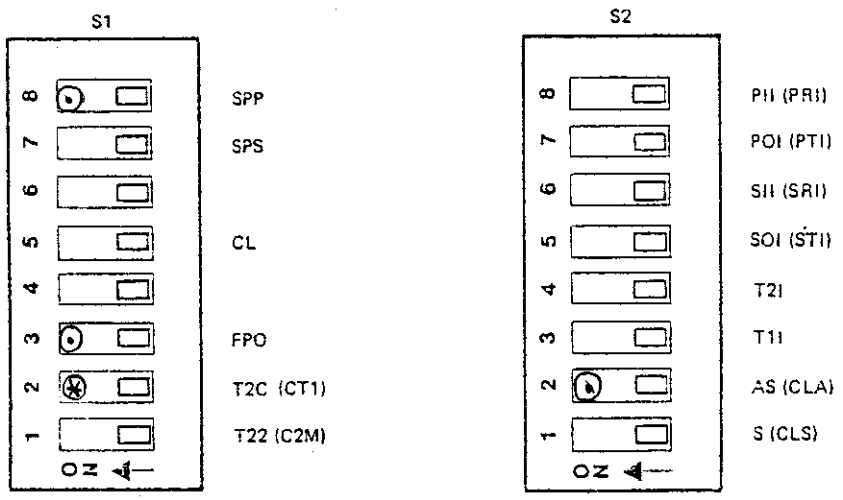
All the options which might normally be changed are provided with switch positions to enable the board to be quickly and easily configured as desired. A brief description of the switches is provided here for reference.

⊙ SHOWS CONFIGURATION AS DELIVERED. SWITCHES ON

⊛ CHANGE - RLV

FIGURE II-1

SWITCH LOCATIONS
TOP OF BOARD



OPTION SWITCHES

SYSTEM PORT SERIAL ENABLE (SPS IN/OUT)
SYSTEM PORT PARALLEL ENABLE (SPP IN/OUT)

Only one of this pair of switches should be enabled at a time. They cause either the serial or parallel I/O port (or neither) to respond to the system port addresses (02 data, 03 status). (They will continue to respond to their standard port addresses.) If both switches are disabled, then ports 00 through 03 are unused, permitting an external I/O board (such as an MIO) to be installed using the bottom 4 ports.

ASYNCHRONOUS CLOCK ENABLE (CLA IN/OUT)
SYNCHRONOUS CLOCK ENABLE (CLS IN/OUT)

Only one of this pair of switches should be enabled at a time. One of these must be down to enable use of serial receive. They connect either the internal baud rate clock (for asynchronous mode) or an externally generated clock signal (for synchronous mode) to the USART's receive clock input.

CURRENT LOOP ENABLE (CL IN/OUT)

If current loop operation is desired, the CL switch should be enabled. If synchronous mode is to be used, then the CL switch should be disabled.

SERIAL TRANSMIT BUFFER EMPTY INTERRUPT (STI IN/OUT)
SERIAL RECEIVE CHAR. READY INTERRUPT (SRI IN/OUT)

Neither, either, or both of these may be enabled as desired. The first will cause an interrupt to occur (RST6.5) whenever the serial I/O transmit is enabled and the transmit buffer is empty. A jump to the interrupt service routine must be located at 0034H. The second will cause an interrupt to occur (RST5.5) whenever the serial I/O receive is enabled and a character is waiting to be input to the processor. A jump to the interrupt service routine must be located at 002CH.

PARALLEL TRANSMIT BUFFER EMPTY INTERRUPT (PTI IN/OUT)
PARALLEL RECEIVE CHAR. READY INTERRUPT (PRI IN/OUT)

Neither, either, or both of these may be enabled as desired. The first will cause an interrupt to occur (RST6.5) whenever the parallel I/O output buffer empty status is true (set by handshake strobe). A jump to the interrupt service routine must be located at 0034H. The second will cause an interrupt to occur (RST5.5) whenever the parallel I/O input data ready status is true (set by the same strobe which latches data). A jump to the interrupt service routine must be located at 002CH.

TIMER 1 INTERRUPT ENABLE (T1I IN/OUT)
TIMER 2 INTERRUPT ENABLE (T2I IN/OUT)

Neither, either, or both of these may be enabled as desired. Each switch will cause an interrupt to occur (RST7.5) when its respective timer output goes low (terminal count or 1/2 terminal count depending on programmed mode). A jump to the interrupt service routine must be located at 003CH. Timer 0 is the baud rate generator, and does not connect to the interrupts. Note that RST7.5 is an edge triggered interrupt, so that timer modes which produce a terminal pulse may be used without danger of missing the interrupt.

TIMER 2 CLOCK 2 MHz ENABLE (C2M IN/OUT)
TIMER 2 CLOCK T1 OUT ENABLE (CT1 IN/OUT)

Only one of these two switches should be enabled at a time. One or the other must be down to provide a clock input to timer 2. When the first one is enabled, timer 2 counts at a 2 MHz rate (2 million counts/second). With the second switch enabled, timer 2 increments one count at each high to low transition of the output of timer 1. Cascading the timers this way and enabling only timer 2 interrupt permits extended periods to be timed without software intervention.

FRONT PANEL INSTALLATION (Panel IN - FP1)
(Panel OUT - FP0)

This switch must be placed in the enabled position in any system with a CP-A installed, and must be placed in the disabled position in any system without a CP-A. Even when temporarily inserting a front panel for maintenance, this switch should be moved to IN. Note that the sense of IN and OUT is reversed from the other switches.

3. ADDRESS DECODING AND CONTROL

There are six circuits on the MPU-B which appear in the memory or I/O space of the processor. They are:

1. Firmware ROM
2. 256 byte RAM
3. Three TIMERS
4. 8 bit parallel I/O port
5. Serial I/O port
6. MPU-B control port

Some of the six circuits can appear in multiple (redundant) locations in memory or I/O space. Some can be made to appear/disappear from memory space using the control port.

For descriptive purposes, the functions are divided into three groups according to how the sections are enabled/disabled by the control port. The allocation of memory - I/O space to these six functions is as follows:

GROUP	BYTES	FUNCTION	ADDRESS
0	2K	ROM for system initialize	0000-07FF
1	2K	ROM for monitor functions	D800-DFFF
1	256	RAM for monitor variables	D000-D0FF
1	4	TIMERS & serial clk.	D100-D103
1		RESERVED do not use	D104-D7FF
2	2	PARALLEL I/O	14-15
2	2	SERIAL I/O	12-13
2	2	SERIAL I/O	04-05
2	2	SYSTEM PORT for SER. or PAR.	02-03
2	1	CONTROL PORT for address mode	F3

The four digit numbers refer to hexadecimal memory addresses, and the two digit numbers refer to hexadecimal port addresses. The ROM in groups 0 and 1 is actually the same ROM which can appear at two different locations. The serial and parallel I/O can also appear at multiple locations.

CONTROL PORT AND SOFTWARE MEMORY CONTROL

The MPU-B control port is a single I/O mapped port address (F3). It is a write-only port and only two of the eight data bits are active. The two bits control whether group 0 or 1 circuits appear in the memory space or not. When they are 0, the circuits appear in the memory space; when they are 1, the circuits do not appear in the memory space and whatever memory is installed in the system at that location then appears.

Control port:

Group 0 control: bit 6

BIT 6 = 0 2K ROM appears at 0000-07FF
BIT 6 = 1 System memory (as installed) appears at 0000-07FF

Group 1 control: bit 7

BIT 7 = 0 2K ROM appears at D800-DFFF
256 byte RAM appears at D000-D0FF
TIMERS appear at D100-D103

BIT 7 = 1 System memory (as installed) appears at D000-DFFF

CONTROL PORT DEFAULT AND SWITCHING RESTRICTIONS:

At power on time and at any reset, control port bits 6 and 7 are both set to 0 so that the firmware ROM, RAM, and timers are all enabled. The two control bits operate independantly, and there are no restrictions on acceptable combinations. Either group may be enabled or disabled at any time or sequence desired. The control bits are changed with an out F3.

If a read is attempted from the control port, the resulting data will be FF. Due to the logic implementation, this will also result in both groups 0 and 1 being disabled. That is, a read from the control port is equivalent to a write with data bits 6 and 7 equal to 1's.

CONTROL PORT EFFECT:

When the 2K ROM is enabled, all memory reads from those locations read from the ROM. Thus the program in the ROM can be executed. All memory writes from those locations, however, will succeed in writing data to the system memory (if installed).

Both the RAM and timers are read/write devices. When they are enabled, all memory reads and writes from those locations are from/to the RAM and timers. During the time they are enabled, no access is possible to system memory at those locations.

The combination of devices and addresses labeled group 1 above cannot be enabled/disabled separately. They can only be switched in and out as a group. As a result, when group 1 is enabled, system memory from D800-DFFF is accessible for writes only, and system memory from D000-D7FF is not accessible at all.

When group 0 (firmware ROM) is enabled, system memory at 0000-07FF is not accessible for reads, since reads now come from the firmware ROM. Writes, however, are not restricted from system memory at all by group 0 being enabled. If group 1 is disabled to remove its write restrictions, group 0 can be enabled and still permit writing to the entire 65K memory space. Thus a loader or other system initialization routine executed from the firmware ROM can load/initialize the entire memory as needed.

POWER-ON JUMP

When the MPU-B is run without a front panel, it will come up running starting at location 0000 at power-on time or reset time. Since the firmware ROM defaults to being enabled at the same time, the MPU-B will begin to execute the code stored in the ROM at 0000. The code contained in the ROM can range from a disable group 0 and jump to monitor sequence, to a complete automatic system initialize and load from external mass storage, ending with a disable firmware ROM and jump to system start.

The factory installed ROM functions vary for assembled systems. The ROM with kits or assembled boards sold separately is the same as that supplied with the PCS-80 system model 30 unit. The factory installed ROM can be 1K bytes or larger, depending on the system.

CONTROL PORT ENABLE / DISABLE TIMING:

When either group 0 or 1 is either enabled or disabled by program control, there is a 3 CYCLE DELAY before the commanded change in status takes place. This is to allow for a 3-byte instruction fetch, such as a jump or call, before the ROM appears/disappears. If a 3-byte instruction is not desired, the programmer must take precautions to allow for the status change being delayed 3 cycles. The delay is for 3 true 8085 cycles, and is not affected by any DMA cycles which may occur during this time. Interrupt cycles, however, will be counted, so if the 3-byte instruction is in 0000-07FF or D800-DFFF, then the programmer needs to insure that no interrupt will occur. Otherwise the instruction fetched will not be what the programmer

expected.

Example of a POWER-ON JUMP to location E000

ROM ADDR.	CONTENT		Notes:
0000	MVI	A,0C0H	Get byte to turn off group 0 and 1
0002	OUT	F3	Output byte to control port
0004	JMP	E000	Jump to location E000 (Group 0 and 1 will both become disabled after reading all the jump and before the fetch following the jump.)

CONTROL PORT STATUS

The current status of the group 0 and 1 enable can be read by the processor. It appears in the status byte from the parallel port, bits 6 and 7. A 0 indicates enabled, and a 1 indicates disabled. The capability to read this status permits a system to use interrupt routines which can sense the system status, modify it as necessary to service the interrupt, and return it to its original state before returning. The function of the status bits is the same as the function of the control bits. A status read from the parallel pot address followed by an output of the result to the control port will result in no change.

MEMORY STATUS SENSE (in parallel I/O status byte)

Bit 6 = 0 ROM at 0000 enabled

Bit 6 = 1 System memory 0000-07FF enabled

Bit 7 = 0 ROM at D800 enabled
RAM at D000 enabled
Timers at D100 enabled

Bit 7 = 1 System memory D000-DFFF enabled

SERIAL and PARALLEL I/O PORT ADDRESSING

The parallel port is always accessible at I/O port 14 (data) and I/O port 15 (status). If the option switch SPP (System port parallel) is enabled, then the parallel port also appears at I/O port 2 (data) and 3 (status). When it is enabled at both locations, accessing it by either set of addresses or a mixture of the two produces identical results. For example, it would make no difference if data was written to port 02 or port 14, whether the status was read from port 03 or port 15.

The serial port is always accessible at I/O port 12 (data) and I/O port 13 (status) as well as always available at I/O port 04 (data) and 05 (status). If the option switch SPS (System pi serial port also appears at I/O port 02 (data) and 03 (status). As in the parallel port, there is no differentiation between the multiple ports at which it may be accessed.

The port addresses 12,13,14,15 are the same configuration of data and status as would be presented by an SIO 2-2 board addresses at 10H. The port addresses 02,03,04,05 are the same configuration as an SIO 2-2 board addressed at 00.

The typical consol I/O addresses for IMSAI software are 02 and 03. The MPU-B permits switching either the serial port (for a terminal) or the parallel port (for a keyboard to be used with a video display) to respond to these locations. A serial device is expected at ports 04 and 05 by some IMSAI software. New firmware intended for the MPU-B alone would use the permanent addressing at 12,13,14 and 15.

4. PROGRAM ROM / PROM

The firmware ROM supplied with the MPU-B will vary for different systems. For details of functions included in the ROM, the appropriate USER MANUAL should be consulted. For boards purchased separately, the ROM included is that used in the PCS-80 model 30. It provides system initialization and loaders with a limited monitor.

For a system modified for a special application, the firmware ROM may be replaced with a PROM or a ROM containing firmware to support that application. This would be done if the system was not going to be used as a general purpose machine, but rather for only the special application. Since complete system initialization, load, and error recovery can be put in the PROM, special application systems may be easily made which involve no operator actions other than the power switch to begin operation.

Jumper options permit the use of 1K (2708 type) or 2K (2716 type) Eraseable PROM's, and also 1K, 2K, or 4K ROM's for systems produced in higher quantities.

CAUTION: Factory jumpers have been installed for the ROM supplied. Other PROM's or ROM's may be DESTROYED if installed without changing the jumpers. Refer to Appendix 2.

When a 1K PROM/ROM is installed without changing the address decode ROM, it occupies 2K of memory space. It will appear in both the upper and lower halfs of the 2K space reserved for the 2K ROM. If a 4K ROM is installed, the address decode ROM must be changed in order to permit access to the upper half of the 4K ROM. Contact the factory for details.

ADDRESSING:

The ROM/PROM can appear in two locations. It will appear at 0000 when group 0 is enabled (see CONTROL PORT section above). It will also appear at D800 when group 1 is enabled.

This is not two different ROM's, but rather it is the same ROM appearing in two different locations. It can appear at both simultaneously if both groups 0 and 1 are enabled. In that case a read from 0000 will get the same data as a read from D800. A read from 01E4 will get the same data as a read from D9E4, etc.

The programs run from each area (0000-07FF and D800-DFFF) must share the space in the ROM. The program running in 0000-07FF will need to be at the bottom of the ROM to start properly at location 0000 at power-on time. A program running in the D800-DFFF area would then be ORG'ed at D800 plus the length of the program at 0000.

For example, a system initialization and boot from floppy disk routine may take 140H bytes. It would be ORG'ed at 2000 and placed in the bottom 140 bytes of the ROM. A set of standard utilities can then be put in the ROM to run at the upper location. It would be ORG'ed at D940H and could extend up to DFFF. It would be placed in the ROM following the initialization and boot routine.

The ROM may be used at either location at any time, but the two locations have different facilities. Running from only 0000 permits a routine to have WRITE access to the entire 65K memory space of the 8085 for complete system initialization. Loader variable storage is limited to the registers in the processor or reserved system memory, however. When running from D800-DFFF, WRITE operations are restricted from D000-D7FF, but 256 bytes of RAM are available (D000-D0FF) which are not part of the system memory and are switched out when the ROM is switched out. For a routine needing complete access as well as larger variable storage, it is possible to run the routine at 0000-07FF, enable the RAM / upper ROM location before variable access, and disable the RAM / upper ROM location before accessing system memory at D000-DFFF.

5. RAM MEMORY

There are 256 bytes of read/write memory on the MPU-B, which can be enabled or disabled as part of group 1 circuits. When it is enabled, it is at D000-D0FF. The upper part of the RAM area is used for variable storage by the monitor program in the ROM, and the rest of the RAM area is free for user storage or programs. If the monitor is not to be used, nor any routines called in the ROM, then the entire 256 bytes may be safely used.

If the RAM has been disabled (through the control port), then it is not available for use until re-enabled. While it is disabled, whatever memory is loaded into the system at that location responds to the processor; when it is enabled, the memory in the system at the RAM address is disabled and the RAM on the MPU-B responds to the processor. Any time the program enables or disables the RAM, there is a 3 CYCLE DELAY before the status change takes place, to allow for a jump or call if desired. The RAM is enabled/disabled along with the other functions located in DXXX. Its enable is not separable.

Information is not lost in either the 256 bytes of RAM or in the system memory it blocks as the program enables or disables it. At power up time or on reset, the RAM is enabled.

6. SERIAL I/O PORT

The MPU-B has a serial I/O port implemented with an 8251 USART. Both synchronous and asynchronous operation are possible. It is configured to allow both current loop and RS-232 operation with no jumper changes. The baud-rate clock is obtained from a TIMER section under software control. For details on the clock generation, see the TIMERS section (II-B,7).

CURRENT LOOP

The current loop interface provides 20-mil current loop including the current source. It is suitable for ASR-33 teletypes and devices with similar interfaces.

The MPU-B current source is referenced to ground. The interface to which it is connected must be isolated from ground to avoid possible damage.

The normal internal configuration for an ASR-33 teletype is with no current source and isolated from ground. However, it is possible that a unit would be jumpered to provide a current source. Before connecting a unit to the MPU-B, it should be checked to insure that the interface lines are isolated from ground. This can be done easily by measuring the resistance between the interface lines and the chassis in the ASR-33. A standard VOM is adequate, and the resistance measurement should indicate an open circuit.

The MPU-B current loop interface operates full duplex. The ASR-33 jumpers should be configured for full duplex if they are not already configured that way.

DIRECTIONS FOR JUMPERING

There is a terminal strip located at the right rear of the Teletype (ASR33). The terminal strip is behind a panel of square white plastic connectors and also connects to the TTY power cord. The terminals are numbered from 1 to 9. The connections required between the MIO and these terminals are shown in Table II-1. In addition to making these connections, it may be necessary to perform the following operations on your Teletype.

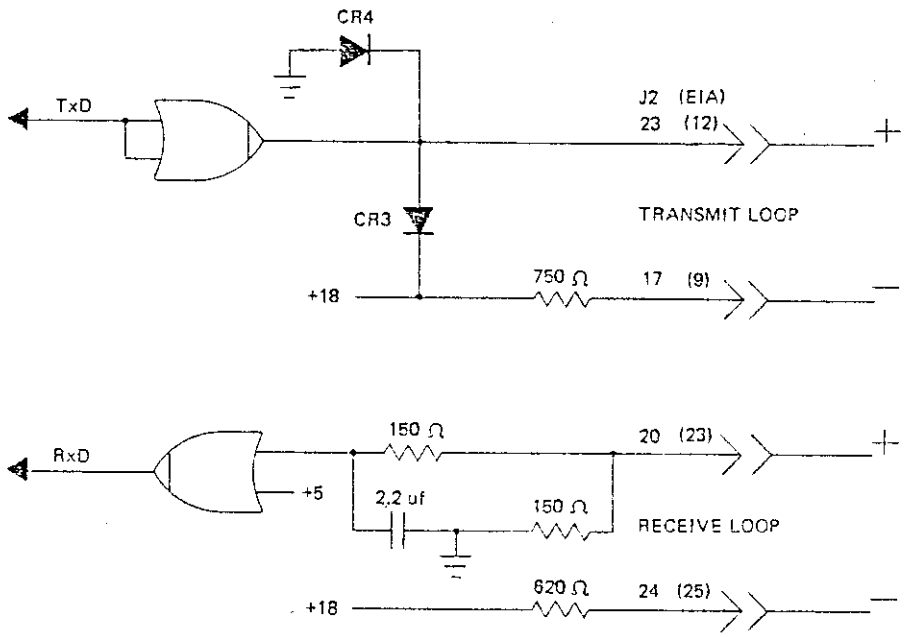
1. Full Duplex Operation - Move YEL/BRN wire from Terminal 3 to Terminal 5 and move WHT/BLU wire from Terminal 4 to Terminal 5.
2. Change receiver current level from 60 ma to 20 ma; move VIO wire from Terminal 8 to Terminal 9.
3. Change current source resistor to 1450 ohms. Locate the current cource resistor in front of the power supply and move the BLU wire to the tap labeled 1450.

TABLE II-1 CONNECTIONS FOR ASR-33

Signal Name	26 Pin Edge Connector	25 Pin EIA Connector	Terminal Strip
Current Loop Out +	20	23	7
Current Loop Out -	24	25	6
Current Loop In +	8	17	3
Current Loop In -	22	24	4

FIGURE II 2

CIRCUIT FOR CONNECTIONS TO ASR-33



The serial interface lines are brought to the rear of the cabinet through a flat cable to a standard 25 pin D type connector. The MPU-B end of the cable should be plugged into either J2 or J3. The current loop pins on J2 are configured the same as those on the SIO board, while the current loop pins on J3 are configured the same as those on the MIO board. On J2, the current loop pins are numbers 9 (+) and 12 (−) for the output loop, and numbers 25 (+) and 23 (−) for the input loop. An external cable should connect these pins to the ASR-33, attaching the wires according to the diagram above. On J3, the corresponding pins are 23 (+) and 25 (−) for output and 17 (+) and 24 (−) for input. Unless a cable wired for the MIO current loop pin configuration is already available, it is suggested that the SIO configuration on J2 be used.

OPTION SWITCHES

To run current loop, option switch CL must be in the enabled (IN) position. Current loop mode may only be used with asynchronous mode, for which switch CLA must be enabled, and switch CLS must be disabled.

RS-232

Devices with RS-232 interfaces can be plugged directly into the 25 pin D type connector at the rear of the cabinet. The MPU-B end of the cable can be plugged into either J2 or J3 for different configurations. It should be plugged into J2 (Computer or modem end configuration) to run terminal type devices, or into J3 (Terminal end configuration) to run with a modem.

Handshaking and clock lines are included in the RS-232 interface as well as send & receive data. They are terminated so that no jumpers are needed whether or not the interface to be driven implements these lines. The lines implemented on the MPU-B are:

RS-232 INTERFACE LINES

PIN	LINE	NAME	FUNCTION
1	AA	CG	Chassis ground
7	AB	SG	Signal ground
2	BA	TxD	Transmit data
3	BB	RxD	Receive data
4	CA	RTS	Request to send
5	CB	CTS	Clear to send
6	CC	DSR	Data set ready
20	CD	DTR	Data terminal ready
8	CF	CD	Carrier detect
15	DB	TxC	Transmit clock
17	DD	RxC	Receive clock

The RS-232 signal names and functions refer to the operation from the terminal end of the line. For example, TxD is data transmitted from the terminal device to the computer device. Thus pin 2 is an output on a terminal device, and an input on a computer device.

All the signals named above are fully implemented except for carrier detect, which is driven true in computer configuration but cannot be sensed for use in a terminal configuration, and chassis ground, which can be jumpered to ground or connected to chassis ground with an external wire and a .040 pin. Only transmit and receive data are required to operate the RS-232 interface in asynchronous mode; and only transmit and receive clock need to be added to operate in the synchronous mode. The other lines can be left unterminated or connected to the peripheral device as desired.

OPTION SWITCHES for RS-232

No jumpers are needed to configure the interface for any situation. However, there are three option switches which must be in the correct position.

Option switches CLA and CLS configure the clock lines to run the interface either asynchronous or synchronous. If it is desired to run the interface in the asynchronous mode, then switch CLA should be enabled (IN) and switch CLS should be disabled (OUT). This connects the clock in the receive section to the internal baud rate generator. If operation in the synchronous mode is desired, then switch CLS should be enabled (IN), and switch CLA should be disabled (OUT). This connects the receive clock to the external (RxC) clock line. The transmit clock is always connected to the internal baud rate generator.

Option switch CL only applies if operation in the synchronous mode is desired. For synchronous mode, make sure switch CL is disabled (OUT).

PROGRAMMING THE SERIAL INTERFACE

GENERAL

The serial interface is capable of supporting virtually any serial data technique currently in use with RS-232 or ASR-33 type current loop, including IBM bi-sync. Before data can be sent and received through the interface, the mode and baud rate must be initialized through software. Refer to the TIMER section for details on initializing the baud rate clock. Programming the mode and data operations are described in this section.

SETTING THE MODE

The mode byte output to the 8251 defines the general operational caracteristics of the serial interface. It MUST follow a reset, either a power on reset, system reset (front panel button), or a software reset. The first byte output to the control port address of the 8251 (same as the status read port address) will be interpreted as a mode byte.

If a change in the operational mode is desired under software control, the 8251 may be reset by the following sequence:

```
SUB     A,A     Set accumulator to 00
OUT     13      Output 00 to 8251 control port
OUT     13      Second time
OUT     13      Third time
MVI     A,40    Put 40 into accumulator
OUT     13      Output to 8251 to reset
```

Port 05 could be used, as could (with option switch SPS enabled) port 03. Outputting three zeros is done to insure that even if (worse case) the 8251 is expecting the first of two sync bytes, the programming sequence is taken far enough that the chip will for sure interpret the 40 as a command byte. A command byte of 40 will reset the 8251, and must be followed by a mode byte.

The mode byte for asynchronous operation is defined as follows:

BIT 7	BIT 6	BIT 5	BIT 4	BIT 3	BIT 2	BIT 1	BIT 0
S2	S1	EP	PEN	L2	L1	B2	B1

BAUD RATE FACTOR, B2 & B1:

B2	B1	Function
0	0	Synchronous Mode
0	1	1 X clock rate
1	0	1/16 X clock rate
1	1	1/64 X clock rate

CHARACTER LENGTH, L2 & L1:

L2	L1	Character length (data bits)
0	0	5 bits
0	1	6 bits
1	0	7 bits
1	1	8 bits

PARITY ENABLE, PEN:

PEN	Function
0	Parity disabled
1	Parity enabled

PARITY SENSE, EP:

EP	Function
0	Odd parity
1	Even parity

STOP BITS, S2 & S1:

S2	S1	Function
0	0	Invalid code, do not use
0	1	1 stop bit
1	0	1.5 stop bits
1	1	2 stop bits

An asynchronous mode byte which will enable running with a model 33 TELETYPE and most CRT terminals is AE. The normal baud rate factor used is x16, and this is assumed in the TIMER section on setting the baud rate.

For synchronous mode the mode byte is defined as follows:

BIT 7	BIT 6	BIT 5	BIT 4	BIT 3	BIT 2	BIT 1	BIT 0
SCS	ESD	EP	PEN	L2	L1	0	0

Bits 0 through 5 are the same as defined under asynchronous.

EXTERNAL SYNC DETECT, ESD

ESD	Function
0	Sync is detected by the 8251 internally
1	An externally detected sync is expected

SINGLE CHARACTER SYNC, SCS

SCS	Function
0	Double sync character mode
1	Single sync character mode

Note that the sync detect pin is pulled true, so that if external sync detect is programmed, the 8251 is told immediately that sync has been detected, regardless of the data. The normal setting for this option will be for internal sync detect.

Either one or two sync characters, as programmed by the mode byte, must be sent to the command port following the mode byte. If the two sync characters are different, the first one given to the 8251 should be the first one expected on the data line.

COMMAND BYTE

Once the mode byte and sync bytes, as required, have been output to the command port, the 8251 will interpret all further bytes to the command port as command bytes until the chip is reset.

The command byte is the same for synchronous and asynchronous, and is defined as follows:

BIT 7	BIT 6	BIT 5	BIT 4	BIT 3	BIT 2	BIT 1	BIT 0
EH	IR	RTS	ER	SBRK	RxE	DTR	TxEN

TRANSMIT ENABLE, TxEN

TxEN	Function
0	Transmit section disabled
1	Transmit section enabled

DATA TERMINAL READY, DTR

DTR	Function
0	RS-232 line DTR set FALSE
1	RS-232 line DTR set TRUE

RECEIVE ENABLE, RxE

RxE	Function
0	Receive section disabled
1	Receive section enabled

SEND BREAK CHARACTER, SBRK

SBRK	Function
0	Normal send operation
1	Transmit data held continuously true

ERROR RESET, ER

ER	Function
0	No effect
1	Reset error flags PE, OE, and FE

REQUEST TO SEND, RTS

RTS	Function
0	RS-232 line RTS set FALSE
1	RS-232 line RTS set TRUE

INTERNAL RESET, IR

IR	Function
0	No effect
1	Resets 8251, Next command byte must be mode

ENTER HUNT MODE, EH

EH	Function
0	No effect
1	Enables search for sync character(s) in data stream

In asynchronous mode, the normal command byte desired is 37. Once this command is given, no further writes to the command port are normally needed. The serial port will send and receive data through the data port, signalling the processor that it is ready for the next byte through a status read from the command port (or through an interrupt if the option switches are enabled).

In synchronous mode, the normal command byte desired is B7. The serial port will begin scanning the incomming data for a match to the sync character(s). Once a match is found, the status or interrupt will indicate as each character is received. Between each message in synchronous mode, it is often desired to re-synchronize. This can be done by sending another command byte (B7) to the command port to cause the 8251 to re-enter the sync hunt mode.

STATUS BYTE

The status byte can be read by the program at any time, by an INPUT from the command port address. It contains the transmit and receive ready bits, error flags, and other status information needed for the program to interface to the serial port.

The status byte is defined as follows:

BIT 7	BIT 6	BIT 5	BIT 4	BIT 3	BIT 2	BIT 1	BIT 0
DSR	SYNDET	FE	OE	PE	TxE	RxRDY	TxRDY

TRANSMITTER READY, TxRDY

TxRDY	Function
0	Transmitter buffer full, waiting for transmission
1	Transmitter buffer ready to accept next character

RECEIVER READY, RxRDY

RxRDY	Function
0	No further characters received yet
1	Next received character ready to be input to program

TRANSMITTER EMPTY, TxE

TxE	Function
0	Character currently being sent by USART
1	Last character transmission has been completed

PARITY ERROR, PE

PE	Function
0	No parity error has occurred since the last reset command (bit ER in the command byte)
1	A parity error has occurred since the last reset (PE does not inhibit further operation of the USART)

OVERRUN ERROR, OE

OE	Function
0	No character has been lost since last reset command
1	The program has failed to read a character before the next one was received. Further operation is not inhibited; however, the overrun character is lost.

FRAMING ERROR, FE

FE	Function
0	No framing error has occurred since the last reset
1	A valid stop bit was not detected on at least one character since the last reset. FE does not inhibit further operation.

SYNC DETECT, SYNDET

SYNDET	Function
0	No sync byte(s) has been detected since the last status read (or reset operation).
1	A valid sync byte(s) has been detected. This status bit is reset by a status read.

DATA SET READY, DSR

DSR	Function
0	RS-232 line DATA SET READY is false
1	RS-232 line DATA SET READY is true

EXAMPLE INTERFACE ROUTINES

The following instruction sequence will set up the USART for asynchronous X-16 clock, 8-bit characters without generating or checking parity. This mode will run an ASR-33 or any standard CRT terminal. The sequence should be executed after a reset (either hardware of software as shown above) before any other I/O to the serial port is executed. It leaves both the transmitter and receiver sections running, and for normal operation no further mode or command operations need take place.

```
MVI     A,AE    Get mode byte
OUT     13      Output to serial port command address
MVI     A,37    Get command byte
OUT     13      Output to serial port command address
```

The next routine can be called each time another input character is desired. NOTE: Only one character is stored, so if the program asks for characters slowly enough that the operator gets ahead more than one character, then characters will be lost. If too much processing is needed for fast enough operation, then a whole line can be accepted from the operator at once. The

program can then process the input line, prompting the operator when it is ready for the next line. Alternatively, interrupt operation could be implemented, with the interrupt service routine putting the characters in a FIFO buffer, and the main program picking up new characters from the buffer. This routine is not suitable for interrupt operation, but rather the case where the program is to wait for the next character. The routine will return with the character in the accumulator.

```
CHRIN   IN      13      Read serial input status
        ANI     02H     Mask all but the input byte ready status bit
        JZ      CHRIN   Return and wait for character
        IN      12      Read character
        ANI     7FH     Mask off upper bit if present (for ASCII only)
        RET             Return with character in accumulator
```

The third routine is used for sending characters. It is called once for each character sent, with the character in the accumulator. If multiple characters are being sent, as in a message, another part of the program must pick up the characters one at a time and call this routine.

```
CHROUT  MOV.    B,A     Temporarily store the character in B register
CHOUT1  IN      13      Read serial status
        ANI     01H     mask off all but transmitter ready status bit
        JZ      CHOUT1  Return and wait until ready
        MOV     A,B     Get character back in accumulator
        OUT     12      Send character to serial port
        RET             Return to main program
```

OPERATION HINTS

1. Output of a command to the USART will overwrite any character which is stored in the buffer waiting for transfer to the parallel to serial register. This can be avoided by waiting for TxRDY to be true before sending a command if transmission is taking place (bit 0 of the status byte). It is also possible to disturb the transmission if a command is sent while a SYN character is being generated by the device (in synchronous mode if the software fails to respond to TxRDY). When running in synchronous mode, commands should be transferred only when a positive going edge is detected on the TxRDY status.

2. RxE (Receiver Enable command bit) does not control the receiver, it only masks the RxRDY (Receive Character Ready) status bit. It is possible for the receiver to have one or two characters ready at the time the receiver is enabled. Especially in the synchronous mode, the program should read the USART data twice (it is not necessary to check status) to ensure that the next character ready was received after the receiver was enabled.

3. TxEN (Transmitter Enable) and RTS (RS-232 line) should remain asserted until the last bit of the last character has been shifted out of the USART. A delay of 1 msec after the occurrence of TxE (status byte bit 2) is usually sufficient. Loss of CTS or TxE will immediately clamp the serial output line. Also note, the loss of TxE in the synchronous mode clamps the data at a SPACE instead of normal MARK. This does not occur in asynchronous mode.

4. TxE can momentarily go true while data (including USART generated SYNs) is transferred to the parallel to serial register. If TxE is being used, check it for several consecutive status reads.

5. A BREAK results in a string of characters with framing errors. If reception is to be continued after a BREAK, note that the last character received during a break, including any framing error associated with it, is indeterminate.

7. TIMERS

The MPU-B has three 16-bit timers with programmable mode and count. Program access to the timers is through 4 memory-mapped locations (D100-D103), which are enabled/disabled by control port F3 bit 7. They are part of the memory-mapped functions in block D which are all simultaneously enabled/disabled through the control port. Block D may be left normally disabled, and enabled momentarily to access the timers. If this mode is used, note the 3 CYCLE DELAY in both enable and disable. (Refer to ADDRESS DECODE AND CONTROL section.)

The timers are address as follows:

TIMER READ MODE

ADDRESS	FUNCTION
D100	Read counter No. 0
D101	Read counter No. 1
D102	Read counter No. 2
D103	No function - read FF data

TIMER WRITE MODE

ADDRESS	FUNCTION
D100	Load counter No. 0
D101	Load counter No. 1
D102	Load counter No. 2
D103	Write mode byte

Since the timer access is memory mapped, data can be read or written to the timer with any of the many 8085 memory reference instructions. There are some restrictions, however, in the order of operations. These restrictions are described in this section, and must be followed for correct operation of the timers.

The three timers are completely independant of each other, with the single exception that an option switch will allow timer No. 2 to count the output pulses from timer No. 1. All timers count at a 2 MHz rate, except No. 2 when it is counting No. 1 output pulses. They may be programmed to count either in 16-bit binary, or in 4-digit Binary Coded Decimal (BCD).

Single counts or automatic reload and repeat modes are available. Switch options permit connecting the output of timer No. 1 and/or No. 2 to interrupt the processor to location 003C.

TIMER NO. 0

Timer No. 0 is normally dedicated entirely to providing a programmable baud rate clock for the serial I/O interface, but can be programmed for other functions during any time that the serial interface is not being used. No interrupt is provided for on timer No. 0 output, although it could be jumpered to an interrupt jumper pad.

TIMER CONFIGURATION

TIMER SECTION	COUNT MODE	INTERRUPT LOCATION	COUNT RATE	OPTION SWITCH SETTINGS
Timer 0	BINARY	no int.	2 MHz	none
Timer 0	BCD	no int.	2 MHz	none
Timer 1	BINARY	no int.	2 MHz	T1I OUT
Timer 1	BCD	no int.	2 MHz	T1I OUT
Timer 1	BINARY	003C	2 MHz	T1I IN
Timer 1	BCD	003C	2 MHz	T1I IN
Timer 2	BINARY	no int.	2 MHz	T2I,CTI OUT - C2M IN
Timer 2	BCD	no int.	2 MHz	T2I,CTI OUT - C2M IN
Timer 2	BINARY	003C	2 MHz	CTI OUT - T2I,C2M IN
Timer 2	BCD	003C	2 MHz	CTI OUT - T2I,C2M IN
Timer 2	BINARY	no int.	T-1 OUTPUT	T2I,C2M OUT - CT1 IN
Timer 2	BCD	no int.	T-1 OUTPUT	T2I,C2M OUT - CT1 IN
Timer 2	BINARY	003C	T-1 OUTPUT	C2M OUT - T2I,CT1 IN
Timer 2	BCD	003C	T-1 OUTPUT	C2M OUT - T2I,CTI IN

Except for the lack of interrupt on timer No. 0, and the switch selectable clock input on timer No. 2, The three timers are completely identical in capabilities. Each timer consists of a single 16-bit, pre-settable DOWN counter. The counter can operate in either binary or BCD, and each has a hardware gate and output. The exact function of the gate and output is configured by the mode bytes stored in D103. The timers can be used by the software without any external connections to the gate and output connections. However, they have been provided with solder pads for wires or pin sockets in case it is desired to use the gate or output for any external function. (For example, the outputs could be used to provide a programmable baud rate clock to another I/O board such as the SIO.)

The counters are fully independant and each can have separate Mode configuration and timing operation, binary or BCD, single count or repetitive. Special capabilities in the mode byte handle the loading of partial count values so that software overhead can be minimized for these functions. Commands and logic are included so that the contents of each counter can be read "on the fly" without having to inhibit the clock input.

PROGRAMMING THE TIMERS

The complete mode of operation is programmed by the software. A set of control bytes MUST be sent out by the processor to initialize each timer with the desired Mode and quantity information. These control bytes program the Mode, loading sequence, and selection of binary or BCD counting. Since the timers have no power on reset, they should be programmed even if they are not to be used immediately, especially if the interrupts are connected. A counter not to be used can be programmed to count to a short termination and stop.

All of the modes for each timer are programmed by writing bytes to D103. Each timer is individually set-up by writing a Mode byte into this location. The Mode byte format is:

D7	D6	D5	D4	D3	D2	D1	D0
SC1	SC0	RL1	RL0	M2	M1	M0	BCD

The four control fields are defined as follows:

SC SELECT COUNTER

SC1	SC0	Function
0	0	Select Counter No. 0
0	1	Select Counter No. 1
1	0	Select Counter No. 2
1	1	Not Used

RL READ / LOAD

RL1	RL0	Function
0	0	Counter Latching operation (see read procedure)
0	1	Read/Load Least significant byte ONLY
1	0	Read/Load Most significnt byte ONLY
1	1	Read/Load Least significant byte first, then Most significant byte second.

M MODE

M2	M1	M0	Function
0	0	0	Mode 0
0	0	1	Mode 1
X	1	0	Mode 2
X	1	1	Mode 3
1	0	0	Mode 4
1	0	1	Mode 5

BCD

BCD	Function
0	Binary count (16 bits)
1	Binary coded decimal (BCD) counter - 4 decades

MODE DEFINITIONS

Many of the mode definition affect the detailed nature of the signal on the out pin of interest if used for external purposes. The program can only read count.

MODE 0: OUTPUT HIGH ON TERMINAL COUNT

The output will be initially low after the Mode set operation. After the count is loaded into the selected count register, the output will remain low and the counter will count. When terminal count is reached the output will go high and remain high until the selected count register is reloaded with the MODE.

If the counter register is reloaded during counting:

Loading the first byte stops the current count
Loading the second byte starts the nw count.

The gate input must be high to enable counting; if it is pulled low, counting will be inhibited.

MODE 1: PROGRAMMABLE ONE-SHOT

This mode needs an external connection to the gate to operate. After being set-up, the output will go low on the clock following the RISING EDGE of the gate signal.

The output will go high on the terminal count. If a new count value was loaded during the timeout, it will not affect the duration of this output-low time. The new count value will be used following the next RISING EDGE of the gate input. The current count can be read without affecting the duration of the one-shot pulse.

The "one shot" is retriggerable, i.e. the output will remain low for the full count after the latest rising edge of the gate input.

MODE 2: PROGRAMMABLE RATE PULSE GENERATOR

Divide by N counter. The output will be low for one period of the clock input. The period from one pulse to the next equals the number of clock pulses input to the count register. If the count register is reloaded between output pulses, the present period will not be affected, but the next period will reflect the new value.

The gate input, if pulled low, will inhibit counting and force the output high. When the gate input returns high, the counter will start from the initial count. Thus the gate input can be used to synchronize the counter.

When setting this mode, the output will also remain high until after the count register is loaded. Thus the output can also be synchronized through software.

MODE 3: SQUARE WAVE GENERATOR

This is the mode that would normally be used in timer 0 to generate a baud rate clock for the serial interface.

This mode is very similar to MODE 2, except that the output remains high for only one half the count, and goes low for the entire other half. If the count loaded in the register was odd, the output will be high for (N+1)/2 counts, and low for (N-1)/2 counts.

If the counter register is reloaded with a new value during counting, the new value will be reflected after the low to high output transition of the current count.

MODE 4: SOFTWARE TRIGGERED STROBE

After the mode is set, the output will be high. When the count is loaded, the counter will begin counting. On terminal count, the output will go low for one input clock period, then return high.

If the count register is reloaded between output pulses the present period will not be affected, but the next period will reflect the new value. If the gate input is pulled low, counting will be inhibited. If the counter register is reloaded while the count is inhibited by the gate input, the counter will begin counting the new number when the gate returns high.

MODE 5: HARDWARE TRIGGERED STROBE

The use of this mode requires an external connection to the gate input. The counter will start counting after the RISING EDGE of the gate input and go low for one clock period when the terminal count is reached. The counter is retriggerable, the output will not go low until the full count after the most recent rising edge of the gate input.

SUGGESTED MODES

Several of the modes require external connections to be made to the gate input, some are primarily useful for external connections to the output. For timer operation from software only (no hardware interaction external to the MPU-B board), the following modes are suggested:

Baud rate generation for serial port:

Timer 0, Mode 3

Note: Using the square-wave mode keeps pulse length times within limits for typical UARTS & USARTS (including the 8251 used on the MPU-B). To obtain the typical X16 baud rate clock, set the count value to:

COUNT = 2,000,000 / (BAUD RATE * 16)

This formula is in base 10. COUNT must be converted to hexadecimal to be loaded into the timer, unless the time is run in BCD mode! Round the calculated value to the next LOWER integer. This will result in the transmitted signal being slightly faster than the received signal, preventing occasional lost characters when echoeing all received characters at the maximum rate (as in reading paper tapes). The small percentage difference from nominal speed is insignificant for any standard rates. For operating the 8251 with synchronous data links, the clock should be X1, and the formula is:

COUNT = 2,000,000 / BAUD RATE

Again, this formula is in base 10. The resulting COUNT should be rounded to the nearest integer.

Some count values for common asynchronous baud rates (X16 clock):

BAUD RATE	DECIMAL COUNT	HEXADECIMAL COUNT	ERROR
9600	13	000D	-.0016
4800	26	001A	-.0016
2400	52	0034	-.0016
1200	104	0068	-.0016
600	208	00D0	-.0016
300	416	01A0	-.0016
150	833	0341	-.0004
134.5	929	03A1	-.0004
110	1136	0470	-.00032
75	1666	0682	-.0004
45	2777	0AD9	-.00028

Some count values for common synchronous baud rates (X1 clock):

BAUD RATE	DECIMAL COUNT	HEXADECIMAL COUNT	ERROR
56000	36	0024	+.0079
38400	52	0034	-.0016
19200	104	0068	-.0016
9600	208	00D0	-.0016
4800	416	01A0	-.0016
2400	833	0341	-.0004
1200	1666	0682	-.0004

Timer for tracking real time (as in a time of day clock, timed delay, timed intervals, etc.)

Timer 1, Mode 2 and Timer 2, Mode 2
(Option switch CT1 IN and C2M OUT, so that timer 2 counts output pulses of timer 1.)

A) Timer 1 and 2 both in binary mode, count values both set at maximum (0000). This forms a 32 bit counter with a count rate of 2 MHz. Maximum time is 2**32 periods of .5 microsecond (2147.48 seconds, 35.79 minutes). Timer 1 is the two low-order bytes, and timer 2 is the two high-order bytes. Only timer 2 need be reinitialized each time if an accuracy of +0/-.0328 seconds is sufficient.

B) Timer 1 in binary mode, count value set at 4E20 hex; Timer 2 in BCD mode, count value set at maximum (0000). This forms a 4 digit BCD counter counting .01 second intervals (timer 2), for a maximum count of 100 seconds. Only timer 2 need be reinitialized each time if an accuracy of +0/-.01 second is sufficient.

NOTE: In both A and B above, the counters are counting DOWN from the initial value. To calculate the actual time elapsed from initialization the binary value read from the two timers (A) or the BCD value read from timer 2 (B) must be subtracted from the initial value (0000). If the count read from timer 2 is equal to the minimum count (also 0000), then the maximum timer capacity has been exceeded, assuming care has been taken that it is not tested before one time increment (.0328 sec (A), or .01 sec (B)) has occurred.

LOADING THE MODE AND COUNT VALUES

The order of programming is quite flexible. Since each timer's Mode control byte has a unique address, loading the Mode control bytes is independant of sequence. Not only can the Mode control bytes be written in any sequence, but they can be separated by other operations, such as loading the count registers.

The loading of the count registers with the actual count value, however, must be done in the sequence programmed in the Mode control byte. There is no separate addressing for low or high count values, so the timers will assume the count value bytes output are in sequence according to the mode programmed. Since the three timers have unique addresses for loading count values, the order of timers programmed does not matter.

The one or two count value bytes do not have to immediately follow the associated Mode control byte. They can be loaded at any time after the Mode control byte has been written as long as the correct number of bytes are loaded in order. The two following examples are valid sequences:

D103	36	Mode byte - counter 0, two bytes, mode 3
D100	XX	Low order count value
D100	YY	High order count value
D103	A4	Mode byte - counter 2, High order only, mode 2
D103	36	Mode byte - counter 0, two bytes, mode 3
D103	79	Mode byte - counter 1, two bytes, mode 4, BCD
D100	XX	Low order count value, counter 0
D100	YY	High order count value, counter 0
D102	SS	High order count value, counter 2
D101	QQ	Low order count value, counter 1
D101	RR	High order count value, counter 1

READ OPERATIONS

In many timer applications it becomes necessary to read the value of a count in progress to make a computational decision. The counters contain logic that will allow the programmer to easily read the contents without disturbing the count in progress.

If the gate is being used to control the timer, and the programmer is assured that the counter will be disabled during the read, no special action is necessary. Reads from the appropriate address will be responded to with the current count value.

The first read contains the least significant byte.

The second read contains the most significant byte.

It is necessary to complete the entire reading procedure. If two bytes are programmed to be read then two bytes MUST be read before any loading command can be sent to the same counter.

READING WHILE COUNTING

In order for the programmer to read the contents of a counter without effecting or disturbing the counting operation while in progress, the timers have logic that can be accessed through a write to the mode register. When it is wished to read the contents of a selected counter "on the fly", a write is made to the Mode register with a code which latches the present count value into a storage register so that a stable count value may be read. The programmer then issues a normal read command to the selected counter to read the contents of the latched register. Note from the previous table that the latch command is affected by a Mode byte with bits 4 & 5 (RL1,RL0) equal to 00. The counter to be latched is selected as before. Note that it is still necessary to complete the entire read sequence as programmed.

INTERRUPTS FROM TIMERS 1 & 2

Option switches are provided to connect the outputs of timers 1 & 2 to interrupt 7.5 (to location 003C). Either timer or both may be connected. When both are connected, the program must check the contents of the two timers to determine which one issued the interrupt (i.e. which one has reached terminal count).

The outputs of these two timers have been inverted, so that an interrupt will occur when the output is LOW. This permits use to be made of modes 2 through 5, but not of mode 0 or 1. (Only for interrupts, all modes operate for program reads of count status.) Mode 0 will request an interrupt at the beginning of the count. Mode 1 will request an interrupt once the count is started by the signal on the gate terminal. The other modes will request an interrupt either at terminal count or 1/2 terminal count, depending on the mode. Repetitive modes will produce periodic interrupts. Since RST7.5 is edge sensitive, an interrupt will not be missed even though the termination count output is a short pulse. Also, there is no need to reset the counter in any way before interrupts can be re-enabled. NOTE: Since the outputs of timers 1 and 2 are simply ORed together to create the interrupt signal, an extended low output signal on one (such as during the second half of the count in mode 3) will mask any output change on the

other. If both timers are used in pulse output modes, no interrupts will be lost. It would still be possible for both to request an interrupt on the same cycle, which case must be resolved by the software.

Examples

baud rate clock for 110, 9600
auto baud rate select

8. USING INTERRUPTS

Provisions have been made so that the four levels of interrupt provided completely internally by the 8085 can be used for the MPU-B I/O circuits, or jumpered to any of the V10 through V17 lines on the backplane for use by other peripheral boards. Switch options are used for connecting the on-board I/O to the interrupt lines.

INTERRUPTS FROM SERIAL AND PARALLEL I/O

Only three of the interrupt lines are used for the on-board I/O. The serial and parallel input status bits are OR'ed together and provided with a switch to the RST 5.5 line. If either the PRI switch (Parallel Rec. Interrupt) or the SRI switch (Serial Rec. Interrupt) is closed, a jump to an interrupt service routine must be located at 002CH. An interrupt will occur whenever the interrupt mask enables that interrupt and a receive character is ready.

If both switches are closed, it is up to the interrupt service routine to determine from the I/O status bytes whether the character which is ready is from the serial or parallel port (or possibly a character will be ready at both ports). Whenever the status bit (Receive Character Ready) is true, an intertupt will occur as soon as RST 5.5 is enabled. The request for interrupt will be reset by a data read from the active port.

The serial and parallel output ready status bits are similarly OR'ed together and provided with a switch each to the RST 6.5 interrupt line. The jump to the interrupt service routine must be located at 0034H. An interrupt will occur whenever the interrupt mask enables that interrupt and the transmit portion of the I/O interface is ready to accept the next character.

Again the interrupt service routine must resolve the ambiguity if both switches are closed.

The parallel I/O port cannot be used in the interrupt mode unless it is connected to a device which uses the handshaking mode of operation with the port, as it is the handshake strobes that set the respective ready bits.

INTERRUPTS FROM TIMERS 1 AND 2

Option switches are provided to connect the outputs of timers 1 and 2 to interrupt 7.5 (to location 003C). Either timer or both may be connected. When both are connected, the program must check the contents of the two timers to determine which one issued the interrupt (i.e., which one has reached terminal count).

The outputs of these two timers have been inverted, so that an interrupt will occur when the output is LOW. This permits use to be made of modes 2 through 5, but not of mode 0 r 1 (only for interrupts, all modes operate for program reads of count status). Mode 0 will request an interrupt at the beginning of the count. Mode 1 will request an interrupt once the count is started by the signal on the gate terminal. The other modes will request an interrupt either at terminal count or 1/2 terminal count, depending on the mode. Repetitive modes will produce periodic interrupts. Since RST 7.5 is edge sensitive, an interrupt will not be missed even though the termination count output is a short pulse. Also, there is no need to reset the counter in any way before interrupts can be re-enabled. NOTE: Since the outputs of timers 1 and 2 are simply OR'ed together to create the interrupt signal, an extended low output signal on one (such as during the second half of the count is Mode 3) will mask any output change on the other. If both timers are used in pulse output modes, no interrupts will be lost. It would still be possible for both to request an interrupt on the same cycle, which case must be resolved by the software.

OTHER INTERRUPTS

The TRAP interrupt (RST 4.5) is not connected to anything on the MPU-B board. It is provided with a traced-in jumper to V16 to be available to peripheral boards. It is a non-maskable interrupt and is the highest priority.

The interrupt request line is present in the 8085 as it is in the 8080. It is still available for use by an interrupt board such as the PIC-8 and in normal use (using RST-0 through RST-7) provides an additional 8 interrupt levels. It is also provided with a traced-in jumper to the V17 line.

APPENDIX 1

8085 INSTRUCTION SET

The following two pages are reproduced with the permission of Intel Corporation, Santa Clara California.

8085 INSTRUCTION SET SUMMARY

Mnemonic	Description	Instruction Code[1] D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0	Clock[2] Cycles
MOVE, LOAD, AND STORE										
MOV r1,r2	Move register to register	0	1	D	D	D	S	S	S	4
MOV M,r	Move register to memory	0	1	1	1	0	S	S	S	7
MOV r,M	Move memory to register	0	1	D	D	D	1	1	0	7
MVI r	Move immediate register	0	0	D	D	D	1	1	0	7
MVI M	Move immediate memory	0	0	1	1	0	1	1	0	10
LXI B	Load immediate register Pair B & C	0	0	0	0	0	0	0	1	10
LXI D	Load immediate register Pair D & E	0	0	0	1	0	0	0	1	10
LXI H	Load immediate register Pair H & L	0	0	1	0	0	0	0	1	10
LXI SP	Load immediate stack pointer	0	0	1	1	0	0	0	1	10
STAX B	Store A indirect	0	0	0	0	0	0	1	0	7
STAX D	Store A indirect	0	0	0	1	0	0	1	0	7
LDAX B	Load A indirect	0	0	0	0	1	0	1	0	7
LDAX D	Load A indirect	0	0	0	1	1	0	1	0	7
STA	Store A direct	0	0	1	1	0	0	1	0	13
LDA	Load A direct	0	0	1	1	1	0	1	0	13
SHLD	Store H & L direct	0	0	1	0	0	0	1	0	16
LHLD	Load H & L direct	0	0	1	0	1	0	1	0	16
XCHG	Exchange D & E, H & L Registers	1	1	1	0	1	0	1	1	4
STACK OPS										
PUSH B	Push register Pair B & C on stack	1	1	0	0	0	1	0	1	12
PUSH D	Push register Pair D & E on stack	1	1	0	1	0	1	0	1	12
PUSH H	Push register Pair H & L on stack	1	1	1	0	0	1	0	1	12
PUSH PSW	Push A and Flags on stack	1	1	1	1	0	1	0	1	12
POP B	Pop register Pair B & C off stack	1	1	0	0	0	0	0	1	10
POP D	Pop register Pair D & E off stack	1	1	0	1	0	0	0	1	10
POP H	Pop register Pair H & L off stack	1	1	1	0	0	0	0	1	10
POP PSW	Pop A and Flags off stack	1	1	1	1	0	0	0	1	10
XTHL	Exchange top of stack, H & L	1	1	1	0	0	0	1	1	16
SPHL	H & L to stack pointer	1	1	1	1	1	0	0	1	6
JUMP										
JMP	Jump unconditional	1	1	0	0	0	0	1	1	10
JC	Jump on carry	1	1	0	1	1	0	1	0	7/10
JNC	Jump on no carry	1	1	0	1	0	0	1	0	7/10
JZ	Jump on zero	1	1	0	0	1	0	1	0	7/10
JNZ	Jump on no zero	1	1	0	0	0	0	1	0	7/10
JP	Jump on positive	1	1	1	1	0	0	1	0	7/10
JM	Jump on minus	1	1	1	1	1	0	1	0	7/10
JPE	Jump on parity even	1	1	1	0	1	0	1	0	7/10
JPO	Jump on parity odd	1	1	1	0	0	0	1	0	7/10
PCHL	H & L to program counter	1	1	1	0	1	0	0	1	6
CALL										
CALL	Call unconditional	1	1	0	0	1	1	0	1	18
CC	Call on carry	1	1	0	1	1	1	0	0	9/18
CNC	Call on no carry	1	1	0	1	0	1	0	0	9/18
CZ	Call on zero	1	1	0	0	1	1	0	0	9/18
CNZ	Call on no zero	1	1	0	0	0	1	0	0	9/18
CP	Call on positive	1	1	1	1	0	1	0	0	9/18
CM	Call on minus	1	1	1	1	1	1	0	0	9/18
CPE	Call on parity even	1	1	1	0	1	1	0	0	9/18
CPO	Call on parity odd	1	1	1	0	0	1	0	0	9/18
RETURN										
RET	Return	1	1	0	0	1	0	0	1	10
RC	Return on carry	1	1	0	1	1	0	0	0	6/12
RNC	Return on no carry	1	1	0	1	0	0	0	0	6/12
RZ	Return on zero	1	1	0	0	1	0	0	0	6/12
RNZ	Return on no zero	1	1	0	0	0	0	0	0	6/12
RP	Return on positive	1	1	1	1	0	0	0	0	6/12
RM	Return on minus	1	1	1	1	1	0	0	0	6/12
RPE	Return on parity even	1	1	1	0	1	0	0	0	6/12
RPO	Return on parity odd	1	1	1	0	0	0	0	0	6/12
RESTART										
RST	Restart	1	1	A	A	A	1	1	1	12
INPUT/OUTPUT										
IN	Input	1	1	0	1	1	0	1	1	10
OUT	Output	1	1	0	1	0	0	1	1	10
INCREMENT AND DECREMENT										
INR r	Increment register	0	0	D	D	D	1	0	0	4
DCR r	Decrement register	0	0	D	D	D	1	0	1	4
INR M	Increment memory	0	0	1	1	0	1	0	0	10
DCR M	Decrement memory	0	0	1	1	0	1	0	1	10
INX B	Increment B & C registers	0	0	0	0	0	0	1	1	6
INX D	Increment D & E registers	0	0	0	1	0	0	1	1	6
INX H	Increment H & L registers	0	0	1	0	0	0	1	1	6
INX SP	Increment stack pointer	0	0	1	1	0	0	1	1	6
DCX B	Decrement B & C	0	0	0	0	1	0	1	1	6
DCX D	Decrement D & E	0	0	0	1	1	0	1	1	6
DCX H	Decrement H & L	0	0	1	0	1	0	1	1	6
DCX SP	Decrement stack pointer	0	0	1	1	1	0	1	1	6
ADD										
ADD r	Add register to A	1	0	0	0	0	S	S	S	4
ADC r	Add register to A with carry	1	0	0	0	1	S	S	S	4
ADD M	Add memory to A	1	0	0	0	0	1	1	0	7
ADC M	Add memory to A with carry	1	0	0	0	1	1	1	0	7
ADI	Add immediate to A	1	1	0	0	0	1	1	0	7
ACI	Add immediate to A with carry	1	1	0	0	1	1	1	0	7
DAD B	Add B & C to H & L	0	0	0	0	1	0	0	1	10
DAD D	Add D & E to H & L	0	0	0	1	1	0	0	1	10
DAD H	Add H & L to H & L	0	0	1	0	1	0	0	1	10
DAD SP	Add stack pointer to H & L	0	0	1	1	1	0	0	1	10
SUBTRACT										
SUB r	Subtract register from A	1	0	0	1	0	S	S	S	4
SBB r	Subtract register from A with borrow	1	0	0	1	1	S	S	S	4
SUB M	Subtract memory from A	1	0	0	1	0	1	1	0	7
SBB M	Subtract memory from A with borrow	1	0	0	1	1	1	1	0	7
SUI	Subtract immediate from A	1	1	0	1	0	1	1	0	7
SBI	Subtract immediate from A with borrow	1	1	0	1	1	1	1	0	7
LOGICAL										
ANA r	And register with A	1	0	1	0	0	S	S	S	4

8085 INSTRUCTION SET SUMMARY (Cont.)

Mnemonic	Description	Instruction Code[1] D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0	Clock[2] Cycles
XRA r	Exclusive Or register with A	1	0	1	0	1	S	S	S	4
ORA r	Or register with A	1	0	1	1	0	S	S	S	4
CMP r	Compare register with A	1	0	1	1	1	S	S	S	4
ANA M	And memory with A	1	0	1	0	0	1	1	0	7
XRA M	Exclusive Or memory with A	1	0	1	0	1	1	1	0	7
ORA M	Or memory with A	1	0	1	1	0	1	1	0	7
CMP M	Compare memory with A	1	0	1	1	1	1	1	0	7
ANI	And immediate with A	1	1	1	0	0	1	1	0	7
XRI	Exclusive Or immediate with A	1	1	1	0	1	1	1	0	7
ORI	Or immediate with A	1	1	1	1	0	1	1	0	7
CPI	Compare immediate with A	1	1	1	1	1	1	1	0	7
ROTATE										
RLC	Rotate A left	0	0	0	0	0	1	1	1	4
RRC	Rotate A right	0	0	0	0	1	1	1	1	4
RAL	Rotate A left through carry	0	0	0	1	0	1	1	1	4
RAR	Rotate A right through carry	0	0	0	1	1	1	1	1	4
SPECIALS										
CMA	Complement A	0	0	1	0	1	1	1	1	4
STC	Set carry	0	0	1	1	0	1	1	1	4
CMC	Complement carry	0	0	1	1	1	1	1	1	4
DAA	Decimal adjust A	0	0	1	0	0	1	1	1	4
CONTROL										
EI	Enable interrupts	1	1	1	1	1	0	1	1	4
DI	Disable interrupt	1	1	1	1	0	0	1	1	4
NOP	No-operation	0	0	0	0	0	0	0	0	4
HLT	Halt	0	1	1	1	0	1	1	0	5
NEW 8085 INSTRUCTIONS										
RIM	Read Interrupt Mask	0	0	1	0	0	0	0	0	4
SIM	Set Interrupt Mask	0	0	1	1	0	0	0	0	4

NOTES: 1. DDD or SSS: B=000, C=001, D=010, E=011, H=100, L=101, Memory=110, A=111.
2. Two possible cycle times, (6/12) indicate instruction cycles dependent on condition flags.

SUMMARY OF 8085 INSTRUCTIONS

INSTR	STATES	CYCLES	NOTES
AR r	4	1	
AR I	7	2	
AR M	7	2	
CALL	18	5	2
CCN	9/18	2/5	2,4
CMC	4	1	
CMA	4	1	
DAA	4	1	
DAD	10	3	
DCR r	4	1	3
DCR M	10	3	
DCX	6	1	2
DI	4	1	
EI	4	1	
HLT	5	1	5
IN	10	3	
INR r	4	1	3
INR M	10	3	
INX	6	1	2
JMP	10	3	
JCN	7/10	2/3	4
LDS	13	4	
LDAX	7	2	
LHLD	16	5	
LXU	10	3	
MVI M	10	3	
MVI r	7	2	
MOV M,r	7	2	
MOV r,M	7	2	
MOV r,r	4	1	3
NOP	4	1	

INSTR	STATES	CYCLES	NOTES
OUT	10	3	
PCHL	6	1	2
POP	10	3	
PUSH	12	3	2
ROT	4	1	
RET	10	3	
RCN	6/12	1/3	2
RIM*	4	1	
RST	12	3	2
SHLD	16	5	
SIM*	4	1	
STA	13	4	
STAX	7	2	
STC	4	1	
XCHG	4	1	
XTHL	16	5	
SPHL	6	1	2

*New Instruction

NOTES:
1. Two possible state-cycle times indicates dependence on condition flag.
2. Increase in states over 8080 due to necessity of a T_6 during M_1.
3. Decreased from 5 to 4 states during M_1.
4. Instruction branches over last address fetch if condition false.
5. 5 cycles to get a HLT state, 1 cycle necessary to get out a HLT.

APPENDIX 2

CHANGING OPTION JUMPERS

Some of the options provided on the MPU-B will be used only in special situations, such as when a system is being customized for a special configuration or special application. These options have been provided as jumper-pads on the printed-circuit board, and they have a small trace on the board connecting these pads in the standard factory configuration.

When it is desired to change any of these options, the original connecting trace must be broken. This is easily accomplished using a SHARP modelling knife to cut the trace in TWO locations 1/16 to 1/8 inch apart, and pealing the short section out using the point of the knife. The cuts are made more easily if they are slanted so that the bases (at the board surface) are closer than the tops (at the surface of the trace). About 45 degrees is optimum. Be sure to use a SHARP blade.

Once the original factory option traces have been opened, the desired connection can be made by soldering a short wire jumper between the pads provided for the alternate option. Should it later be desired to return the option to its original state, unsolder the jumper wire and re-solder it in the pads between which the previously cut trace runs. As in all soldering work on printed circuit boards, care should be taken to use only the heat necessary to avoid damaging the pads on the board. When removing a jumper, cut it first so that you can remove one end at a time without interference from a second still-connected end. Properly unsoldering anything from a printed circuit board can be difficult. If you do not have the proper tools or skill, protect your board and warrantee by seeking professional aid.

APPENDIX 3

PROM INSTALLATION

The following diagrams show the jumper configuration for two types of PROMs, the 2708 and the 2716. MAKE SURE THE BOARD IS PROPERLY JUMPERED BEFORE POWER IS APPLIED TO AVOID DAMAGE!!!! For ROMs or PROMs other than these, jumper the board for the chip to which the desired PROM is equivalent. If you are not sure of the equivalency, get the assistance of qualified technical help or call IMSAI Customer Service.

FIGURE II-3

JUMPERS FOR 2708

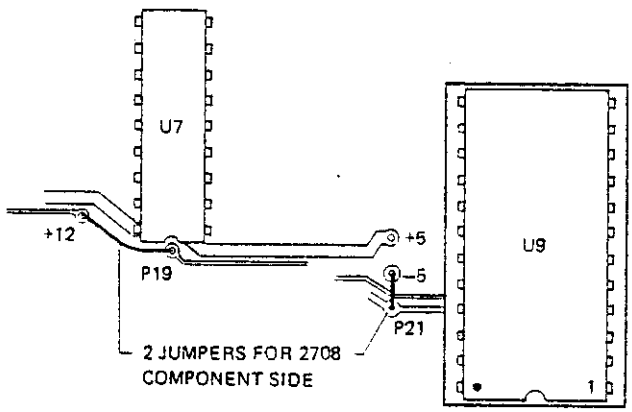
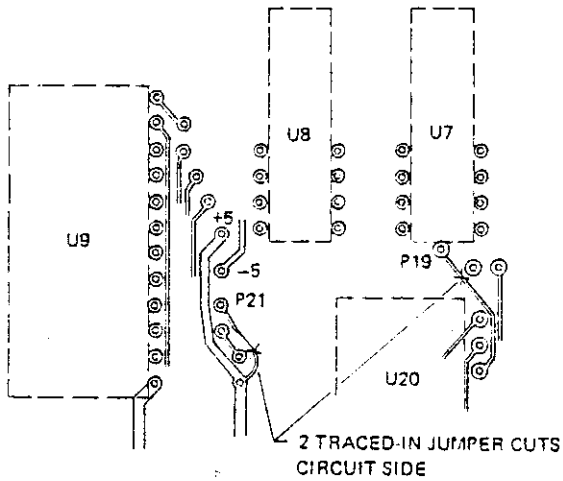


FIGURE II-4

JUMPERS FOR 2708



The bare board is supplied with traced-in jumpers to connect the U9 socket to accept 2716 PROMS.

APPENDIX 4

CUSTOM ADDRESS DECODING

For large quantity OEM uses of the MPU-B, it is possible that an alternate on-board I/O and memory addressing scheme would be desired. IMSAI has the capability of locating the on-board memory-mapped circuits anywhere within 65K in 256 byte blocks. The on-board I/O can be assigned to any ports. Should any change in the addressing scheme described in this Reference Guide be desired, contact IMSAI Customer Service

III. THE VIO BOARD (VIO-C)

III. THE VIO BOARD (VIO-C)

A. THEORY OF OPERATION

1. HARDWARE OVERVIEW

The VIO interfaces to the computer as a 4K byte block of memory. This 4K block includes a 2K byte static video refresh memory, the 2K bytes of VIOROM firmware, and a memory-mapped command port. Figure III-1 indicates the standard VIO memory space configuration. A Block Diagram of the VIO hardware is shown in Figure III-2.

VIO Processes

The operation of the VIO can be broken down into four basic processes: Character storage, display control, refresh display and firmware access.

Character Storage: Characters to be displayed are stored into the video refresh memory by the system CPU. The select and control signals necessary to effect this transfer are generated by the VIO's Bus Interface.

Display Control: The system CPU will assert control over the visual attributes of the screen display by outputting a command byte to the memory-mapped control port of the VIO. This command byte will be latched by the command logic and will specify to the Video Refresh Logic the display parameters (line length, page length, and inverse video).

Refresh Display: The Video Refresh Logic will take characters from the video refresh memory and create the necessary composite video signal to allow the characters to be displayed on an external video monitor device.

Firmware Access: Since the VIO resides in the system memory space, the CPU will execute the VIOROM firmware as if it were a program stored in the system memory. The VIO's Bus Interface allows the CPU to access the VIOROM by providing the necessary select and control signals.

2. VIDEO REFRESH MEMORY

The 2K bytes of video refresh memory is implemented with sixteen 2102 memory chips (U11-U18 and U28-U35). Characters to be displayed are stored in this memory in order beginning with the character in the top left screen location and ending with the character which appears in the lower right corner of the display

The video refresh memory is accessible both by the system CPU as well as by the video refresh logic. Therefore, video refresh memory addressing is achieved through the use of a separate MA bus which may be driven by the S-100 address line during a CPU access, or by the video

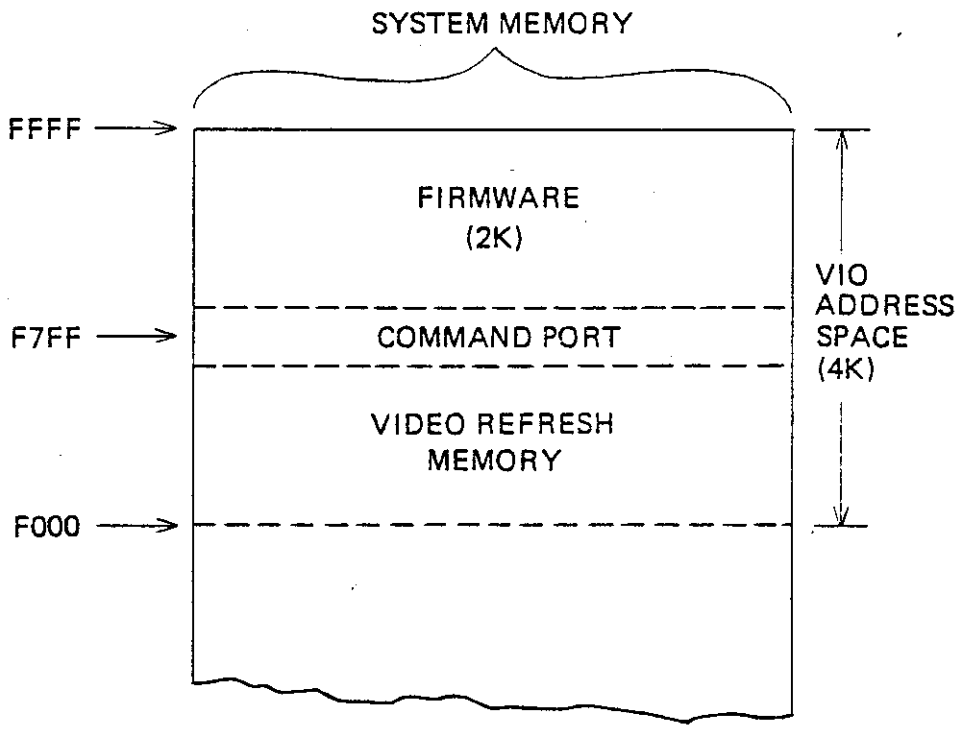


FIGURE III-1 VIO MEMORY SPACE (STANDARD CONFIGURATION)

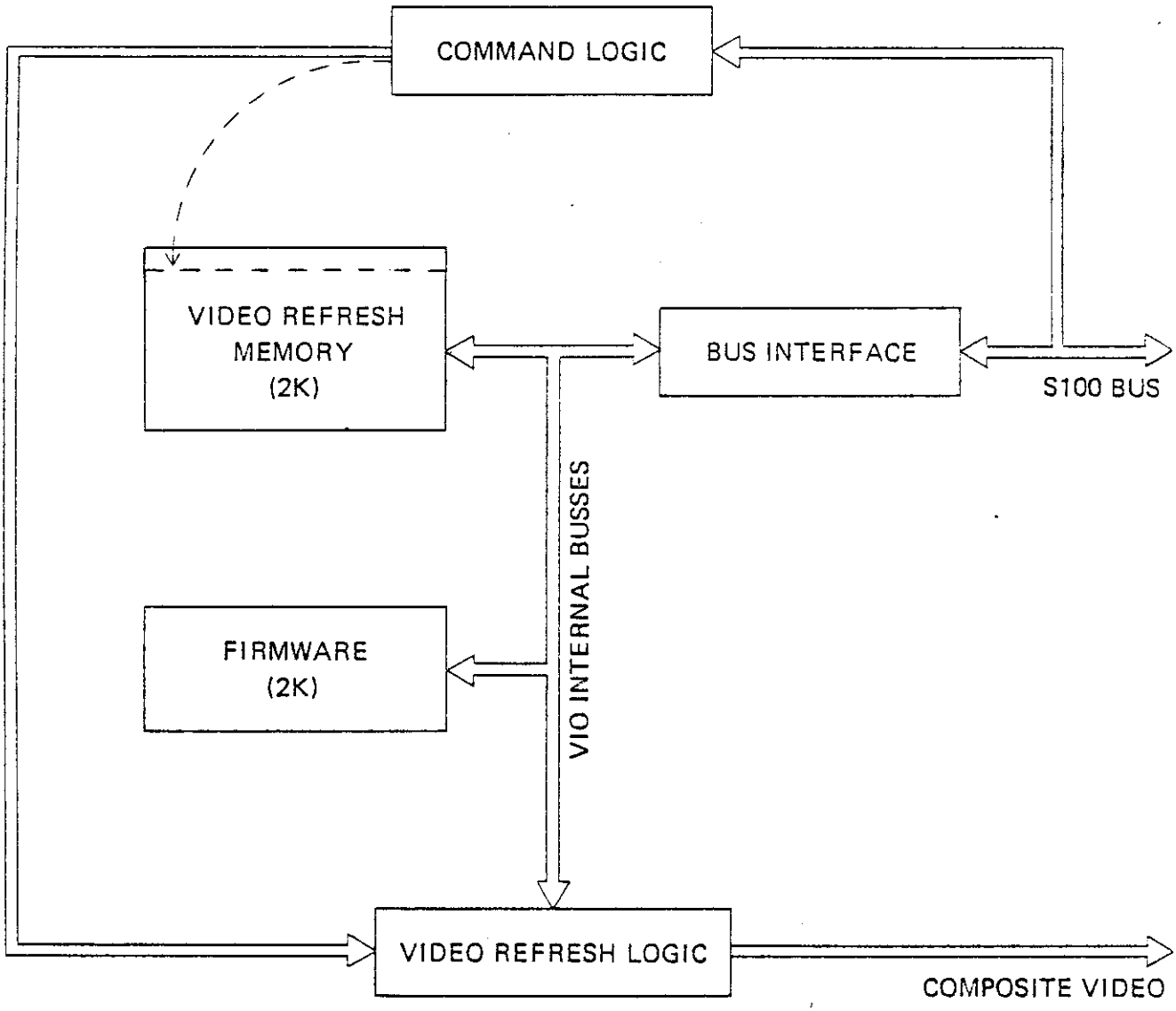


FIGURE III-2 VIO BLOCK DIAGRAM

refresh logic during refresh processes. The necessary multiplexing is achieved with tri-state drivers.

During a CPU write to the refresh memory, data from the system CPU is bussed directly to the data input pins of the memory chips via the S-100 DO bus.

Data out from the video refresh memory appears on the internal PCH bus. During a CPU read of the video refresh memory, the PCH bus drives the internal DB bus which in turn drives the S-100 Data In (DI) bus. During a video refresh, data on the PCH bus is latched by the video refresh logic.

3. VIOROM

The VIOROM firmware resides in the ROM at U46. Since the VIOROM is only accessed by the system CPU, its address lines are driven directly by the S-100 address bus.

During a CPU read of the VIOROM, the data from the PROM chip drives the VIO's DB bus which in turn drives the S-100 DI bus through tri-state drivers.

4. BUS INTERFACE

The VIO's S-100 bus interface provides the control and select signals necessary to effect a CPU transfer or access.

When the VIO is used in a megabyte address space (with MM), the occurrence of A16-A19 (high) and /SINP, /SOUT and /SINTA indicates a CPU access in the top page of memory. The resulting signal enables a 74LS85 comparator at U50.

In a 65K address space, the occurrence of /SINP, /SOUT, and /SINTA is used to directly enable the 74LS85 comparator at U50. The comparator then compares A12-A16 to the settings of the address select switches (SW1). The comparator will drive its "=" output high only when a CPU access to the selected 4K block occurs.

This "=" output is further combined with A11 to determine if the CPU access is to the video refresh memory or to the VIOROM. The indication that the VIOROM is being accessed is used to directly drive the signal ROMSEL to enable the VIOROM at U46. The indication that the access is to the video refresh memory is latched at U40 by the user selected edge of the processor status strobe to prevent transients from interfering with the display. The /Q output of the latch U40 is the signal RAMSEL and is used to enable the refresh memory chips.

In addition to the ROMSEL and RAMSEL select signals, the VIO's bus interface must also provide the necessary control signals to effect a memory transfer.

The WRITE control signal is generated by gating /RAMSEL with /PWR at U26. The resulting signal /RAMWR is active low only during a CPU write to the video refresh memory. /RAMWR directly drives the R/W inputs of the memory chips.

If the wait state option has been selected, wait states are generated by reflecting PWAIT on

the PRDY line when the board is selected. The "not ready" signal is extended using a 74LS74 flip flop (U40) clocked by 02.

5. COMMAND LOGIC

The command port is memory mapped to the top location of the 2K byte video refresh memory space. A0-A10 is ANDed with RAMWR to detect a CPU write to the command port address. This signal then clocks DO0-DO4 into a 74LS74 latch (U44). The outputs of the four bit latch form the control signals /W80, /L24, MD0, MD1 and VIDREV; and are used by the video refresh logic.

/W80 controls line length; /L24 controls page length; MD0 and MD1 are encoded and select screen blanking, the character set and character-by-character reverse video; and VIDREV controls full screen reverse video.

6. VIDEO REFRESH LOGIC

A block diagram of the Video Refresh Logic is shown in Figure III-3. The timing and refresh address logic together generate the sequence of addresses which allow characters to be taken from the Video Refresh Memory. Data from the Refresh Memory (the encoded characters to be displayed) appear at the inputs of the character generator, where they are mapped into the specific pattern of "dots" which will form the visual representation of the characters. The video generation logic then combines the outputs of the character generator with the sync and blanking signals from the Timing Logic to produce the composite video signal required by the video monitor.

a) Timing Logic

The timing circuitry essentially consists of a 12.2304 MHz oscillator and a modulo-203840 counter chain. The counter chain consists of modulo-7, modulo-112 and modulo-260 sections cascaded to provide the 1.7472 MHz character clock, 15.600 KHz horizontal sweep synchronization and blanking and 60.00 Hz vertical sweep synchronization and blanking. This circuitry also provides outputs which specify the current scan position. A block diagram of this circuitry is shown in Figure III-4 and it is described in detail in the following paragraphs.

1) DOT CLOCK: The dot clock (Figure III-5) is a crystal-controlled oscillator which generates the basic 12.2304 Mhz timing reference DCK1 for video generation. It consists of two sections of a 74LS04 cascaded to form a non-inverting amplifier. A series resonant crystal establishes the feed- back path to cause oscillation. A third section of the 74LS04 buffers the oscillator output. A 74LS74 D-flip-flop divides the 12.2304 Mhz signal by two to produce the 6.1152 Mhz clock signal DCK2 required for generation of double width (40 per-line) characters. The reset input of this flip-flop is connected to the composite sync signal /PCOMPSYNC to ensure that the flip-flop starts each line in the zero state (DCK2=1).

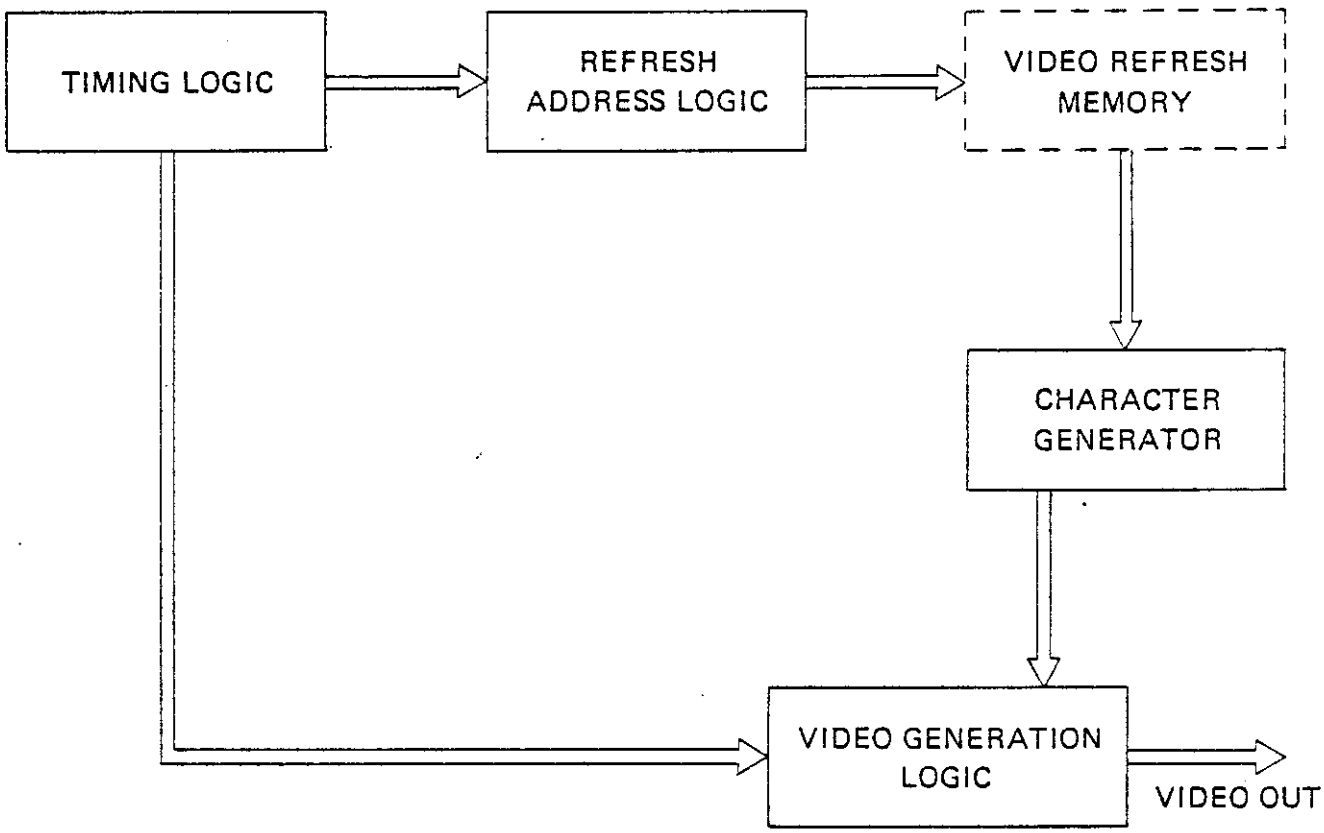


FIGURE III-3 VIDEO REFRESH LOGIC

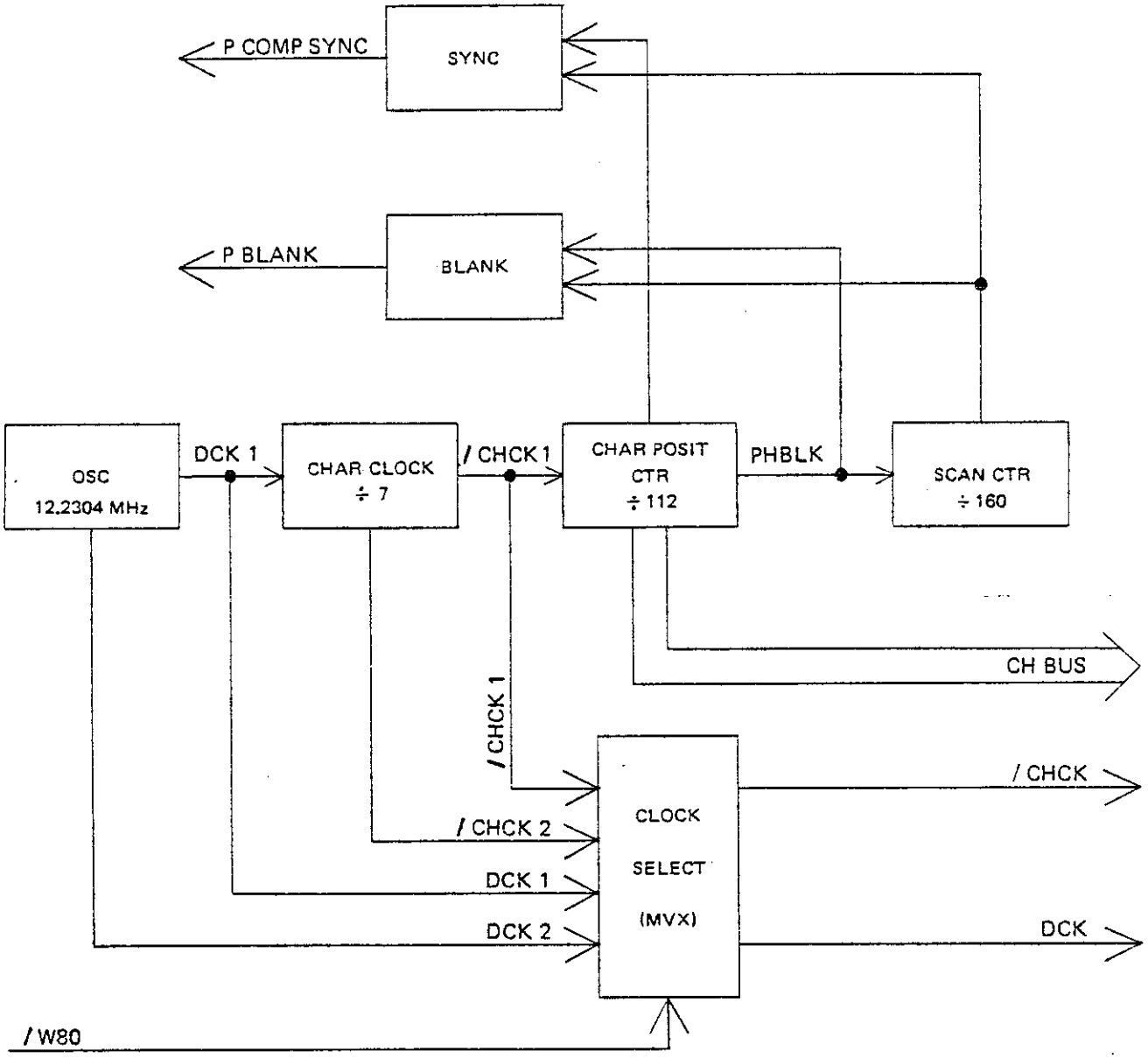


FIGURE III-4 TIMING BLOCK DIAGRAM

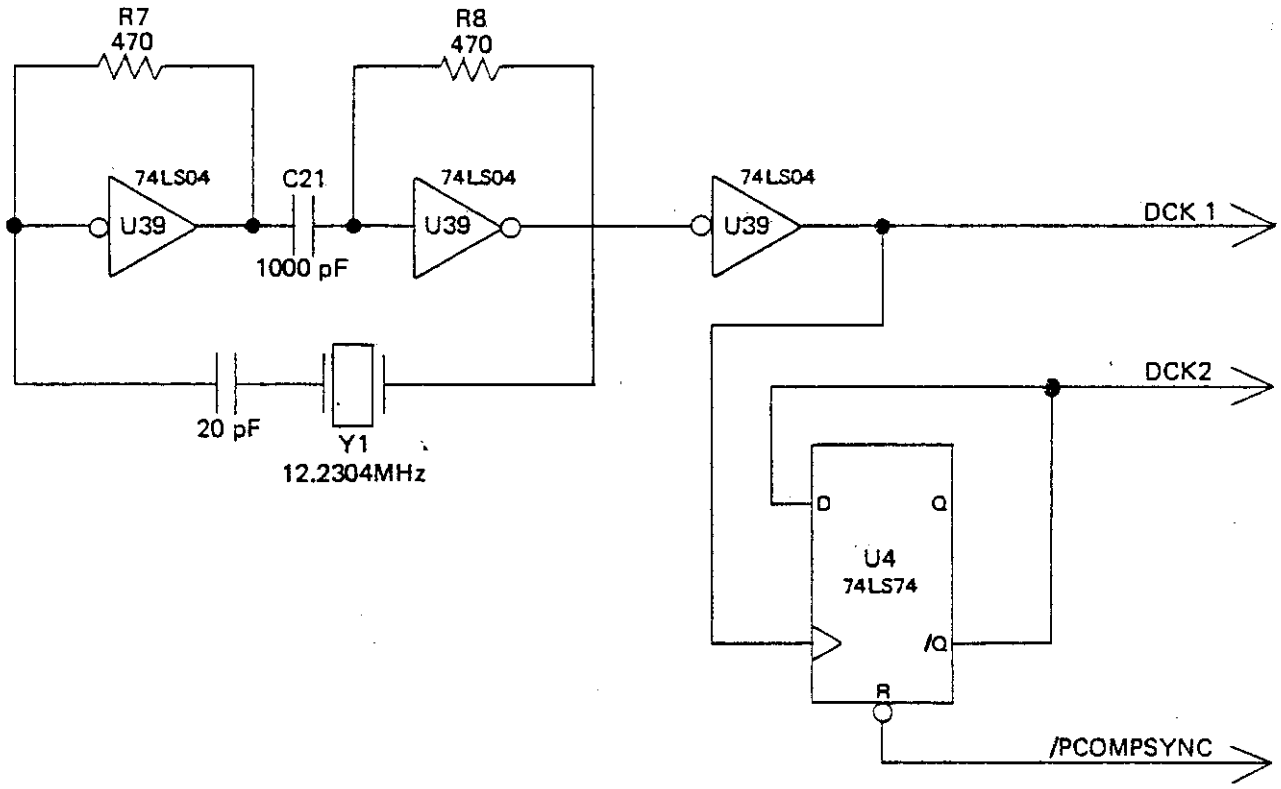


FIGURE III-5 DOT CLOCK

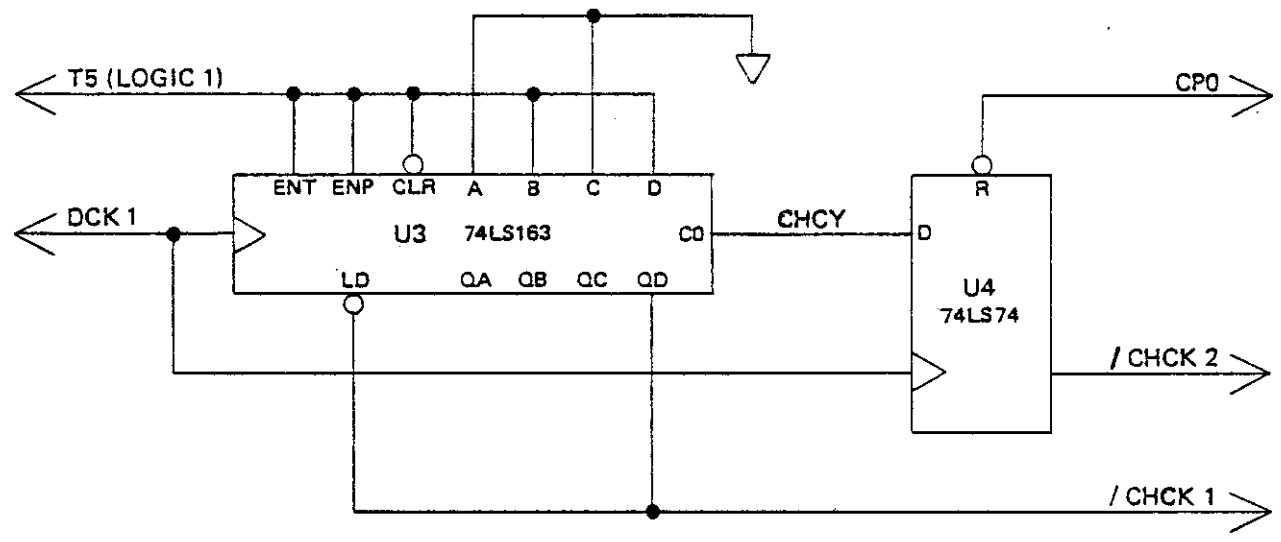


FIGURE III-6 CHARACTER CLOCK

2) CHARACTER CLOCK: The character dot matrix is seven dots wide so the character clock frequency is one-seventh that of the dot clock. It is generated by dividing DCK1 with a 74LS163 configured as a modulo-7 counter (Figure III-6). The counter is pre-loaded with a count of 10 then allowed to count through 15 overflowing to a count of zero. The QD output pulls the /LOAD input low causing a count of 10 to be again loaded. The QD output is high for the first six states and low only for the seventh (zero) state. This is used as the character clock output /CHCK1. A 74LS74 flip-flop generates a signal at half the frequency of /CHCK1, ie., the required character clock for double-wide characters /CHCK2. It is a one-bit shift register which is held in the reset state by CP0 (described in the next paragraph) during the first half of a double wide character. During the second half, the carry output of the 74LS163 CHCY is delayed one period of DCK1 by the 74LS74 to produce /CHCK2. Note that the low-to-high transitions of /CHCK2 are synchronized with the same edge of /CHCK1 and that the leading edge of /CHCK2 has the same timing relationship to DCK2 as /CHCK1 has to DCK1.

3) CHARACTER POSITION COUNTER: The character position counter (Figure III-7) counts the 80 character positions in each scan and generates the timing for blanking during horizontal retrace. It consists of two 74LS163's configured as a modulo-112 counter with input /CHCK1. The states zero through 79 correspond to the 80 character positions. State 79 is decoded by the 74LS11 and the associated inverters. This causes the /LOAD inputs of the 74LS163's to go low during state 79 and 224 to be loaded as the next state. The count continues for 32 states and over-flows back to the zero state. The states 224 through 255 correspond to the horizontal blanking interval. The counter's most significant bit is high only during these states so it's output is used as the horizontal blanking signal PHBLK. The remaining seven outputs drive the character position bus CP0-CP6.

4) SCAN COUNTER: The 240 displayed scans and 20 blanked scans are counted by the modulo-260 counter shown in Figure III-8. It consists of modulo-10 and modulo-26 cascaded counters. The modulo-10 section is a 74LS162 decade counter operating in its fundamental mode. The modulo-26 section consists of two 74LS163's. The counters are loaded with 232 and the states 232 througn 255 correspond to the 240 (24 X 10) displayed scans. The counter overflows to the zero state and states zero and one correspond to the twenty blanked states. The one state is decoded by the 74LS00 and its associated inverter. The decoder causes the /LOAD inputs of the counters to be low during the one state and 232 to again be loaded as the next state. The most significant bit is low only during vertical blanking so its inverted output is used as the vertical blanking signal PVLK.

5) SYNC GENERATION: The horizontal and vertical sync pulses are generated by the circuit shown in Figure III-9. The horizontal sync signal PHSYNC is generated by using a 74LS74 flip-flop to essentially AND together CP3, /CP4 and PHBLK. The resulting signal is high only during the second quarter of PHBLK. The vertical sync signal PVSYNC is generated by using another section of a 74LS74 as a one bit shift register. The first half of PVBLK is decoded then phase-shifted by VCK. Note that PVSYNC is longer than one-half the length of PVBLK because of the phase relationship

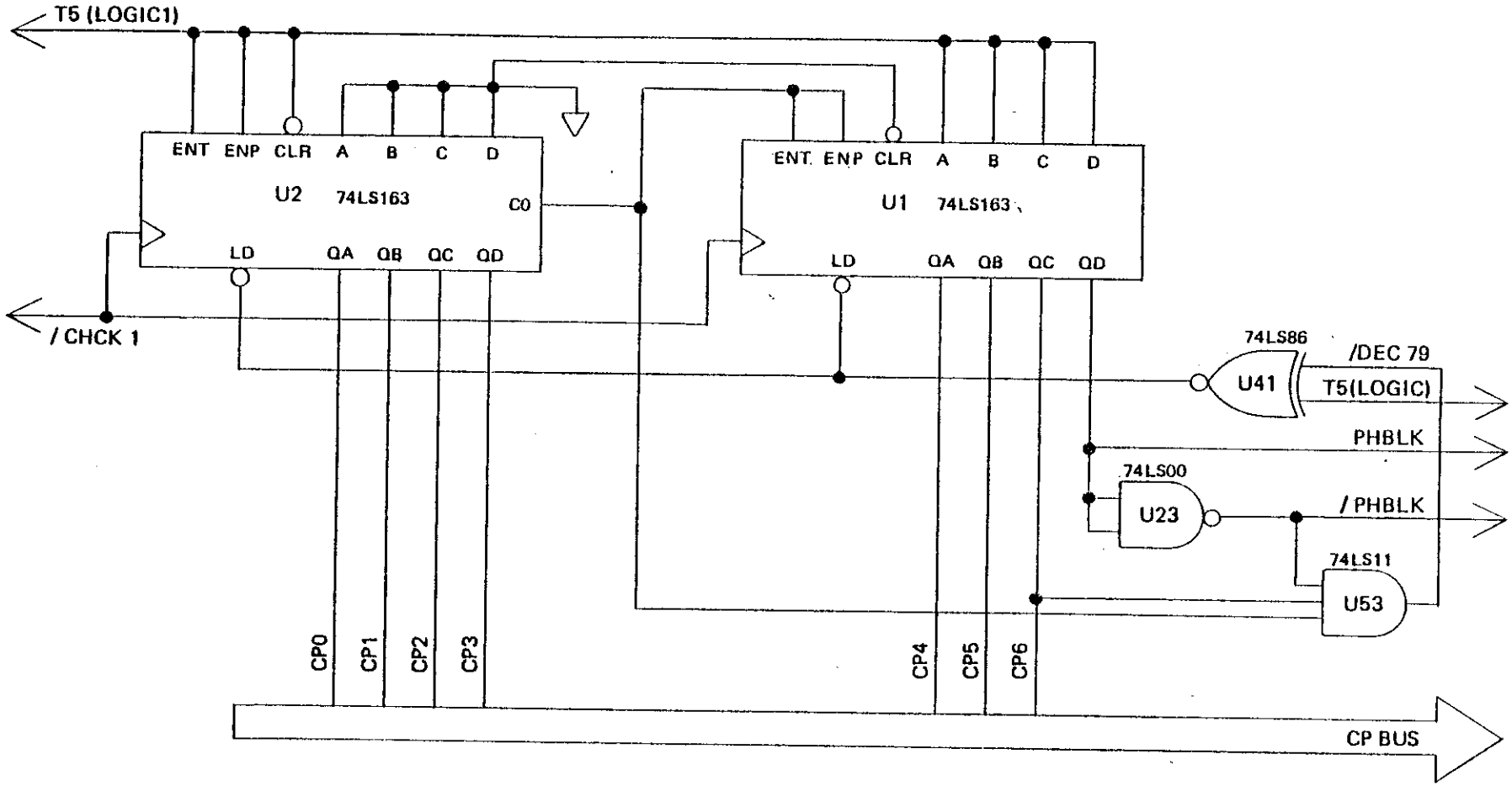


FIGURE III-7 CHARACTER POSITION COUNTER

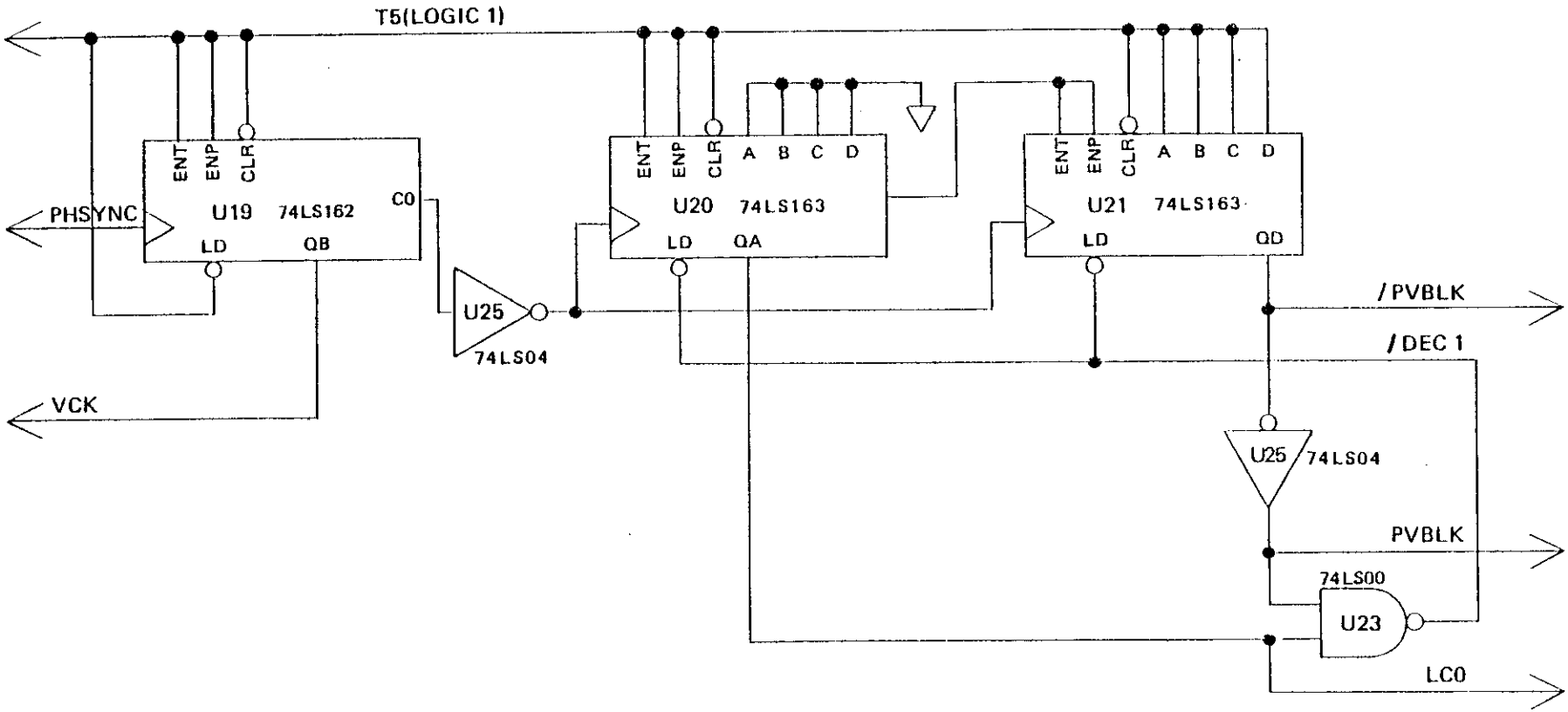


FIGURE III-8 SCAN COUNTER

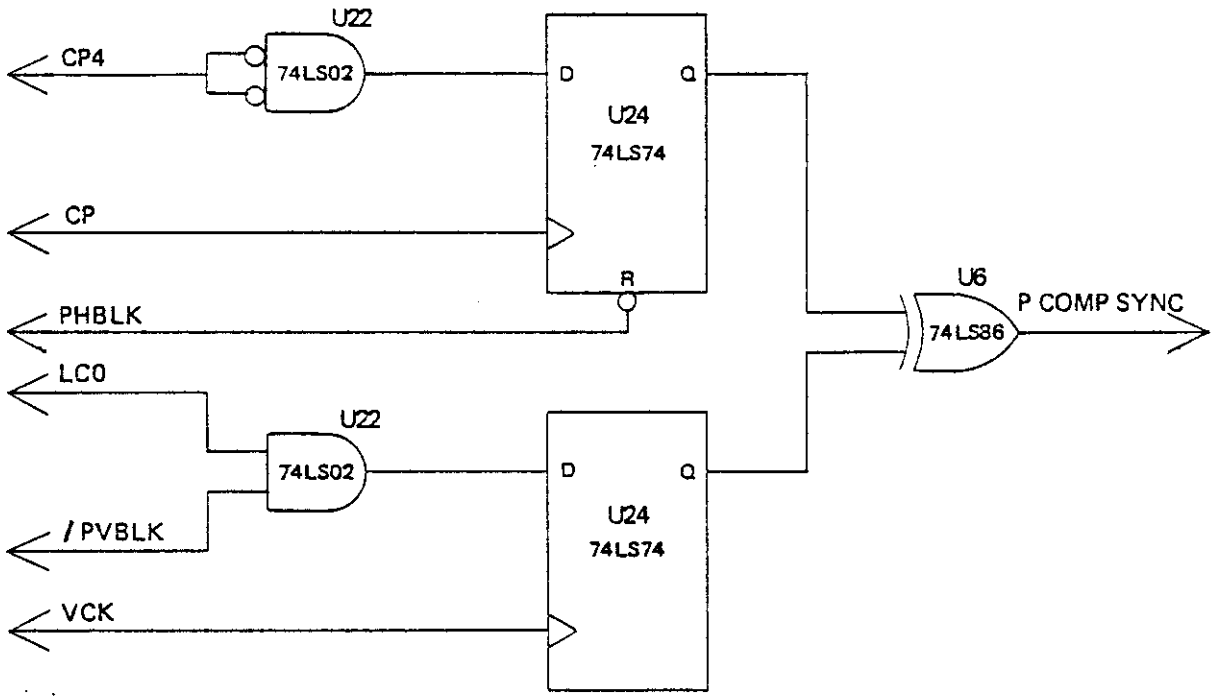


FIGURE III-9 SYNC GENERATION

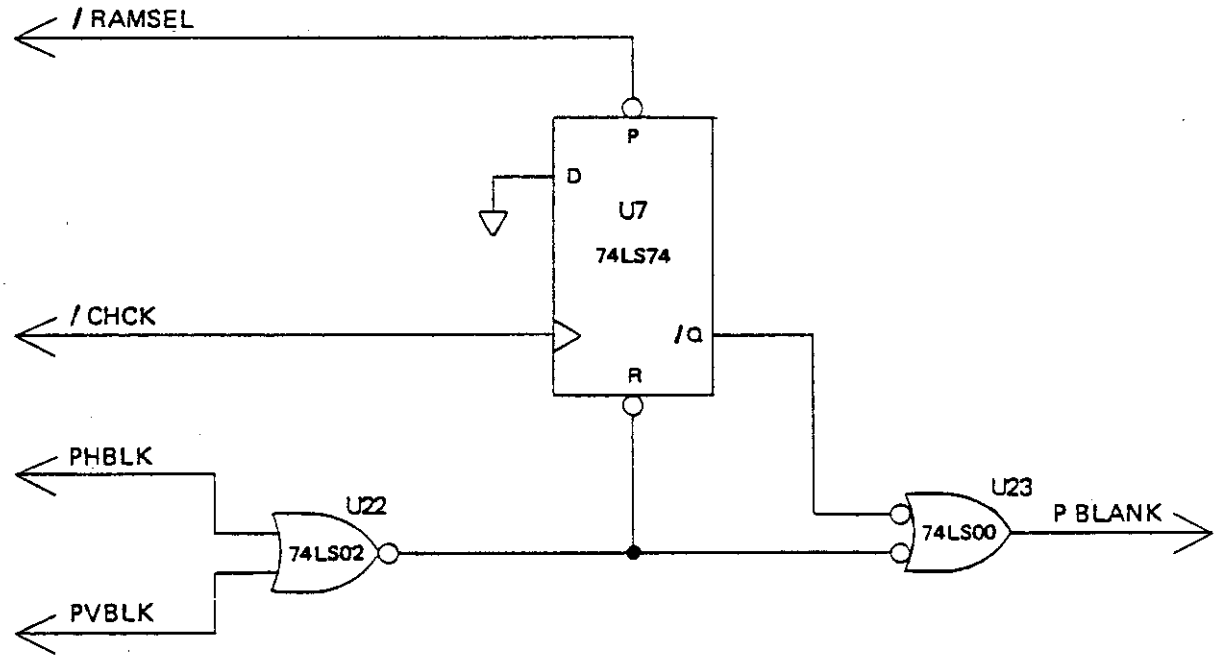


FIGURE III-10 BLANKING GENERATION

between VCK and LC0. The sync pulses are XOR'ed to obtain the composite sync signal PCOMPSYNC. By combining the pulses in this way there is still horizontal sync information available during the vertical sync pulse.

6) BLANKING GENERATION: The horizontal blanking signals (PHBLK and PVBLK) are generated as described above. They are combined in the circuit shown in Figure III-10. In addition this circuit blanks the display when the refresh memory is accessed by the computer. PHBLK and PVBLK are OR'ed with the output of a 74LS74 flip-flop. The flip-flop is set by a computer access of the refresh memory (/RAMSEL low) and is reset by either /CHCK making a low-high transition or by either PHBLK or PVBLK being high. The result is that the circuit output PBLANK is high whenever PHBLK or PVBLK is high and during the short interval when a normally unblanked character is interrupted by a memory access.

b) Refresh Address Circuitry

A significant portion of the VIO's circuitry is associated with providing the refresh memory with appropriate address information via the MA bus. Normally the address provided is the address of the next character to be displayed. The characters are taken in order from memory regardless of screen format. Character addresses are computed from character position in a line, line number and screen format. When the VIO's refresh memory is being accessed by the system, the address on the MA bus is the same as the least significant eleven bits of the system bus. The multiplexing of this address and the character address is obtained using tri-state devices.

1) SCAN/LINE COUNTER: The circuit shown in Figure III-11 is the scan/ line counter. It counts the ten scans in each line and the 12 or 24 lines in each page. The scans are counted by a 74LS162 decade counter operating in its fundamental mode. When a 24 line format is selected the count is advanced once for each raster scan. When a 12 line format is selected the count is prescaled by a 74LS107 J-K flip flop configured as a modulo-2 divider. In this case the count is advanced once for every other raster scan. The outputs of the 74LS162 drive the SCN bus to provide scan number information to the character generator circuitry. The lines of characters on a page are counted by a five bit counter consisting of a 74LS163 and a 74LS107. The outputs of this counter drive the LN bus to provide information for computing character addresses.

2) CHARACTER ADDRESS COMPUTER: The refresh memory address of a character to be displayed is computed from its position in a line, the number of the line and the screen format. The character position counter provides position information via the CP bus. The scan/line counter provides line number information via the LN bus. The screen format is provided by the control signals /L24 and /W80 from the command latch. The four least significant bits of the character address are computed by a linear two to one scaling of the five least significant bits of the CP bus. This is accomplished using 74LS367 tri-state drivers configured as a multiplexor controlled by the line length format (80 or 40 characters). When a 40 character line is selected, bit 3 is inverted for odd numbered lines by XOR'ing this bit with LN0.

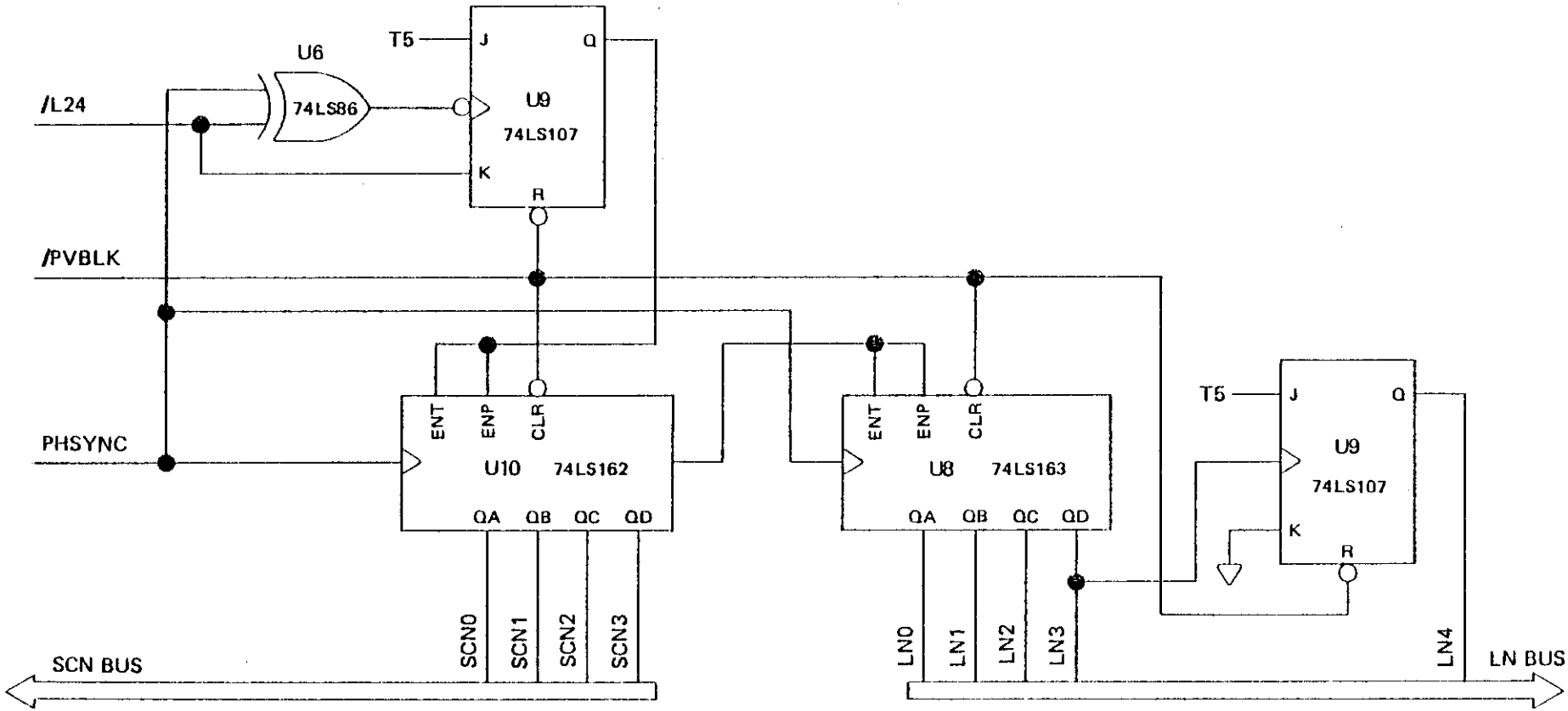


FIGURE III-11 LINE SCAN COUNTER

The high order character address bits are computed using a look-up table implimented in a 512x8 bit PROM.

All of the output structures of the character address computer are tri-state. When the refresh memory is being accessed by the system, all of these outputs are disabled and the MA bus is driven by the system address bus.

c) CHARACTER GENERATION

The character generation circuitry translates the character codes stored in the refresh memory into the dot patterns which form the desired character. The actual translation is performed using 2308/2708 ROM/EPROM's as look-up tables. The data entered in the table is the encoded character (CH bus) and the matrix row number (SCH bus). The data output is seven bits representing the seven dot positions for a row of that character. The ROM's are configured as a 7x10x256 bit memory with two 74LS32 gate sections and two 74LS04 inverter sections providing select decoding.

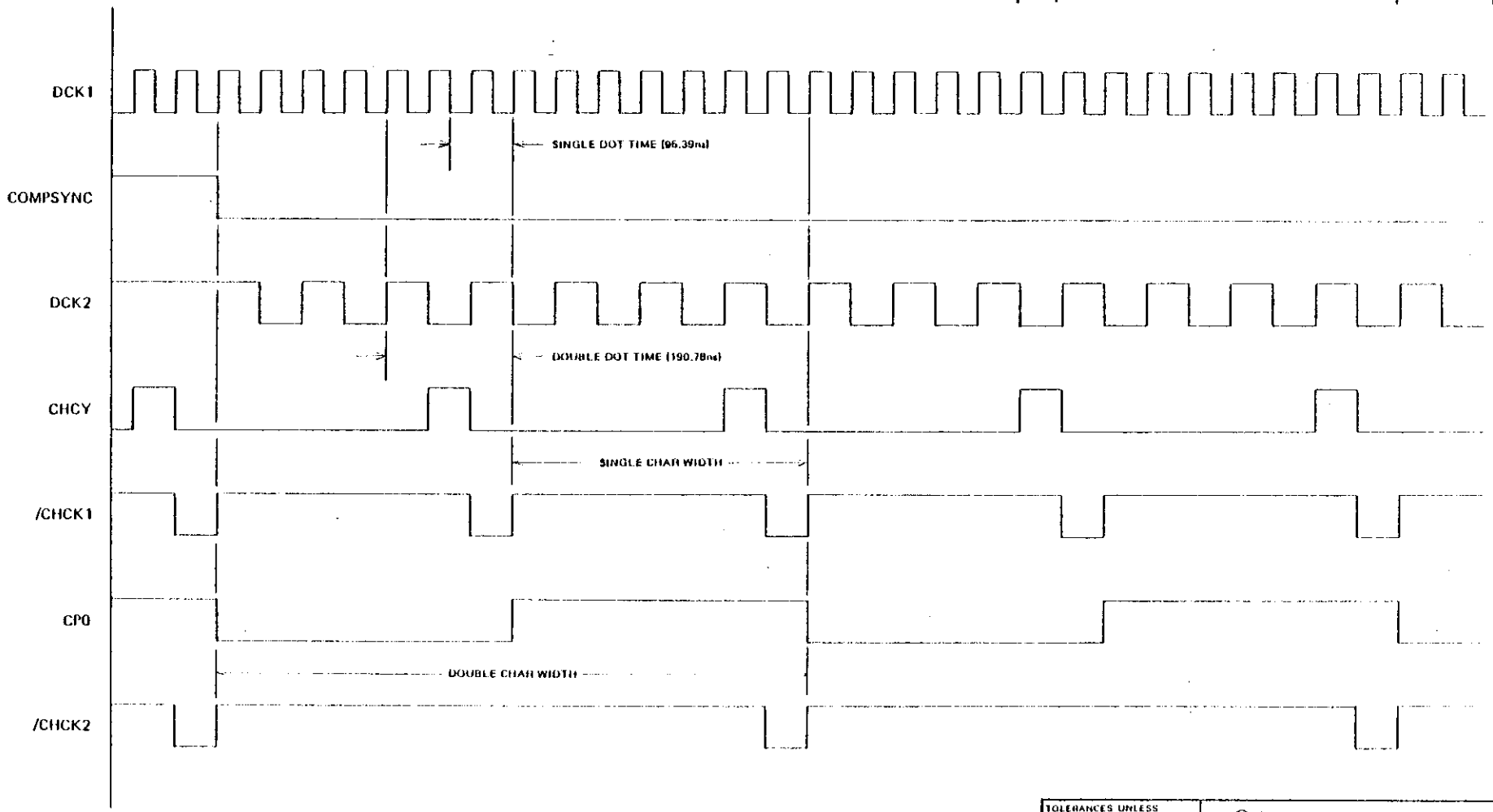
The output of the refresh memory (PCH bus) is first latched in 74LS174 flip-flops. The output of the latch (CH bus) is the encoded character provided to the character generator ROM's. The output of the ROM's is latched by the parallel input of a shift-register (part of video generation circuitry described below). Both latchings are clocked by /CHCK. The intermediate 74LS174 latch is necessary because the total access times of the refresh memory and the ROM's exceed one character time. A side effect of using this technique is that there is a two character time delay from the time when the address of a character is placed on the MA bus to the time when the resulting dot pattern is output by the VIO. As a result, the blanking and sync signals must also be delayed by two character times. This is accomplished using 74LS174 and 74LS74 flip-flops as two bit shift-registers.

d) VIDEO GENERATION

The video generation circuitry serializes the dot patterns output by the character generation circuitry, combines the resulting video with the blanking and sync signals and outputs the combined video signal to the monitor. The parallel to serial conversion is performed by a 74LS166 shift-register. As described above, the parallel output of the character generator ROM's is latched in the shift-register by /CHCK. The patterns are shifted out, clocked by DCK. This raw-video signal is XOR'ed with CH77, the most significant bit of the encoded character. The resulting signal is used for character-by-character reverse video. The 74LS153 mode data-selector selects either this signal or the raw video. The selected signal is XOR'ed with VIDREV for full-screen reverse video.

The video signal is gated with the blanking signal BLANK by a 74LS32 then combined with the sync signal COMPSYNC in a voltage adder circuict. A common-emmiter 2N3904 amplifier provides the impedance matching for the VIO's 72 ohm output.

REVISIONS			
LTR	DESCRIPTION	DATE	APPROVED
0	ORIGINAL	7/77	



TOLERANCES UNLESS OTHERWISE SPECIFIED		
FRACTIONS	DEC.	ANGLES
±	±	±

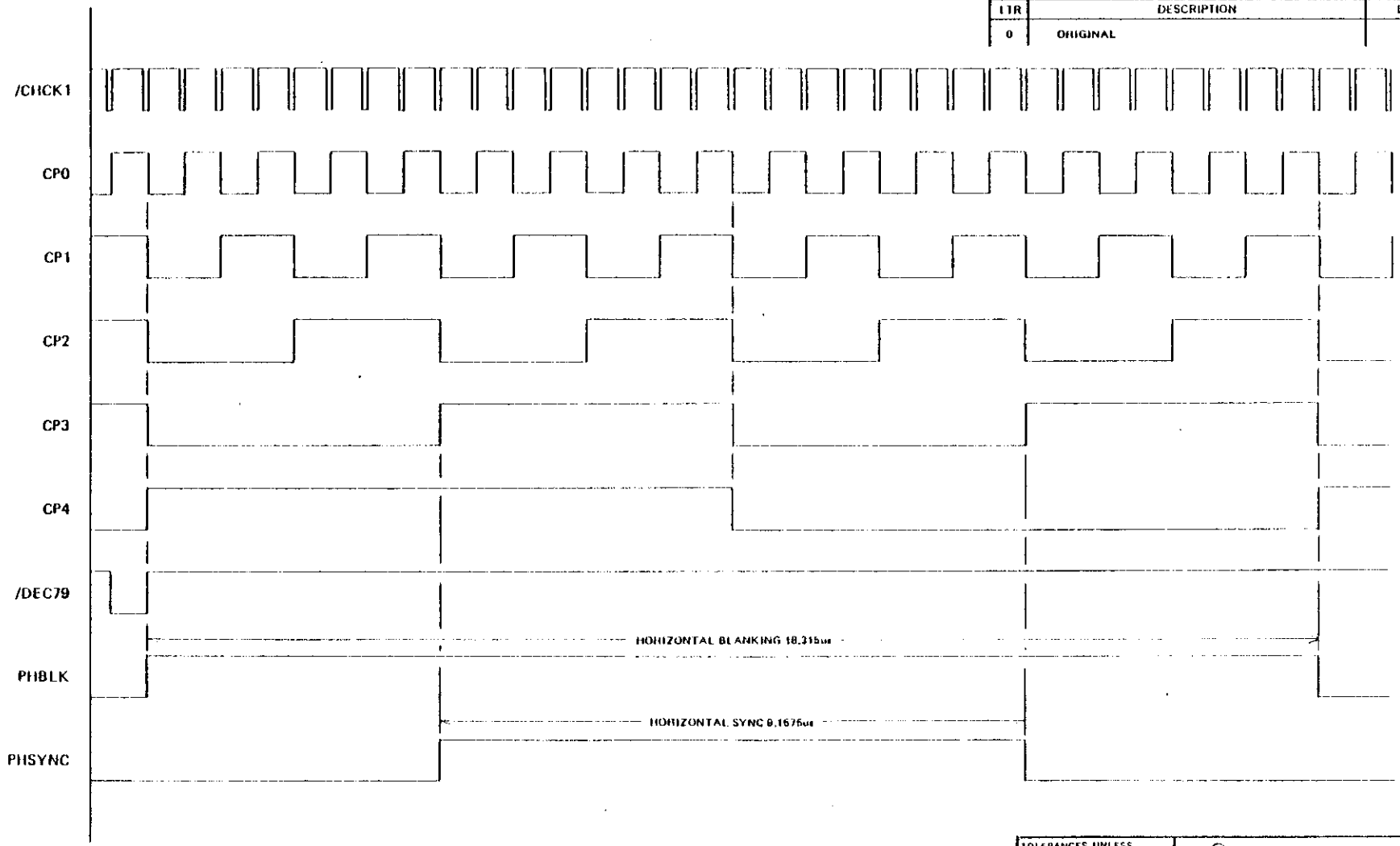
APPROVALS	DATE
DRAWN JCP	
CHECKED PAM	

VIO
DOT AND CHARACTER TIMING

SCALE	SIZE	DRAWING NO.
	B	

DO NOT SCALE DRAWING | SHEET

REVISIONS			
LTR	DESCRIPTION	DATE	APPROVED
0	ORIGINAL	7/77	



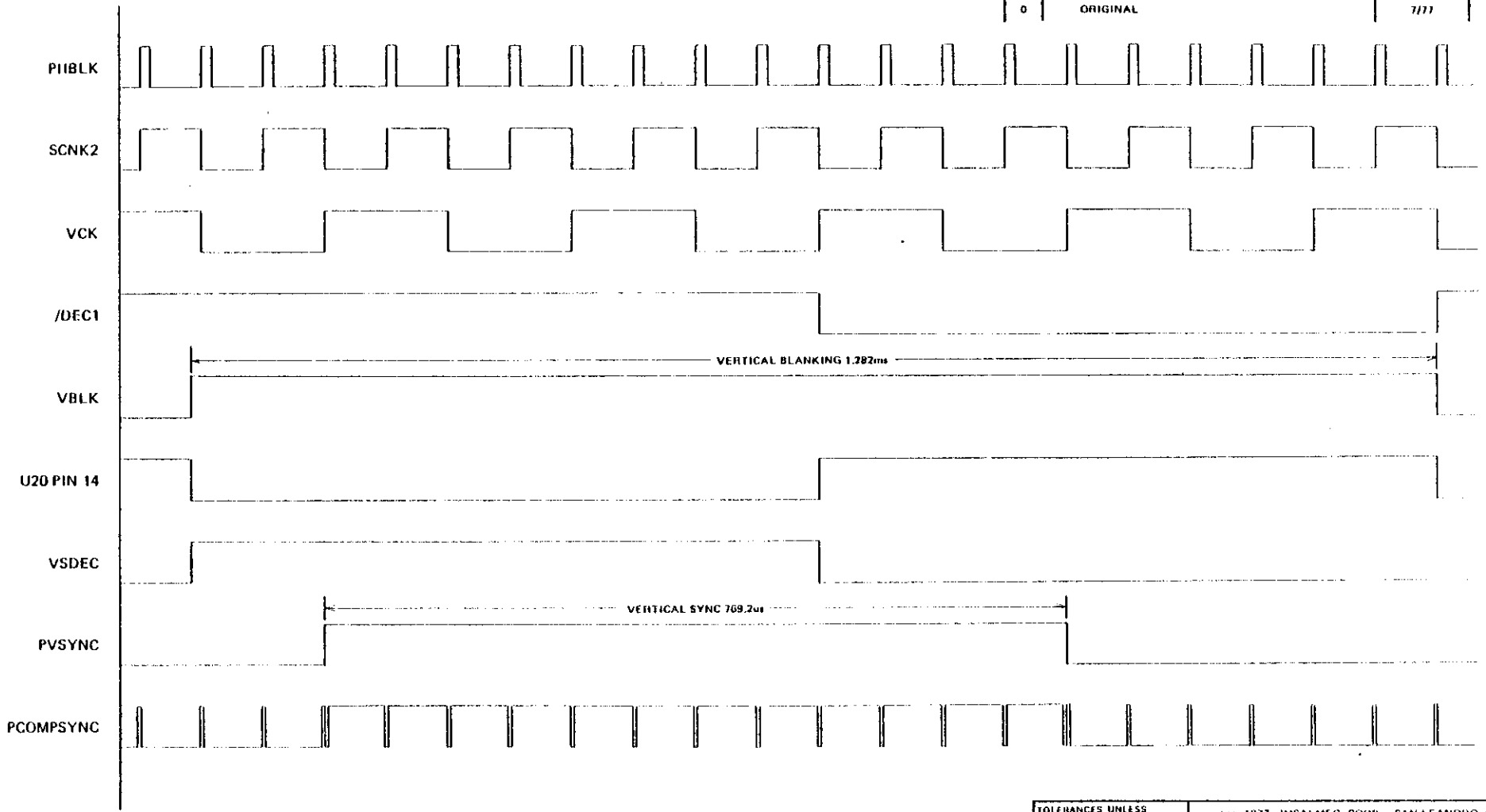
TOLERANCES UNLESS OTHERWISE SPECIFIED	
FRACTIONS DEC. ANGLES	
.1 .4 .5	
APPROVALS	DATE
DRAWN JEB	
CHECKED PRH	

VIO

HORIZONTAL BLANK AND SYNC GENERATION

SCALE	SIZE	DRAWING NO
	B	
DO NOT SCALE DRAWING		SHEET

REVISIONS			
LTR	DESCRIPTION	DATE	APPROVED
0	ORIGINAL	7/77	



TOLERANCES UNLESS OTHERWISE SPECIFIED		
FRACTIONS	DEC	ANGLES
±	±	±

APPROVALS	DATE
DRAWN JEB	
CHECKED	

VIO

VERTICAL BLANK AND SYNC GENERATION

SCALE	SIZE	DRAWING NO
	B	
DO NOT SCALE DRAWING		SHEET

B. USER GUIDE

1. BOARD CONFIGURATION

To properly set-up the VIO board, the board jumpers and switches must be configured for the particular USER application. The board configuration instructions must be followed to insure a properly functioning unit.

SWITCHES 1-8 (SW1)

Switches one through eight are located in position SW1 to the left of U39. Each switch must be set according to the instructions detailed in Steps 1-5 below.

1) Set switches (SW1) 1, 2, 3, and 4 to the OFF position. The VIO board will now occupy the 4K block of memory beginning at address F000 hex.

   NOTE: Switches 1-4 correspond to address bits A12 to A15 respectively. Each switch should be in the OFF position to select a "1", or in the ON position to select a "0" for the appropriate address bit.

2) Set switch 5 (SW1) to the OFF position. The VIOROM firmware is now set to reside in the upper half of the associated 4K block of memory (F800-FFFF).

3) Set SWITCH 6 (SW1) to the ON position. The VIO board will now be selected when either the VIOROM or VIO RAM is being accessed.

TABLE III-1
Address Switches 1-4

ADDRESS	S1	S2	S3	S4
0000-0FFF	ON	ON	ON	ON
1000-1FFF	OFF	ON	ON	ON
2000-2FFF	ON	OFF	ON	ON
3000-3FFF	OFF	OFF	ON	ON
4000-4FFF	ON	ON	OFF	ON
5000-5FFF	OFF	ON	OFF	ON
6000-6FFF	ON	OFF	OFF	ON
7000-7FFF	OFF	OFF	OFF	ON
8000-8FFF	ON	ON	ON	OFF
9000-9FFF	OFF	ON	ON	OFF
A000-AFFF	ON	OFF	ON	OFF
B000-BFFF	OFF	OFF	ON	OFF
C000-CFFF	ON	ON	OFF	OFF
D000-DFFF	OFF	ON	OFF	OFF
E000-EFFF	ON	OFF	OFF	OFF
F000-FFFF	OFF	OFF	OFF	OFF

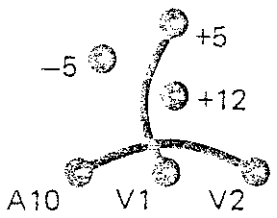


FIGURE III-12 POWER JUMPERS FOR 2316E ROM'S AND 2716/8716 EPROMS

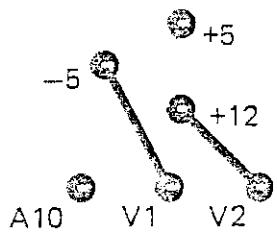


FIGURE III-13 POWER JUMPERS FOR 2308/8308 ROM'S AND 2708/8708 EPROMS

A16/A32/A65

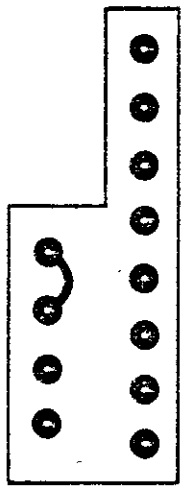
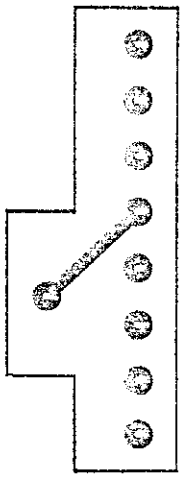


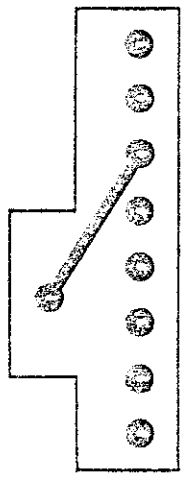
FIGURE III-14 DYNAMIC RAM BOARDS. REMOVE JUMPER FROM A16 TO GROUND ONLY IF USING A16 AS PHANTOM LINE.

FIGURE III-15

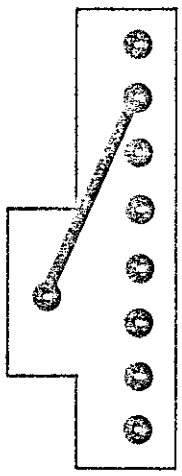
RAM16 B16 JUMPERS



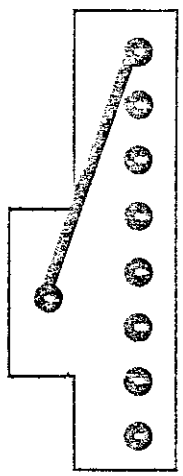
0000-3FFF



4000-7FFF



8000-BFFF

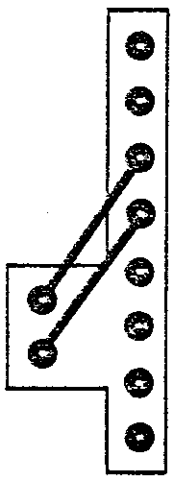


C000-FFFF

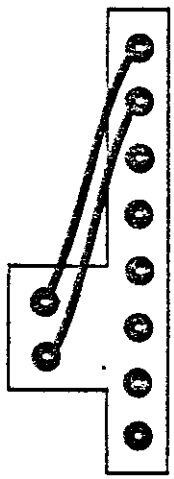
NOTE: VIO USING A16 AS PHANTOM LINE MUST BE INSTALLED.

FIGURE III-16

RAM 32 B32 JUMPERS



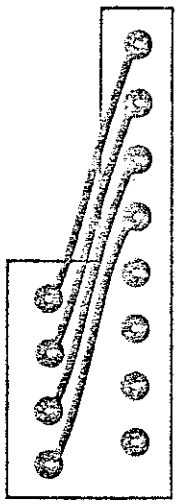
0000-7FFF



8000-FFFF

NOTE: VIO USING A16 AS PHANTOM LINE MUST BE INSTALLED.

FIGURE III-17
RAM 65 865 JUMPERS



0000-FFFF

NOTE: VIO USING A16 AS PHANTOM LINE MUST BE INSTALLED.

4) Set SWITCH 7 (SW1) to the OFF position. The VIO memory will now operate with no WAIT states.

   If it is desired to insert WAIT states into every VIO memory access, set SWITCH 7 to the ON position. Note that the memory devices supplied by IMSAI do not require WAIT states.

5) SWITCH 8 is not used and should be disregarded.

JUMPERS

The board jumper areas are clearly marked both on the board and on the Assembly Diagram. Each jumper area must be configured as required in Steps 6-9 below. Note that assembled VIO's are shipped with the jumpers described in Steps 6 and 7 installed.

6) Connect a jumper between pads "PH" and "B". This will allow the VIO memory to be accessed on the trailing edge of the processor status strobe (as required by 8085 type CPU's).

7) Connect jumpers V1 and V2 as shown in Figures III-12 AND III-13 for the particular type of ROM device used at U46.

   CAUTION!! Incorrect connection of jumpers V1 and V2 may cause damage to the ROM. Make certain that these jumpers are correct for the type of ROM used before applying power to the board. Only the following types of EROM/ROM's may be used at location U46: 2708, 2308, 2716 or 2316E. Do not attempt to use any other types of ROM/EROM devices or damage to the device will result.Ensure that the jumpers do not short to each other, to any board traces, or to any IC pins.

8) If IMSAI RAM-16, RAM-32 or RAM-65 memory boards are not used in the system, leave jumper area "F" open.

   If it is desired to allow the VIO Board to coexist with a full 65K of memory, consisting entirely of RAM-16's, RAM-32's, or a RAM-65 Board, install a jumper in Jumper Area "F", from the pad labelled "F" to the pad labelled "A16". On each memory board, install the jumper shown in Figure III-14, in the A16, A32, or A65 Jumper Area. Re-address each memory board to reside in the second 65K using the jumpers shown in Figures III-15, III-16, and III-17 (B16, B32, or B65 Jumper Areas).

   NOTE: The memory boards are now configured to reside in the second 65K memory space. The VIO will disable the memory boards using address line A16 whenever the VIO is accessed. (A19 may be used in place of A16 if it is so desired). Note that in this configuration, if the VIO is removed from the system, the memory boards will not be accessible unless provision is made to externally drive A16.

9) If an IMSAI IMM board is not being used in the system, Jumper Areas C and D should be disregarded.

If an IMM board is being used, the VIO board may be addressed to reside in the top 65K of the Megabyte Address Space. To do this, cut the two traces connecting the pads labelled "C" and "D" to the pads labelled A15 and A14 respectively. Then install a jumper between the pad labelled "AA15" and the pad labelled "C". Also install a jumper between the pad labelled "AA14" and the pad labelled "D". Install a jumper at location "M".

Refer to the IMSAI IMM Documentation for further details.

2. VIO OPERATION WITH VIOROM

The VIOROM contains two separate and distinct programs. The first is a firmware driver which will allow a user program to easily control all VIO functions and screen displays, (new releases of IMSAI Software will include provisions for using the VIOROM Firmware). The second is a complete system monitor program, IMSAI PCS-80/30, which gives the user a number of extremely useful memory and input/output functions when the VIO and monitor are used with a keyboard or other input device.

Since operation with the VIOROM depends upon the Users application, the User may proceed in one of three ways. If it is desired to use the display functions of the VIO with existing User Software, proceed to "VIO Firmware Driver". If it is desired to exercise the display functions of the VIO and VIOROM Firmware Driver, proceed to "VIO Test Routine". If it is desired to operate the VIO as an Intelligent Video Terminal, proceed to PCS-80/30 Monitor.

a) VIO Firmware Driver

The VIO firmware driver can be called from the user program whenever it is desired to initialize the VIO, place a character on the screen display, or execute a control sequence.

1) SCREEN INITIALIZATION: To initialize the VIO, the user program should CALL the INITIALIZATION ENTRY POINT at F800 H. The firmware will then

clear the screen
put the cursor in the upper left corner (HOME)
select an 80x24 screen format
select UPPER CASE display (ASCII text mode)
activate the screen scroll mode

Following the initialization routine, control will return to the user's calling program.

2) CHARACTER DISPLAY: To display a character on the screen, the user program must CALL the CHARACTER OUT entry point, F803H, with the character code (for the character to be displayed), in the Accumulator. Prior to the CALL, the stack must contain at least 30 bytes of available STACK SPACE. Once the character has been displayed, control will return to the user's calling program with none of the registers or status bits affected.

A list of character codes appears in Appendix 1.

The characters which may be displayed on the screen will depend entirely on the mode

of operation selected by the user: ASCII text mode, extended text mode, or graphic mode. These modes are selected with the ESCAPE SEQUENCES, described in the next section.

Note: characters may also be displayed by directly accessing the refresh memory (as described in section III-A,6-b).

ASCII TEXT MODE: When the ASCII Text Mode is selected, any of the 96 characters represented by the character codes 20-7FH (see Appendix 1) may be displayed through the firmware. This includes the standard 96 character ASCII set (upper and lower case plus punctuation). Character-by-character reverse video is available and both CONTROL CHARACTER COMMANDS and ESCAPE SEQUENCES will be accepted.

To display a character in the ASCII Text Mode, the User program must CALL the Character Out Entry Point, F803 H. Prior to the CALL, the lower 7 bits of the Accumulator must contain a valid ASCII Text Character Code. Bit 7 of the Accumulator will act as the character-by-character reverse video flag (0=positive video, 1=reverse video).

Displayable ASCII Text Character Codes consist of the lower 7 bits of CODES 20-7F H listed in Appendix 1. Note that these codes correspond to the standard 7 bit ASCII Codes 20-7F H.

For example, if we wanted to display an ASCII "1" in positive video, we would CALL the Character Out Entry Point, F803 H, with the code 31 H in the Accumulator. The same code, with bit 7 "on" would cause the ASCII "1" to be displayed in reverse video. Thus if the Accumulator contained a B1 H, the ASCII "1" would be displayed as a dark character on a light background (reverse video).

Note that character codes 00-1F H are not displayable through the firmware (in the ASCII Text Mode). However, they may be displayed by directly accessing the VIO's refresh memory (III-A,6-b).

In most cases, existing User Software may easily be modified to be used with the VIO.

1. Locate the output driver routine.
2. Substitute a CALL to the Character Out Entry Point (CALL F803 H), for the status port input instruction (directly preceding the output instruction to the terminal device).
3. Ensure that the character to be output, is in the Accumulator prior to the CALL.
4. "NOP" out the rest of the driver up to but not including any RET instruction.

For example, suppose the existing output driver appears as follows:

```
1950  DB 03      TOUT     IN   TTY+1        ;GET STATUS
1952  1F                  RAR               ;TEST IF READY
1953  D2 50 19            JNC  TOUT         ;LOOP IF NOT RDY
1956  F1                  POP  PSW          ;GET CHAR
1957  D3 02               OUT  TTY          ;OUT TO TERMINAL
1959  C9                  RET
```

This driver may be modified to be used with the VIO firmware as follows:

```
1950  F1          POP  PSW          ;GET CHAR
1951  CA 03 F8    CALL VIO          ;DISPLAY CHAR
1954  00          NOP
1955  00          NOP
1956  00          NOP
1957  00 00       NOP!NOP
1959  C9          RET
```

EXTENDED TEXT MODE: When the Extended Text Mode is selected, any of the 96 graphic characters, represented by the character codes C0-FFH (see Appendix 1) may be displayed through the firmware. Character-by-character reverse video is available and both CONTROL CHARACTER COMMANDS and ESCAPE SEQUENCES will be accepted.

To display a character in the Extended Text Mode, the User program must CALL the Character Out Entry Point (CALL F803 H). Prior to the CALL, the lower 7 bits of the Accumulator must contain a valid Extended Text Character Code. Bit 7 of the Accumulator will act as the character-by-character reverse video flag (0=positive video, 1=reverse video).

Displayable Extended Text Character Codes consist of the lower 7 bits of Codes A0-FF H listed in Appendix 1. Note that Codes A0-FF H correspond to a partial block drawing and a complete line drawing graphic character set. This set is a subset of the one available in the Graphic Mode, and it is fully explained in the Graphic Mode section (below).

For example, if we wanted to display in positive video, the vertical line segment represented by the Code C5 H in Appendix 1, we would CALL the Character Out Entry Point F803 H, with a 45 H in the Accumulator. The same Code with bit 7 "on" would cause the character to be displayed in reverse video. Thus if the Accumulator contained a C5 H, the vertical line segment would be displayed as a dark character on a light background (reverse video).

Note that Codes 80-9F H are not displayable through the firmware (in the Extended Text Mode). However, they may be displayed by directly accessing the VIO's refresh memory (Section III-A,6-b).

While the Extended Text Mode currently allows graphic characters to be displayed, it is ideally suited for the display of foreign language fonts. This would require the upper half of the character generator PROM's to be appropriately reprogrammed.

GRAPHIC MODE: When the Graphic Mode is selected, any of 255 ASCII and graphic characters may be displayed (codes 00-1A H and codes 1C-FF H). The character codes appear in Appendix 1.

While this mode allows both ASCII and graphic characters to be displayed, it does not allow the use of character-by-character reverse video. Any new CONTROL CHARACTER COMMANDS will not be accepted. Any CONTROL CHARACTER COMMANDS in effect prior to the entry into the Graphic Mode will remain valid and will still affect character display. ESCAPE sequences will be accepted and provide the only means of control when in the Graphic Mode.

To display a character in the Graphic Mode, the User program must CALL the Character Out Entry Point F803 H. Prior to the CALL, the Accumulator must contain a valid 8 bit Graphic Character Code.

All Codes 00-FF H (listed in Appendix 1) are displayable in the Graphic Mode, with the exception of Code 1B H. The character codes 00-1F H correspond to 32 special characters. The character codes 20-7F H, correspond to the standard ASCII codes 20-7F H. The characters represented by the codes 80-FF H, comprise a complete line drawing and block drawing graphics set.

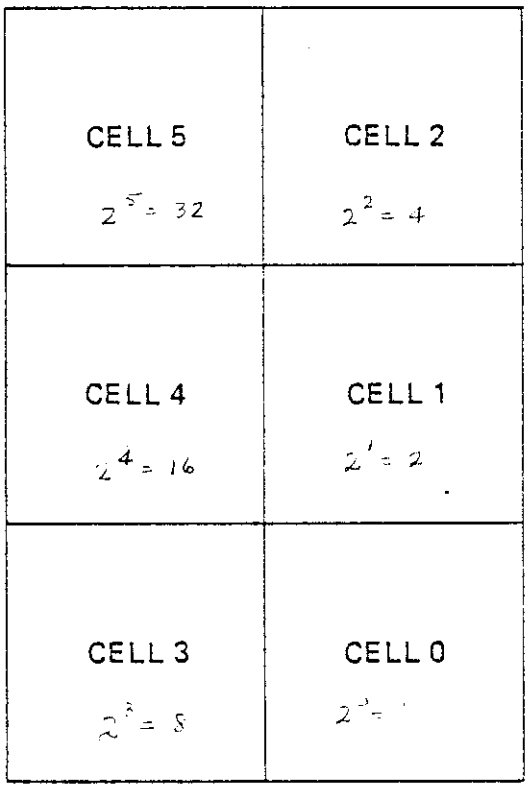
For example, if we wanted to display an ASCII "A", we would CALL the Character Out Entry Point, F803 H, with a 41 H in the Accumulator. Similarly, if we wanted to display the line segment represented by the Code C1 in Appendix 1, we would CALL the Character Out Entry Point F803 H, with a C1 H in the Accumulator.

Note that Character Code 1B H is not displayable through the firmware (in the Graphic Mode). However, it may be displayed by directly accessing the refresh memory (Section III-A,6-b).

Codes 80-BF H correspond to the Block Drawing Graphic Set. The display grid for this set uses a 7x10 dot matrix font, equally divided into 6 cells or display areas as shown in Figure III-18. The characters in the Block Drawing Set consist of the patterns formed by using these six cells in various combinations. Bits 0-5 of the character codes 80-BF H correspond to the six display cells as shown in Figure III-18. A "one" in any of these bit positions will cause the corresponding cell to be used in the display character. For example, if it is desired to display the character formed by using cell 0 and cell 2, the character code will contain a "1" in bits 0 and 2. Bits 1, 3, 4, and 5 will be "0", since cells 1,3,4, and 5 are not used in the display character. Since bit 6 is a "0" and bit 7 is a "1" for all Block Drawing Characters, the resultant character code is B5 H (10000101).

Codes C0-FF H correspond to the Line Drawing Graphic Set. The display grid for this set uses the 7x10 dot matrix font to create the six line segments shown in Figure

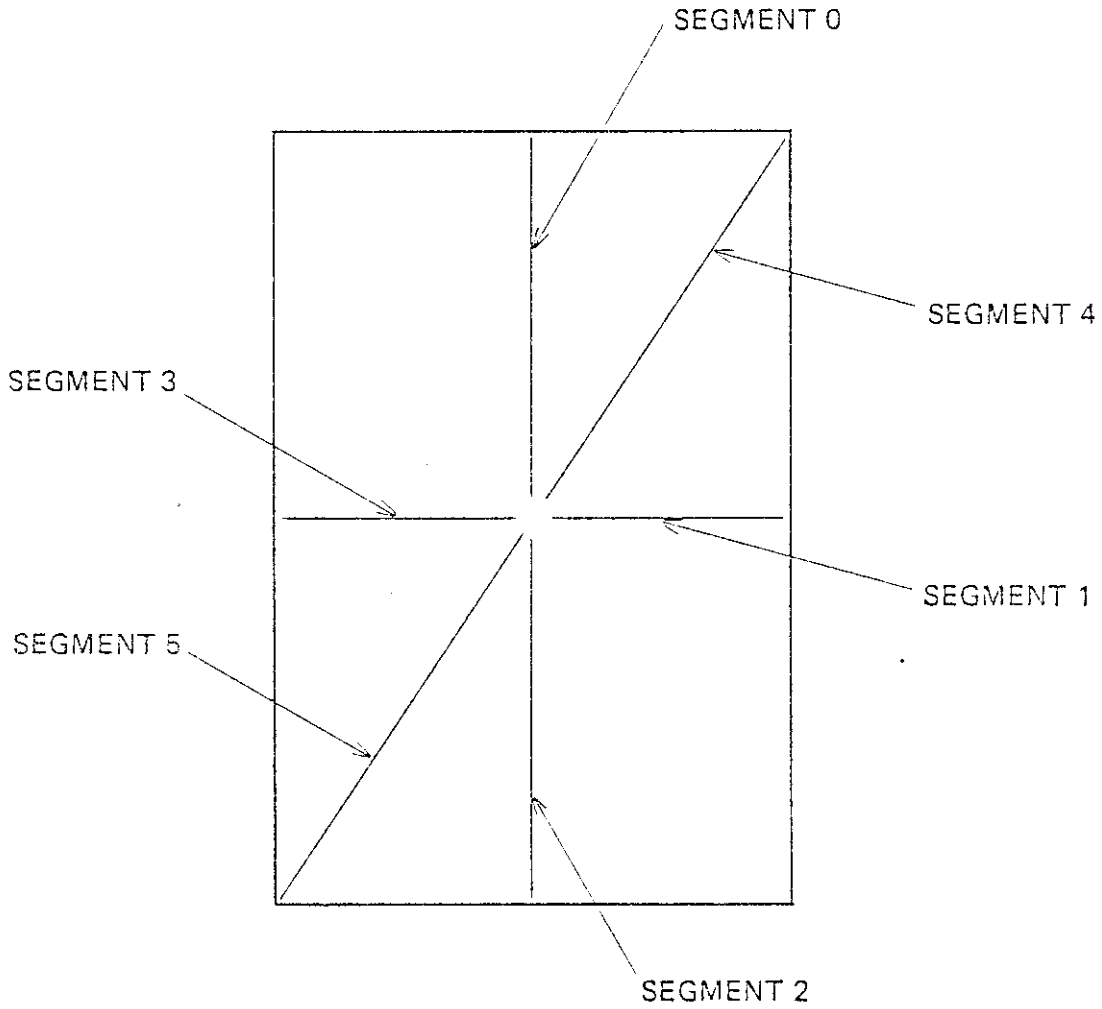
FIGURE III-18 THE CHARACTER CODE – DISPLAY RELATIONSHIP FOR THE BLOCK DRAWING GRAPHIC SET



1	0	CELL 5	CELL 4	CELL 3	CELL 2	CELL 1	CELL 0

CHARACTER CODES 80–BF H.

FIGURE III-19 THE CHARACTER CODE – DISPLAY RELATIONSHIP FOR THE LINE DRAWING GRAPHIC SET



1	1	SEGMENT 5	SEGMENT 4	SEGMENT 3	SEGMENT 2	SEGMENT 1	SEGMENT 0

CHARACTER CODES C0–FF H.

III-19. The characters in the line drawing set consist of the patterns formed by using these six line segments in various combinations. Bits 0-5 of the character codes C1-FF H, correspond to the six display segments as shown in Figure III-19. A "one" in any of these bit positions will cause the corresponding segment to be used in the display character. For example, if it is desired to display the character formed by using segments 1 and 2, the character code will contain a "1" in bits 1 and 2. Bits 0, 3, 4, and 5 will be "0", since segments 0, 3, 4, and 5 are not used in the display character. Since bit 6 is "1" and bit 7 is "1", for all Line Drawing Characters, the resulting character code is C6 H (11000110).

3) CONTROL SEQUENCES: There are two basic methods of controlling or modifying VIO operations: CONTROL CHARACTER COMMANDS and ESCAPE SEQUENCES.

CONTROL CHARACTER COMMANDS

To execute a CONTROL CHARACTER COMMAND, the user program should CALL the CHARACTER OUT entry point at F803 H, with the desired control character in the Accumulator. Prior to the CALL, the stack must contain at least 30 bytes of availiable stack space. Following the execution of the command, control will return to the user's calling program with none of the registers or status bits affected. Note that CONTROL CHARACTER COMMANDS will not be accepted when operating in the Graphics Mode.

Carriage Return (0DH)	causes the cursor to return to position 1 of the current line and forces the overwrite mode.
Line Feed (0AH)	causes the cursor to move one line down in the same column position
Up Cursor (0BH or CTRL-K)	causes the cursor to move one line up in the same column position
Forward Cursor (0CH or CTRL-L)	causes the cursor to move non-destructively to the right one position. An auto line feed/ carriage return occurs at the end of the line.
Back Cursor (08H or CTRL-H)	causes the cursor to move non-destructively to the left one position. The cursor will not move past position 1.
Home Cursor (1EH or CTRL-^)	Moves cursor to column 1, row 1
Erase Screen (1AH or CTRL-Z)	Fills screen with blanks and homes cursor.
Clear to end of Field (15H or CTRL-U)	Clears the characters from the cursor to the end of the line or until the existence of the first protected character, whichever comes first.

Tab (09H or CTRL-I)	Causes the cursor to move to the next tab stop that is not within a protected field or the first unprotected character following the next protected field or the home position, whichever comes first.
Delete Character (7FH or rubout)	Removes the character under the the cursor by shifting the remainder of the line one place to the left. The right-most character is filled with a blank.
Insert Mode (14H or CTRL-T)	Toggles the system in and out of the insert mode. In the insert mode, any character entered at the cursor causes the remaining characters of the line to be shifted to the right one place. The last character on the line is lost to the bit bucket. When not in the insert mode, the system is in the overwrite mode. In the overwrite mode any new character output just overwrites any previous character in that position.
Delete Line (04H or CTRL-D)	Deletes the current line indicated by the cursor by moving all lines below the current line up one line and filling in the last line with blanks. The cursor is returned to the beginning of the line.
Enter Line (05H or CTRL-E)	All lines from the current line to the end of the screen are moved down one line with the last line being lost to the bit bucket. The former current line is then blanked and the cursor returned to position 1.
Protect Fields/Unprotect Fields (10H or CTRL-P)	All inverted characters currently existing or subsequently entered are treated as protected fields when the protect mode is active. When inactive, there is no significance to the inverted video other that visual. When protected, these fields will not allow entry of data in most cases and affect such things as Tabs, and cursor control.

ESCAPE SEQUENCES

To execute an ESCAPE SEQUENCE, the User program should successively place each Escape Sequence character in the Accumulator and CALL the CHARACTER OUT entry point at F803 H. All Escape Sequences begin with the ESCAPE character 1B H, followed by the appropriate character or characters listed below. Following the execution of the Escape Sequence, control will return to the User program, with none of the registers or status bits affected.

Set/Clear Tab (49H or "I")	Set a Tab stop at the current cursor position or clear an existing tab stop at current cursor position.
Clear All Tabs (09H or CTRL-I)	Unconditionally clear all all Tab positions
ASCII Text Mode (54H or "T")	Standard ASCII Text Mode allows for the display of the 96 character ASCII set (upper and lower case with punctuation). Character codes 20-7FH may be displayed (see Appendix 1). Character-by-character reverse video is available in this mode.
Extended Text Mode (45H or "E")	Extended Text mode currently allows for the display of 96 graphic characters but could be used to display foreign language fonts (when the upper half of the character generator ROMs are appropriately programmed). Character codes A0-FFH may be displayed (see Appendix 1). Character-by-character reverse video is available in this mode.
Graphic Mode (47H or "G")	Graphic Mode allows 255 ASCII and graphic characters to be displayed. Codes 00-1AH and Codes 1C-FFH may be displayed (see Appendix 1). Character-by-character reverse is NOT available in this mode. Note that new Control Character Commands will not be accepted, however, escape sequences will be accepted.

Scroll Mode Toggle (53H or "S")	Alternately places the screen mode scroll or wrap-around mode. The Text or Extended Text Modes default to the scroll option while the Graphic Mode defaults to the Wrap-around Option. Note that the scroll option may not be used in the Graphic Mode.
Upper/Lower Case Toggle (55H or "U")	Alternately selects the Upper case only or Upper/Lower case display. Defaults to Upper Case only.
Inverse Video Toggle (56H or "V")	Alternately selects positive video (white characters on a black background) or inverse video (black characters on a white background). Default is positive video (white on black).
Lines Per Page Toggle (4CH or "L")	Alternately selects 12 line pages or 24 line pages. Default is 24 line pages.
Columns per Line Toggle (43H or "C")	Alternately selects 40 column lines or 80 column lines. Default is 80 column lines.
Addressable Cursor (=YX)	Places the cursor at the position defined by y-axis (line number) and X-axis (column number). Y-axis and X-axis position codes may be determined from Appendix 2.

b) PCS-80/30 Monitor

The monitor program allows the VIO to operate as an intelligent terminal when used with an ASCII encoded keyboard or other suitable ASCII encoded input device. It allows for keyboard control of all VIO firmware and hardware features and provides a number of extremely useful memory and I/O commands.

1) INPUT DEVICES

The PCS-80/30 monitor commands allow for the use of a keyboard, cassette recorder and a paper tape reader. It is ideally suited for use with the IMSAI MIO board.

1) To use the monitor, an ASCII encoded keyboard or other suitable ASCII encoded input device must be used. Configure the input interface (MIO) so that input data appears on Port 02H. The input interface (MIO) must also be configured to drive Bit 1 of Port 03H with an input data ready or input data available signal.

2) If the cassette commands are to be used with an IMSAI MIO cassette port, configure the MIO cassette port so that cassette data appears on Port 00H. Configure the cassette status so that Bit 2 of Port 03H is driven by the cassette data ready or cassette data available signal.

3) If the paper tape command is to be used with a Teletype device, configure the input interface (MIO) so that the teletype input data appears on Port 02H. Configure teletype status so that Bit 1 of Port 03H is driven by the input data ready or input data available signal.

2) MONITOR OPERATION

1) The monitor ENTRY POINT is located at F806H. The display will come-up in the default mode:

40x12 screen format
Upper case only
ASCII Text Mode
Screen Scroll Mode

2) Any time a Hex parameter is requested from the user, any number of hex digits will be accepted, but only the last 4 (if a 16 bit value is requested) or the last 2 (if an 8 bit value is requested) will be used. The user may use any type of field delimiter (i.e., ",", "b", ";", etc.).

3) MONITOR COMMANDS

The following is a list of valid PCS-80/30 commands. In all cases the symbol @ represents CARRIAGE RETURN.

DISPLAY MEMORY

D@
D<start>@
D<start>,<end>@

Display the contents of the memory locations beginning at the start address (or 0 if not specified) and continuing until the end address if specified. If no end address is specified only one single location will be displayed.

EXAMINE AND MODIFY MEMORY

E<addr>@ [XX]@

Display the contents of memory location addr. The user then has four options:

a. hitting the space bar will display the next successive location;

b. hitting the minus key (-) will display the previous location;

c. hitting carriage return will terminate the operation; or

d. entering 2 or more valid Hex characters (followed by 1, 2, or 3 above) will modify memory as entered and perform 1, 2, or 3 above.

FILL MEMORY

F<start>,<end>, XX@

Fill memory from starting address through ending address inclusive with specified XX byte.

To Zero all of memory the following could be used (do not zero refresh memory on VIO): F0,EFFF,0@

MOVE MEMORY

M<start>,<end>,<dest>@

Move the block of memory starting at start address through end address to the destination address. The move occurs byte by byte from low memory to high memory. Hence, the following command will fill memory locations 100-1FF with the same byte that is at location 100:

M100,1FE,101@

SEARCH MEMORY

S<start>,<end>,<16 bit value>,<16 bit mask>@

Search memory from starting address through ending address for all instances of 16 bit values which, when ANDed with the specified mask yield the 16 bit specified value. If no mask is specified, FFFF is assumed which implies a full 16 bit compare.

EXAMPLE: You have a version of BASIC in memory locations 0-1FFF and wish to locate the status input instructions so you can locate and modify the output driver for use with the VIO.

S0,1FFF,DB03@ will print the address and contents of all memory locations which contain DB03.

It should be noted that specifying less than 4 digits in the value, or mask, might produce unexpected results since all inputted hex values are left padded with zeros.

VERIFY MEMORY

V<start>,<end>,<dest>@

Do a byte by byte compare from start through end to the destination. Print each pair of bytes which are not identical in the 2 strings.

PROTECT/UNPROTECT RAM 4A MEMORY

P<start>,<end>@
U<start>,<end>@

Protects all 1K blocks of memory (RAM 4A-4 only) which fall within the specified memory addresses. Any 1K blocks which overlap the boundary will also be protected.

JUMP TO MEMORY

J<start>@

Jump to memory at location start.

CALL MEMORY

C<start>@

Push the start address of the monitor on the stack and Jump to memory address start. If the program entered does not alter the stack and terminates with a Return instruction, the monitor will be properly re-entered.

PERFORM DIRECT I/O

I<port #>@
O<port #>,XX@

Input or output from/to the specified port #. If input then display the resulting 8 bit value on the VIO. If output then output the specified value.

This instruction causes an IN (DB) or OUT (D3) instruction with the appropriate port # to be stuffed into RAM in the MPU-B memory and then called.

Loc	Inst	
10FA	DB or D3	IN or OUT
10FB	XX	Port #
10FC	C9	RET

The contents of memory locations 0, 1, and 2 will therefore be destroyed. The following command will change the RAM direct I/O ptr:

Y<addr>@

The address specified will now be used along with addr+1 and addr+2.

MEMORY DIAGNOSTIC

T@
T<start>,<end>@

Test memory from start address to end address. Every memory cell is cycled through all 256 bit patterns and tested to see if it contains the correct pattern. If all memory within the specified range is good, a prompt "?" will appear. If a bad cell or non-existent cell is located, the following display will occur:

AAAA VV SS

where: AAAA is the bad address
VV is the value in the memory
SS is what the value should be

Using the form T@ begins the memory test at location 0 and continues until the first bad cell, non-existent cell, or ROM memory is found.

The memory diagnostic DOES NOT destroy the contents of the memory tested.

GENERATE SYNC STREAM

G@

ADJUST AND ALIGN CASSETTE RECORDER

A@

EXEC FROM CASSETTE FILE

X@
X<start>,<end>,<exec>@

This command is identical to the Load command except that after the file is loaded it is CALLED at its execution address. If the program called does not change the stack ptr and returns properly, the monitor will be re-entered upon return.

LOAD INTEL HEX PAPER TAPE

H@

Reads characters from port 2 expecting to find a file in the Intel HEX format. If found, it is loaded at the specified address contained in the file itself. If a checksum error is detected in loading, a "C" will appear. If an

invalid character is detected, a "T" will appear and the operation will be terminated.

The input drivers are set up to read an MIO or SIO jumpered so the status appears on port 3 with bit 1 going high when data is available. The data should appear on port 2. This is the configuration which would be required if a teletype were being used for input.

LOAD CASSETTE FILE

L@
L<START>,<END>,<EXEC>@

This routine allows the user to load an object file (Type 81H) from cassette. Cassette files generated by IMSAI or by TCOS (Tape Cassette Operating System) may be loaded with this command.

All Type 81H files created by IMSAI or TCOS have a header record in front containing the file name (up to 5 characters), the starting memory address, the ending memory address, and an execution address which is often the same as the starting address. By typing in the L@ command, the user is effectively saying, "Load the next headered file into the memory address specified in the header."

By typing in the "L start, end, exec@" command, the user is saying, "load the next type 81 file whether headered or not using the start, end addresses specified in the command itself.

After the command is entered, the cassette recorder should be started. If a header is encountered, the name of the file to be loaded will be displayed. If no starting and ending addresses were specified, they will be read from the header. If an error is discovered in the header, an "I" initialization error will appear and the operation terminated.

All files created by IMSAI or TCOS block their records into 128 (80H) byte records. As the loader loads the object file, an indicator, "*", is displayed after each record is loaded to indicate a good load or a checksum error, "C", for that particular record.

After the entire file has been loaded, the starting address, ending address, and execution address of the file just loaded will be displayed. If at any point, an incorrect file type is encountered, a "T" for type code error will be displayed and the operation terminated.

CAUTION: DUE TO TIMING CONSIDERATIONS, THE VIO MUST NOT BE ALLOWED TO SCROLL DURING A LOADING FROM CASSETTE. THEREFORE THE SCREEN SHOULD BE ERASED (CTRL-Z) BEFORE KEYING IN THE LOAD COMMAND.

BOOT FROM FLOPPY DISK

B@

READ FROM DISKETTE

R<tt,ss,bbbb,u>@

where:

tt = track
ss = sector
bbbb = buffer address
u = unit

Track and sector must be specified; buffer and unit specifications are optional.

If the information contained within the bracket is not specified, the following values are assumed:

track = 00
sector = 01
buffer address = 0000
unit #(master) = 0

WRITE TO DISKETTE

W@
W<tt,ss,bbbb,u>

JUMP AND SWITCH-OUT MPU-B

K<x>@

Jump to locattion x and switch off MPU-B ROM and system memory

MPU-B ROM residing at D800 - DBFF
MPU-B RAM residing at D001 - D0FF

SWITCH OUT MPU-B AND JUMP TO VIO MONITOR
(INITIALIZE SCREEN)

Q@

SET BAUD RATE

Z@ (requires two terminals)
Zxxxx (where xxxx is baud rate)

Sets baud rate for MPU-B serial port as 12 and 13 and 4 and **5** (the system terminal is tied to 2 and 3).

When Z@ is typed, the system will print:

HIT SPACE BAR

The user then hits the space bar on the terminal tied to 12 and 13. the system will print out:

xxxx BAUD SERIAL PORT

3. VIO OPERATION WITHOUT VIOROM

When the VIOROM is not used, the VIO functions and modes of operation may be controlled directly from user programs.

It should be noted that operation without the VIOROM is intended primarily for those Users who wish to write a display driver for a particular application, or for those Users who wish to use only very minimal display functions. For most video display applications, it is recommended that the VIOROM be used.

a) VIO Initialization

On power-up or reset, the VIO will default to the following mode of operation:

screen blanked (mode 0)
80x24 screen format

It is then up to the user's program to 1) initialize the VIO's refresh memory for the initial display desired (Part b); and 2) to select the desired VIO mode of operation using the appropriate command byte (Part c).

b) Character Display

To display a character on the screen, the user's program must store the character code for the character to be displayed into the VIO's refresh memory.

Since the VIO's refresh memory consists of 1K or 2K of RAM residing in the system memory space, the user's program may store a character code into the refresh memory using any memory store instruction (e.g., MOV M,R). It is not possible to write into the refresh memory using a front panel Deposit or DMA.

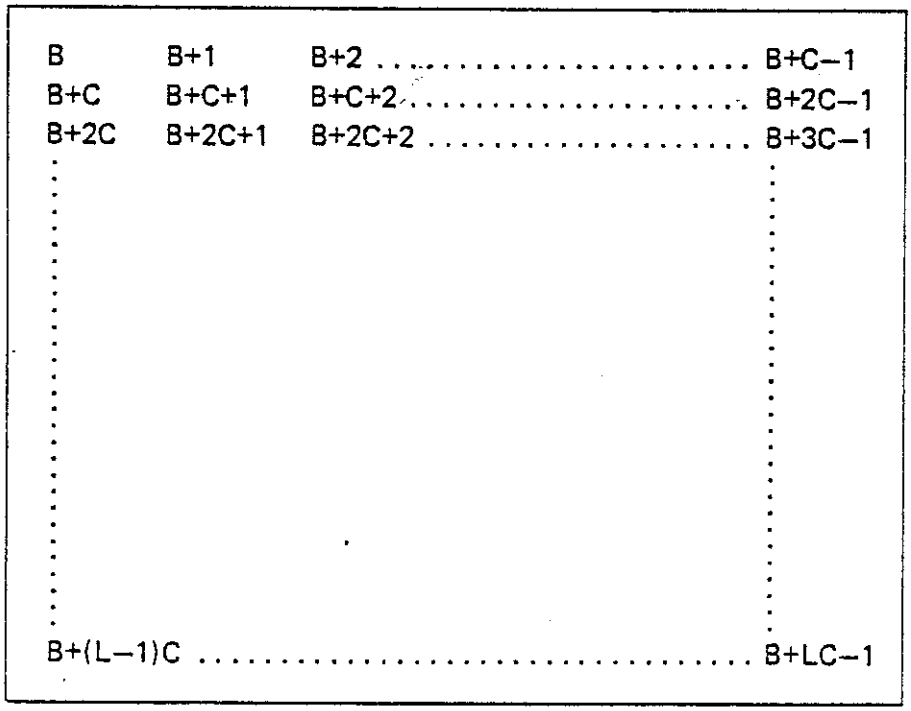
Note that the address of the VIO's refresh memory was set by the user in Section III-B,1, Steps 1 and 2. The position of the displayed character, on the screen, is a function of the location (in the refresh memory) where the character code is stored.

The first memory location in the VIO's refresh memory corresponds to the first display position in the upper left corner of the screen. Each succeeding memory location will correspond to the next display position. Characters are displayed by row, from left to right, with the last display position in the lower right corner of the screen. The relationship between memory locations is shown in Figure III-20.

For example, if the refresh memory were jumpered to reside at F000H (standard address), storing a 31H (ASCII "1") in memory location F000H would cause the character "1" to appear in the upper left hand corner of the screen.

Similarly, a 31H stored in location F028H would cause the character "1" to appear as the 41st character of the first line (if the 80 column option is selected) or as the first character of the second line (if the 40 column option is selected).

A list of the valid character codes appears in Appendix 1. The characters which may be



B=BASE ADDRESS OF REFRESH MEMORY F000H = 61440
C=NUMBER OF COLUMNS PER LINE
L=NUMBER OF LINES PER PAGE

FIGURE III-20 RELATIONSHIP BETWEEN DISPLAY POSITION AND REFRESH MEMORY LOCATION

displayed on the screen depend entirely upon the mode of operation selected by the user: Mode 0, Mode 1, Mode 2, or Mode 3. These modes are selected with the appropriate command byte described in Part c.

MODE 0

When MODE 0 is selected, the screen will be blanked and no characters will be displayed. Display of the characters stored in the VIO's refresh memory will not begin until MODE 0 is terminated.

MODE 1

When MODE 1 is selected, any of the 128 graphic characters represented by character codes 80-FF H (Appendix 1) may be displayed. In MODE 1 bit 7 of the character code acts as the reverse video flag (0=positive video, 1=reverse video).

Note that the character codes listed in Appendix 1 are 8 bits wide. In MODE 1, only the lower 7 bits will be used as the actual character code. The high order bit is used as the reverse video flag.

MODE 2

When MODE 2 is selected, any of the 128 characters represented by the character codes 00-7FH (Appendix 1) may be displayed. This includes the standard 96 character ASCII set and 32 graphic characters. In MODE 2, bit 7 of the character code will act as the reverse video flag (0 = positive video, 1 = reverse video).

For example, a 31H (ASCII "1") stored into the refresh memory would cause the character "1" to be displayed in positive video (white character on dark background). The same code, with bit 7 on, would cause the character to be displayed in reverse video. Thus a B1H would cause the character "1" to be displayed as a dark character on a light background.

Note that the character codes listed in Appendix 1 are 8 bits wide. In MODE 2, only the lower 7 bits will be used as the actual character code. The high order bit is used as the reverse video flag.

MODE 3

When MODE 3 is selected, any of the 256 characters represented by character codes 00-FFH (Appendix 1) may be displayed. Note that in MODE 3, character-by-character video is not available. All 8 bits of the character codes listed in Appendix 1 will be used as the actual character codes.

c) Command Byte

The top location in the VIO's refresh memory serves as a memory mapped command port. Note that the address of this port is dependent upon the location of the refresh memory (set in Section III-B,1, Steps 1 and 2).

If the refresh memory was set to reside in the top half of the VIO's 4K address space, (Section III-B,1, Step 2), the command port address will be of the form XFFF, where X is equal to the high order address determined in Section III-B,1, Step 1. If the refresh memory was set to reside in the lower half of the VIO's 4K address space, the command port address will be of the form X7FF, where X is equal to the high order address determined in Section III-B,1, Step 1.

The command port address is not affected by the amount of refresh memory (1K or 2K) installed.

The command byte which is stored in the command port location will determine line length, page length, full screen reverse video, and mode of operation, according to the following format:

Bit 0 when equal to 0, produces 80 character lines
when equal to 1, produces 40 character lines

Bit 1 when equal to 0, produces 24 line pages;
when equal to 1, produces 12 line pages.

Bits 2-3 are encoded and control the VIO mode of operation
When equal to 00, activates Mode 0,
video off, screen blanked.

When equal to 01, activates Mode 1. Character codes 80-FF (Appendix 1) may be displayed. Bits 0-6 of codes 80-FF are used as the character code. Bit 7 is used as the character-by-character reverse video flag (0 = positive video, 1 = reverse video).

When equal to 10, activates Mode 2. Character codes 00-7F H (Appendix 1) may be displayed. Bits 0-6 of codes 00-7F H are used as the character code. Bit 7 is used as the character-by-character reverse video flag (0=positive video, 1=reverse video).

When equal to 11, activates Mode 3. Character codes 00-FF (Appendix 1) may be displayed. Character-by-character reverse video is not available in this mode, and all 8 bits of codes 00-FF are used as character code.

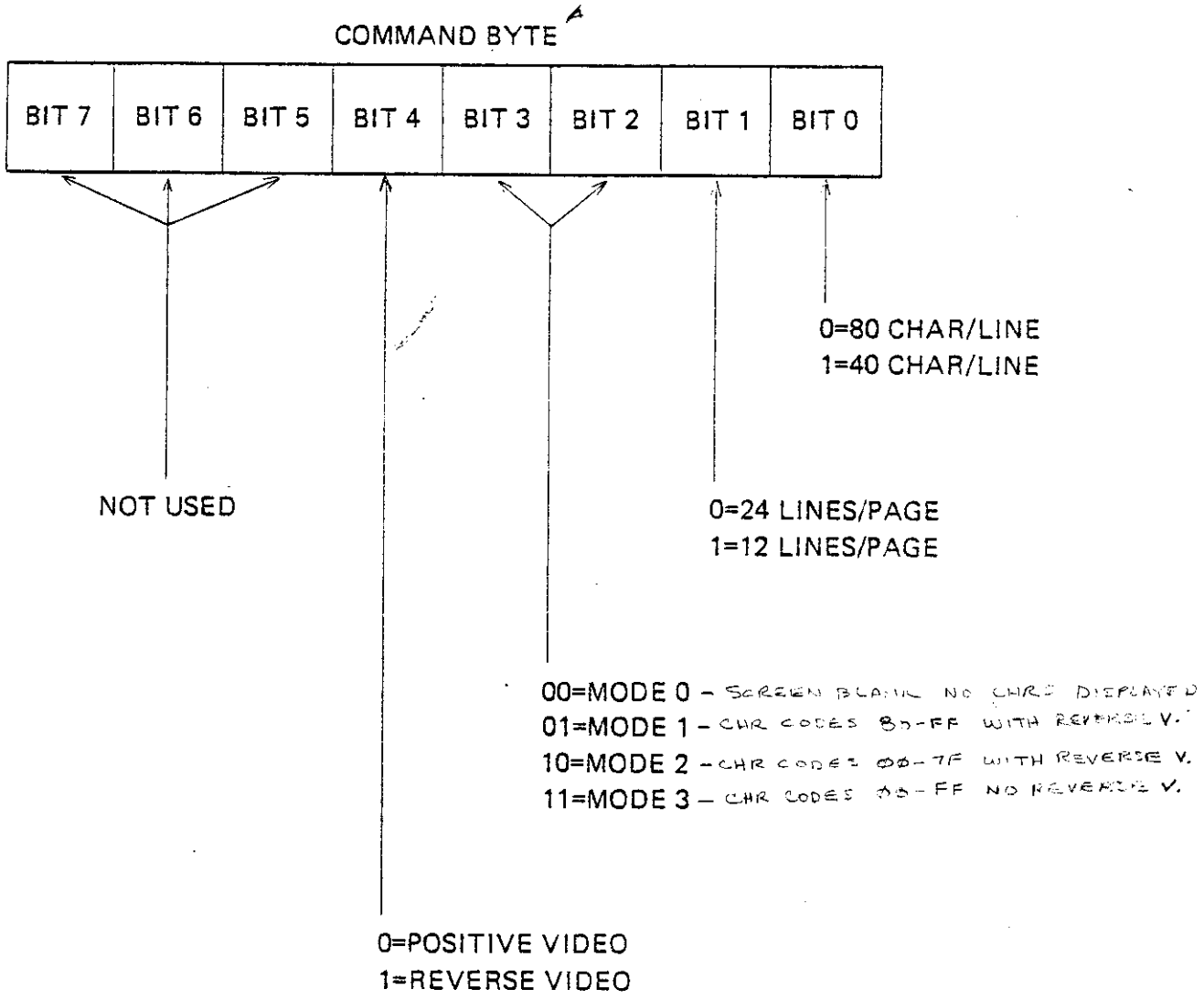


FIGURE III-21 COMMAND BYTE FORMAT

Bit 4 when equal to 0, causes the screen display to appear as positive video (light characters on a dark background).

When equal to 1, causes the screen display to appear as reverse video (dark characters on a light background).

Bits 5-7 are not used.

The command byte format is shown in Figure III-21. On power-up or reset, the command byte will default to 00H (Mode 0) with the screen blanked. The command byte may then be changed using any memory store instruction (e.g., MOV M,R). If the VIO contains a full 2K bytes of refresh memory, the command byte may also be read using any memory read instruction (e.g., MOV R,M). It is not possible to write into the Command Port location using a front panel Deposit or DMA.

NOTE: Reading the command port location after a reset, but prior to writing, will not reflect the default command (00).

For example, if the VIO's refresh memory is set to reside in location F000-F7FF (bottom half of 4K block beginning at F000), we may store the command byte 07H in location F7FF to produce

positive video display
Mode 1
12 lines per page
40 characters per line

Similarly, if the VIO's refresh memory is set to reside in location F800-FFFF (top half of 4K block beginning at F000H), we may store the command byte 08H in location FFFF to produce

positive video display
Mode 2
24 lines per page
80 characters per line

C. APPENDICES

Appendix 1 — Character Codes

Appendix 2 — X, Y Position Codes

Appendix 3 — Simple VIO Driver Listing

Appendix 4 — Programming the Character Generator ROMs/PROMs

Appendix 5 — Using CP/M with the IMSAI VIO Video Display Board

APPENDIX 1

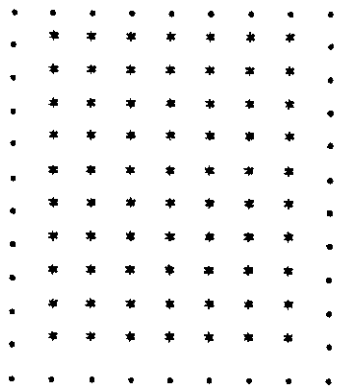
The characters represented by the character codes 00-FF H, comprise the complete 256 character set of the VIO.

Characters which may be displayed depend upon the display mode selected. Codes 20-7F H may be displayed through the firmware, in the ASCII Text Mode. Codes A0-FF H may be displayed through the firmware in the Extended Text Mode. Codes 00-FF H may be displayed through the firmware in the Graphic Mode (with the exception of the character 1B H).

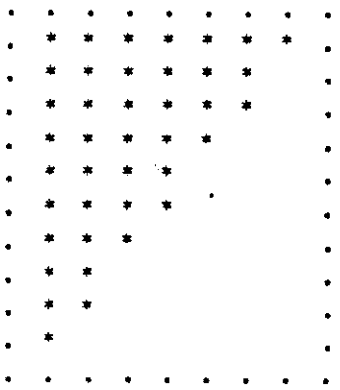
Codes 00-FF H may also be displayed by directly accessing the VIO's refresh memory.

The valid character codes 00-FF are listed in the following pages.

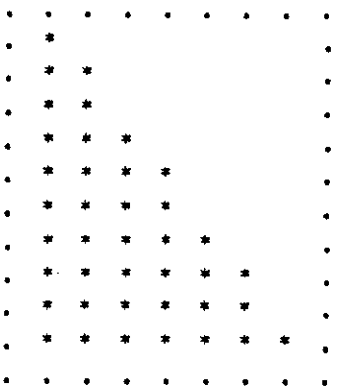
CODE: 00 H



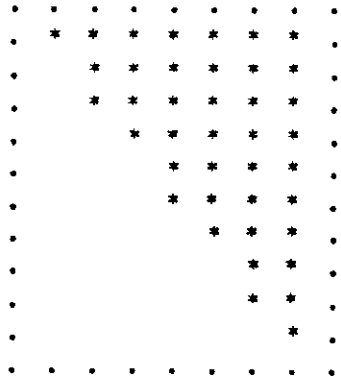
CODE 01 H



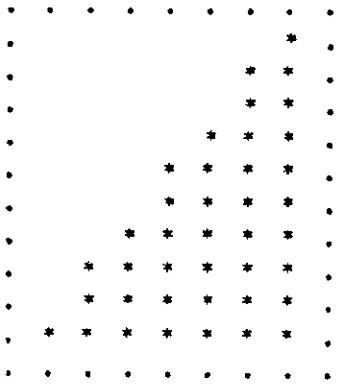
CODE 02 H



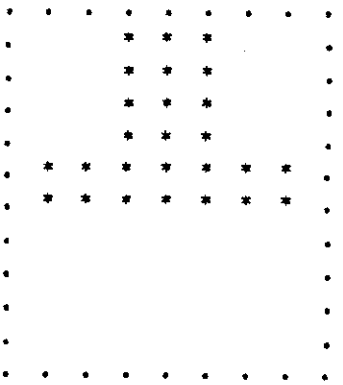
CODE 03 H



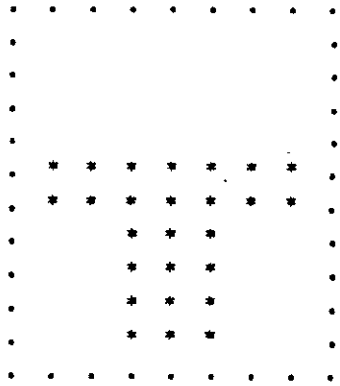
CODE 04 H



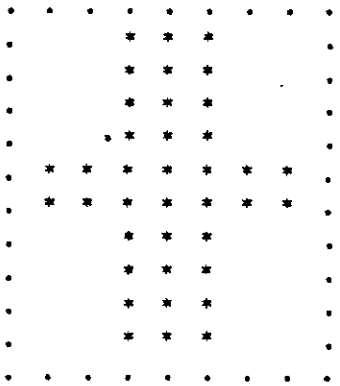
CODE 05 H



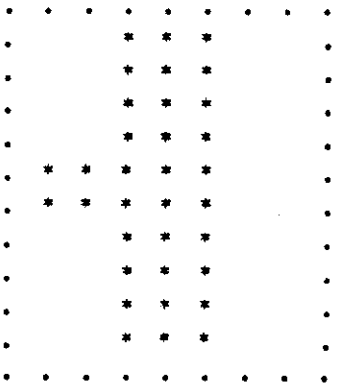
CODE 06 H



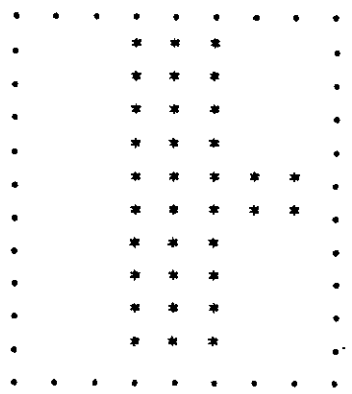
CODE 07 H



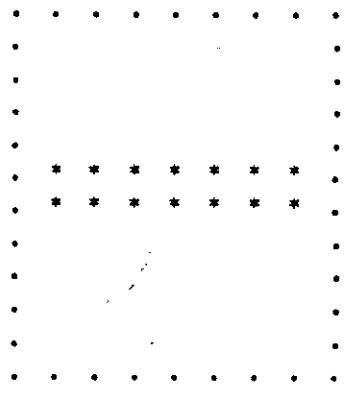
CODE 08 H



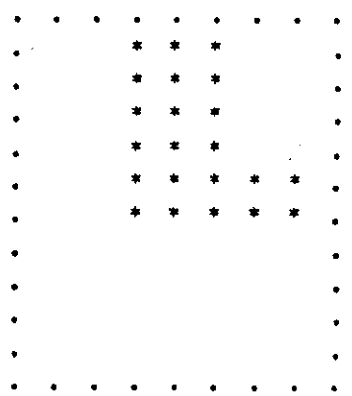
CODE 09 H



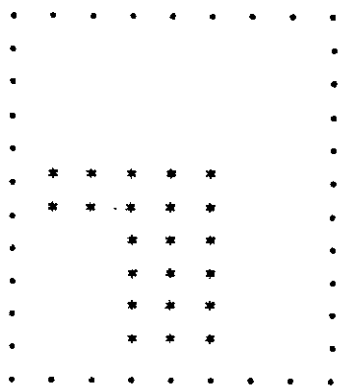
CODE 0C H



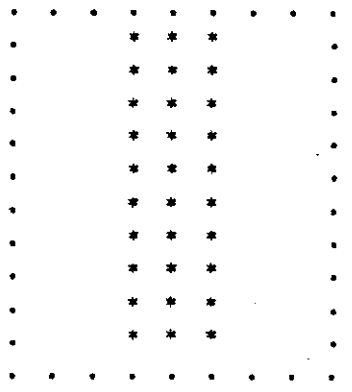
CODE 0F H



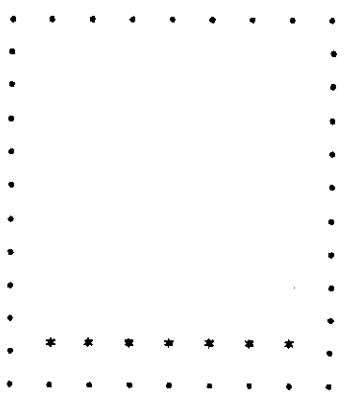
CODE 0A H



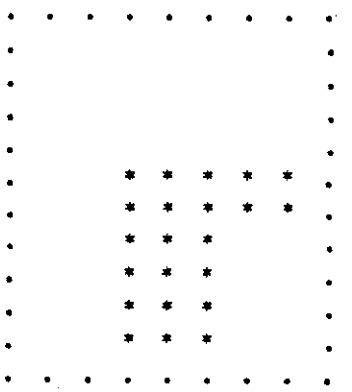
CODE 0D H



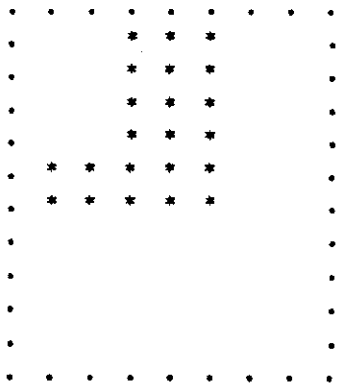
CODE 10 H



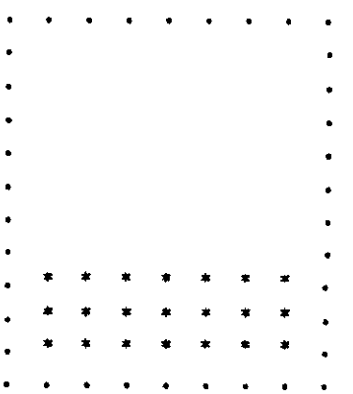
CODE 0B H



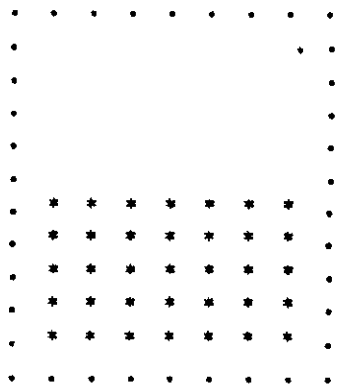
CODE 0E H



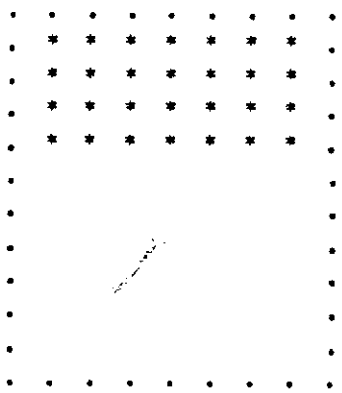
CODE 11 H



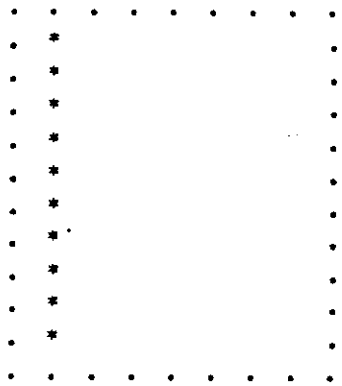
CODE 12 H



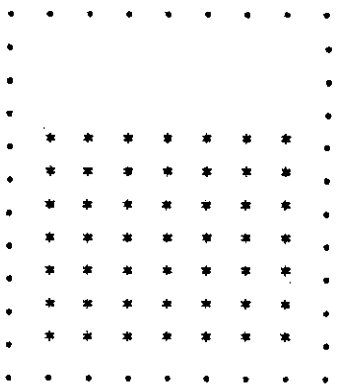
CODE 15 H



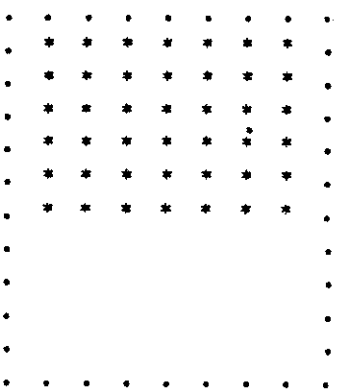
CODE 18 H



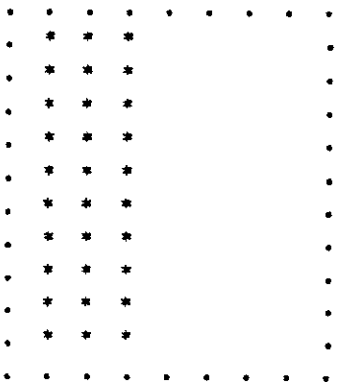
CODE 13 H



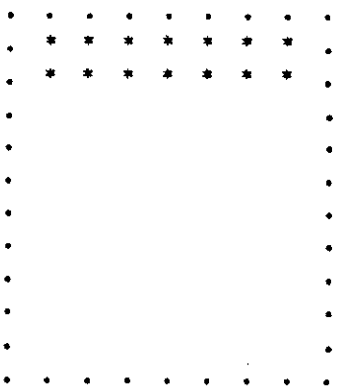
CODE 16 H



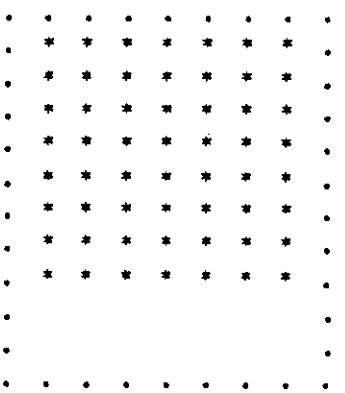
CODE 19 H



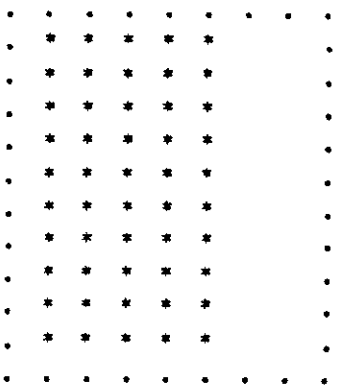
CODE 14 H



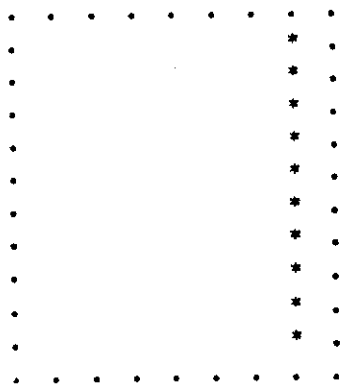
CODE 17 H



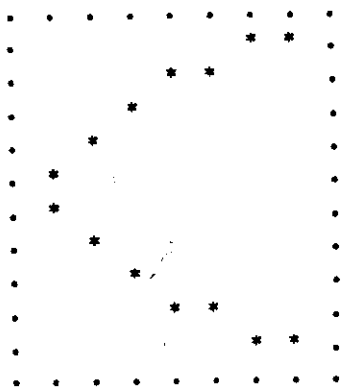
CODE 1A H



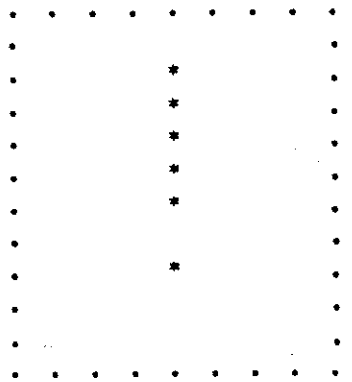
CODE 1B H



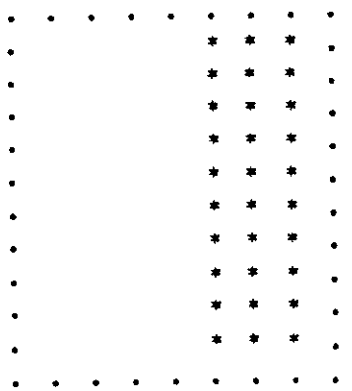
CODE 1E H



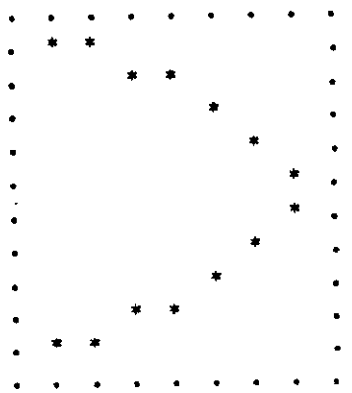
CODE 21 H



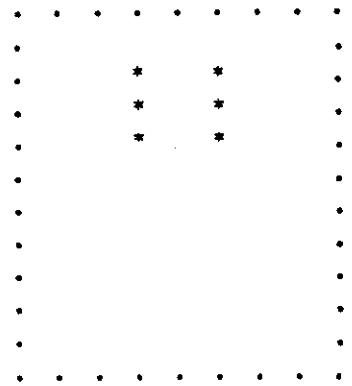
CODE 1C H



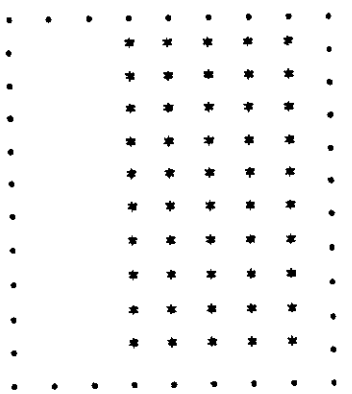
CODE 1F H



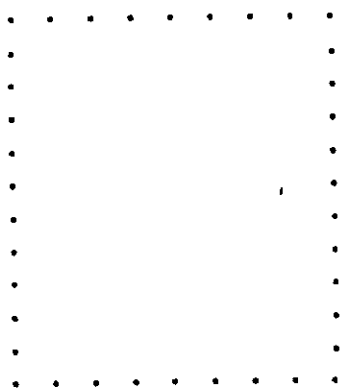
CODE 22 H



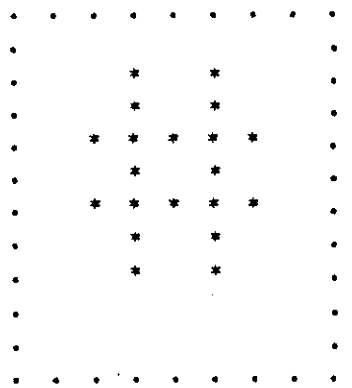
CODE 1D H



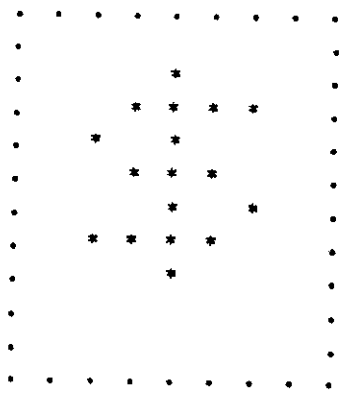
CODE 20 H



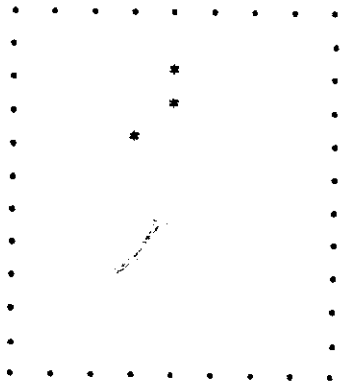
CODE 23 H



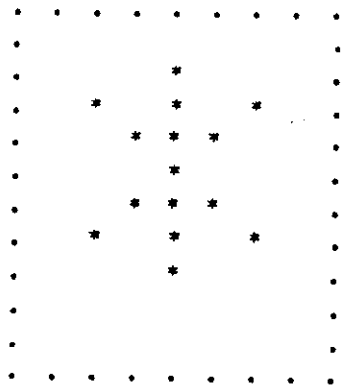
CODE 24 H



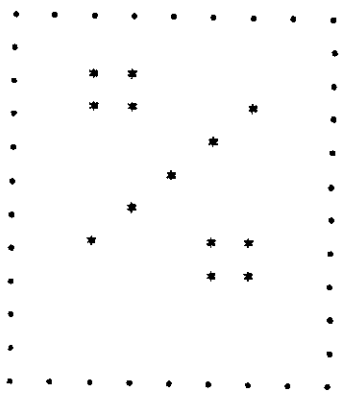
CODE 27 H



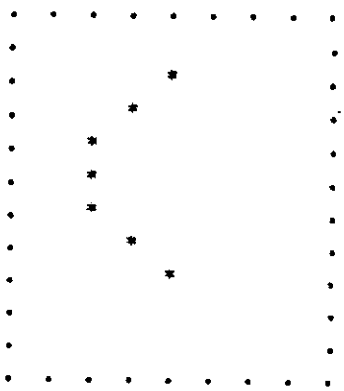
CODE 2A H



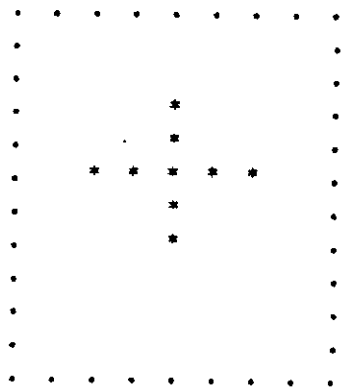
CODE 25 H



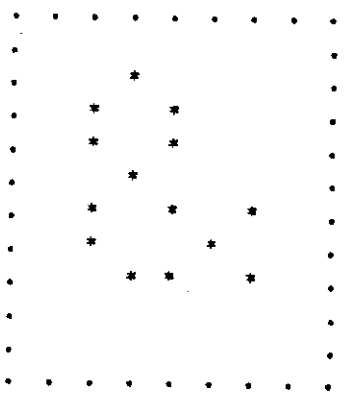
CODE 28 H



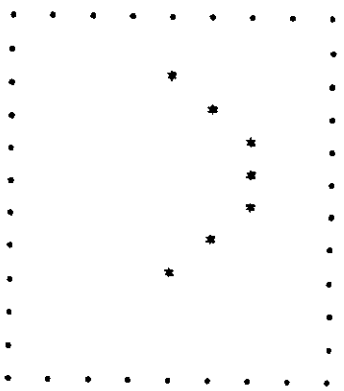
CODE 2B H



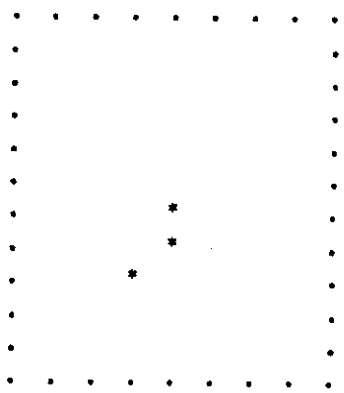
CODE 26 H



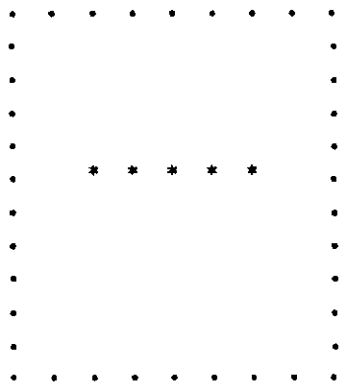
CODE 29 H



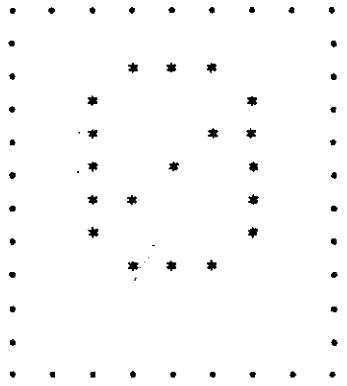
CODE 2C H



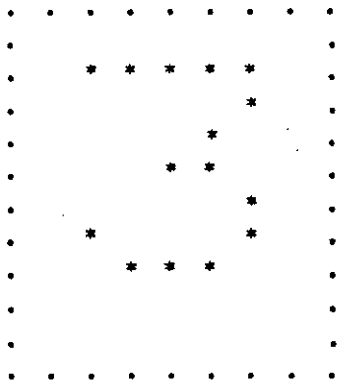
CODE 2D H



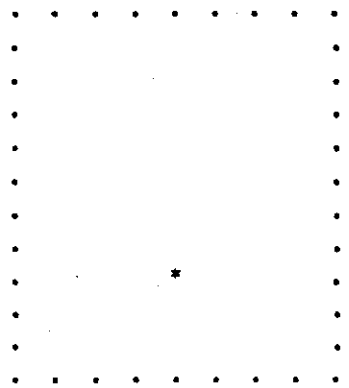
CODE 30 H



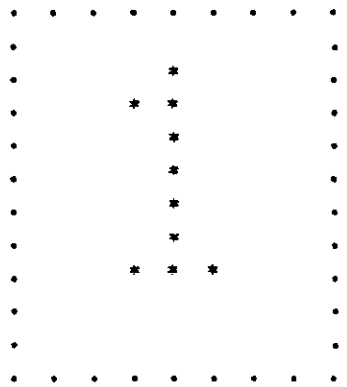
CODE 33 H



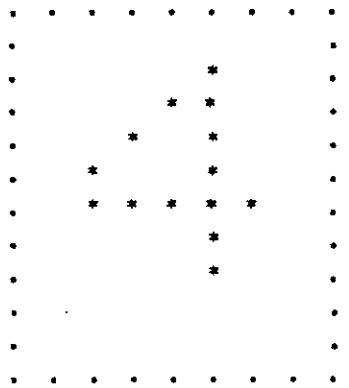
CODE 2E H



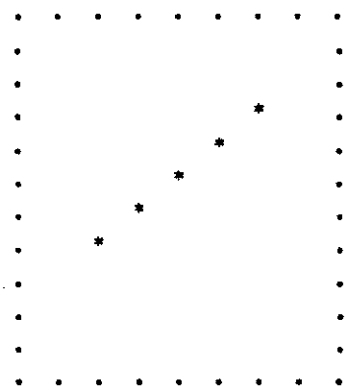
CODE 31 H



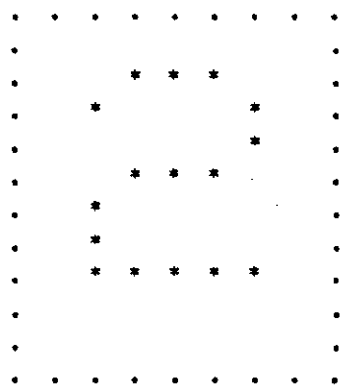
CODE 34 H



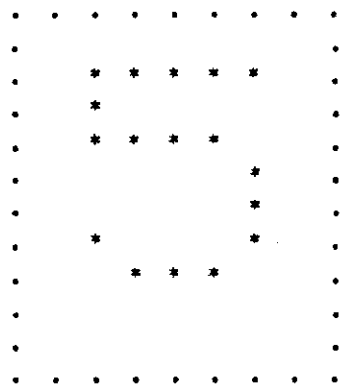
CODE 2F H



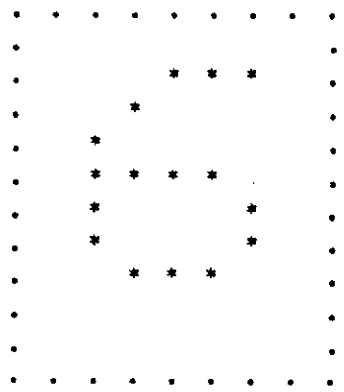
CODE 32 H



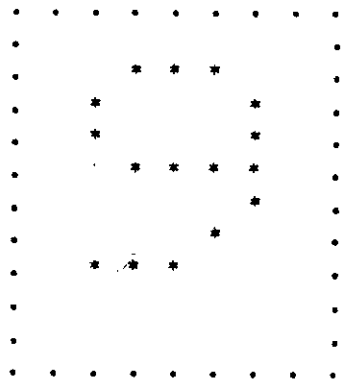
CODE 35 H



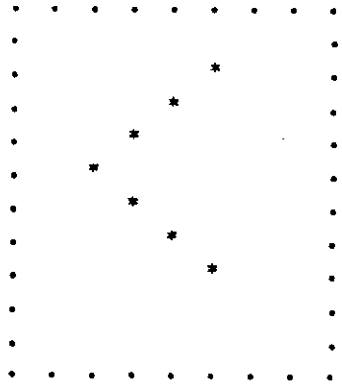
CODE 36 H



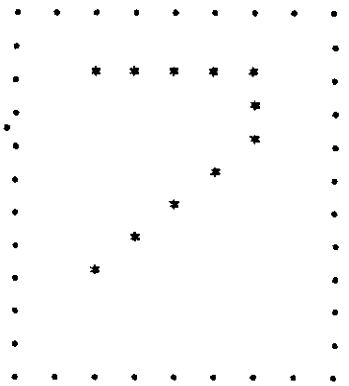
CODE 39 H



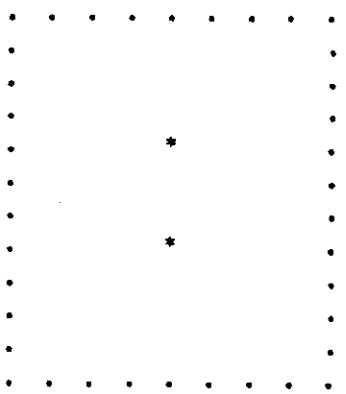
CODE 3C H



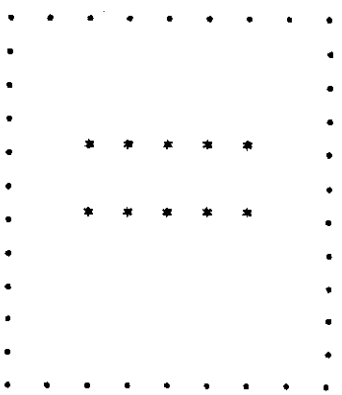
CODE 37 H



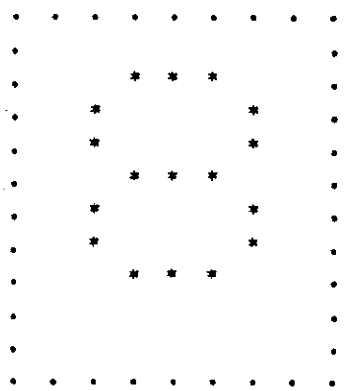
CODE 3A H



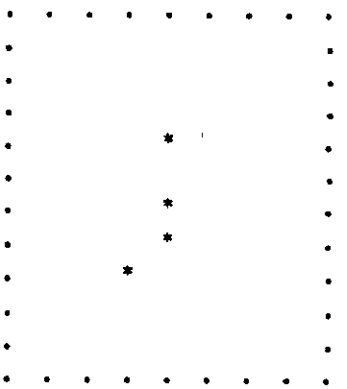
CODE 3D H



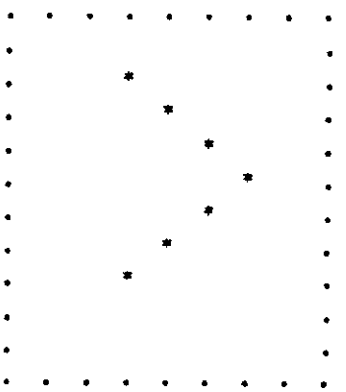
CODE 38 H



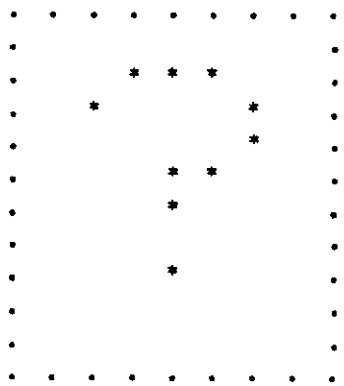
CODE 3B H



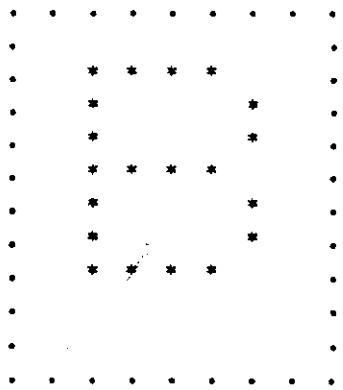
CODE 3E H



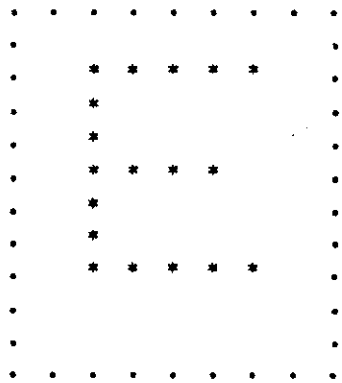
CODE 3F H



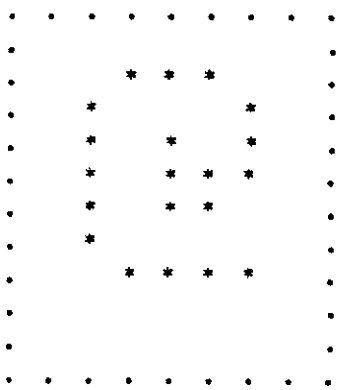
CODE 42 H



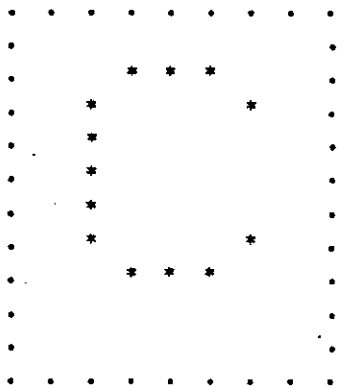
CODE 45 H



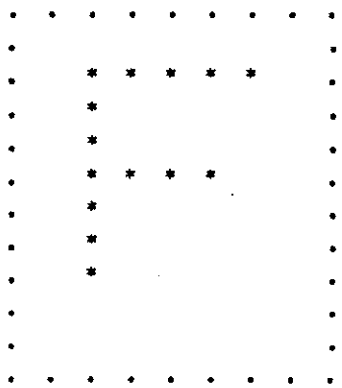
CODE 40 H



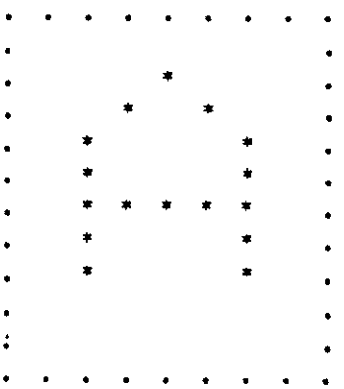
CODE 43 H



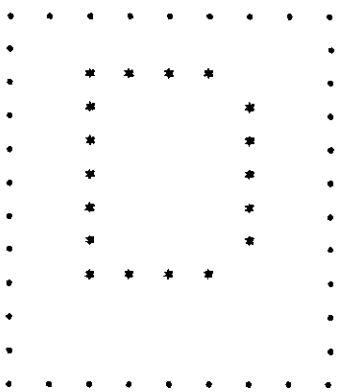
CODE 46 H



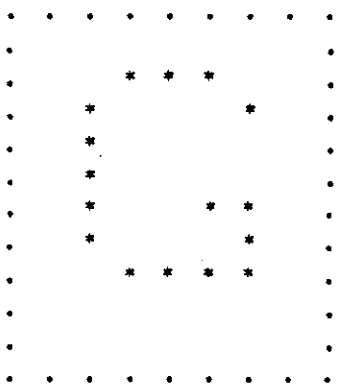
CODE 41 H



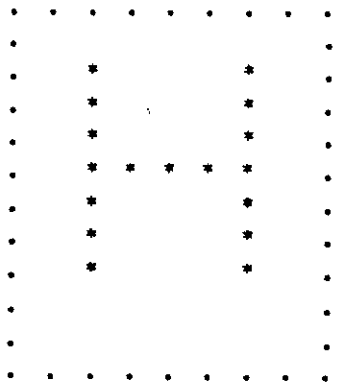
CODE 44 H



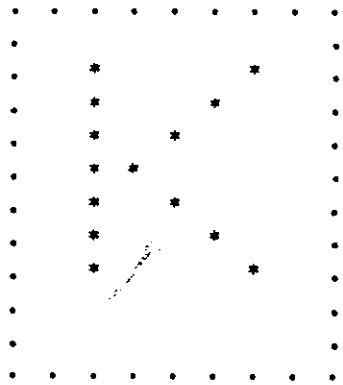
CODE 47 H



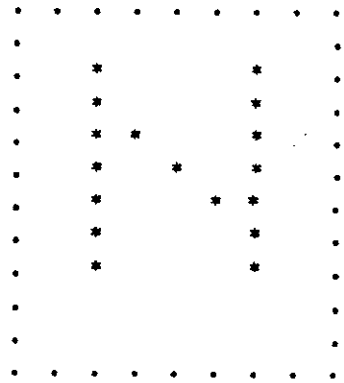
CODE 48 H



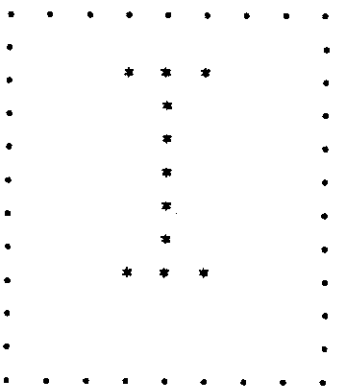
CODE 4B H



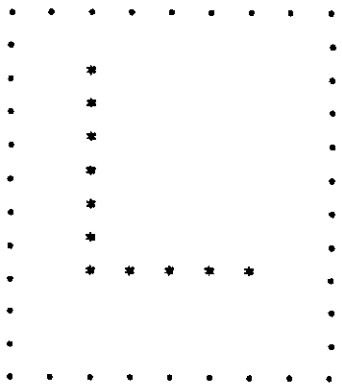
CODE 4E H



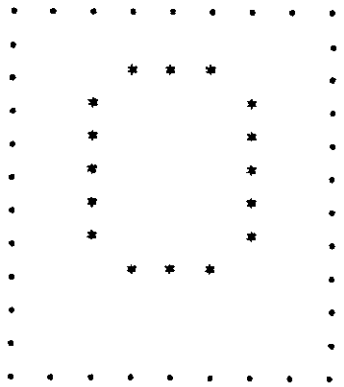
CODE 49 H



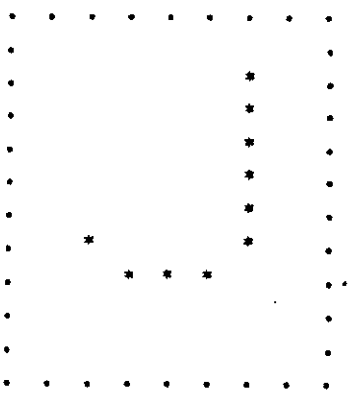
CODE 4C H



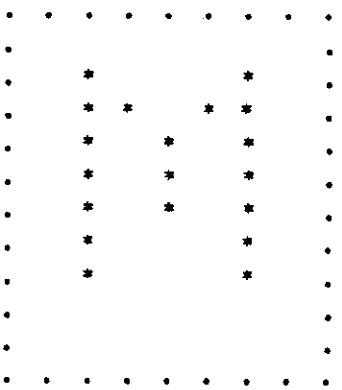
CODE 4F H



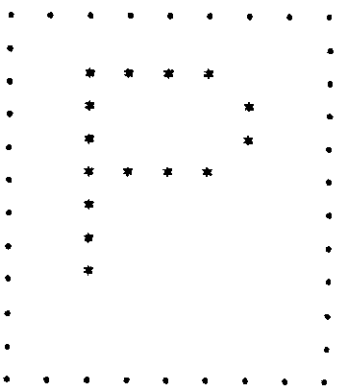
CODE 4A H



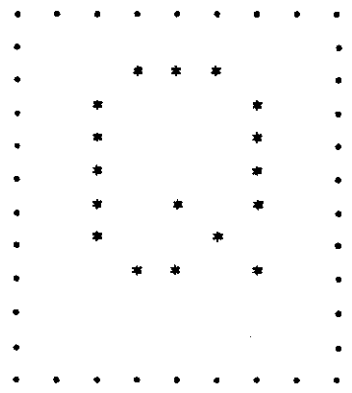
CODE 4D H



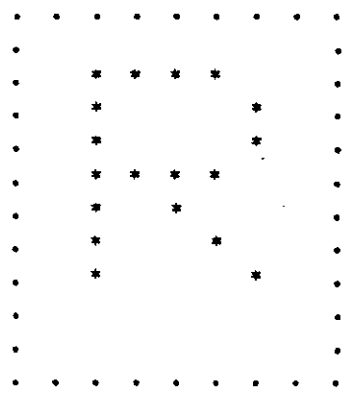
CODE 50 H



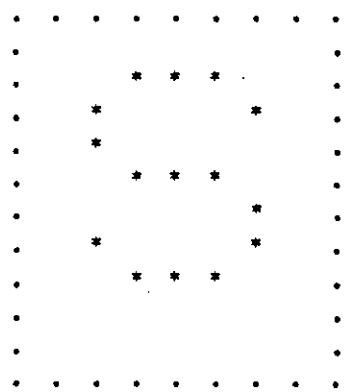
CODE 51 H



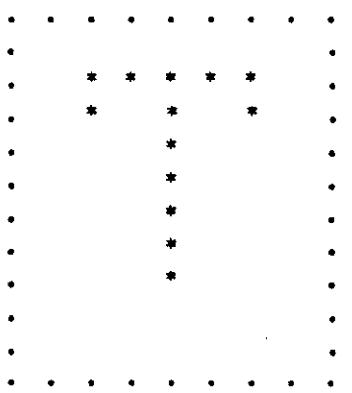
CODE 52 H



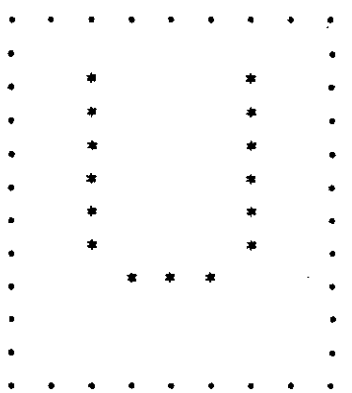
CODE 53 H



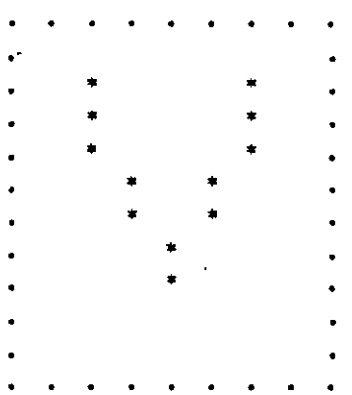
CODE 54 H



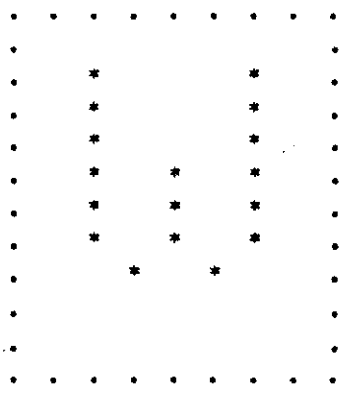
CODE 55 H



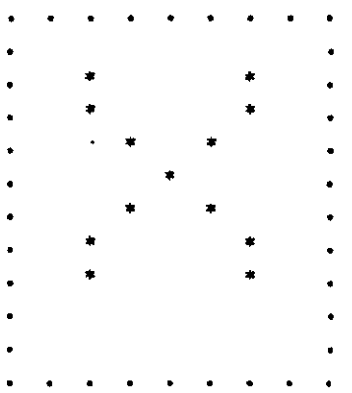
CODE 56 H



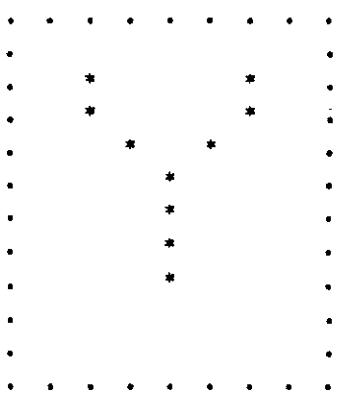
CODE 57 H



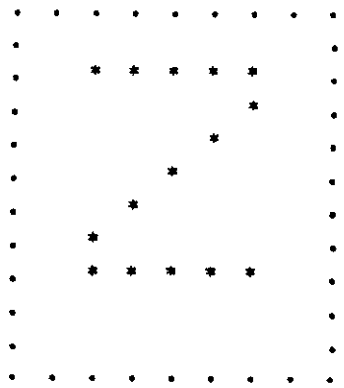
CODE 58 H



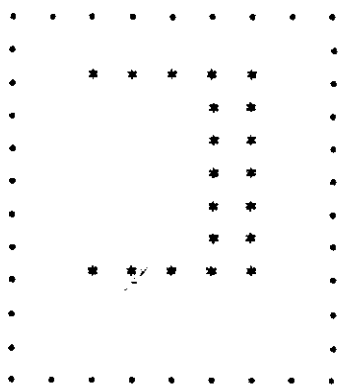
CODE 59 H



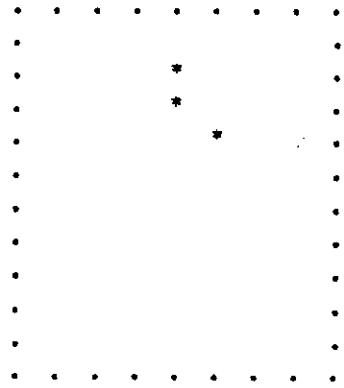
CODE 5A H



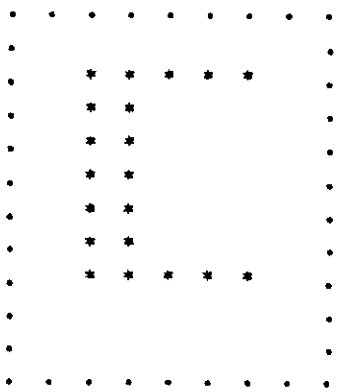
CODE 5D H



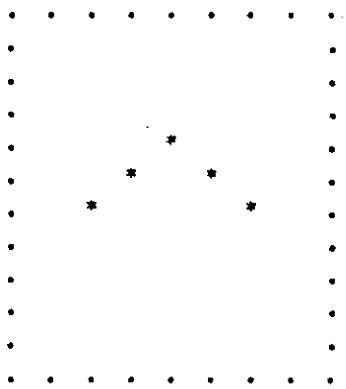
CODE 60 H



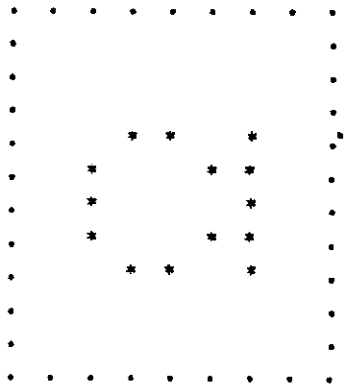
CODE 5B H



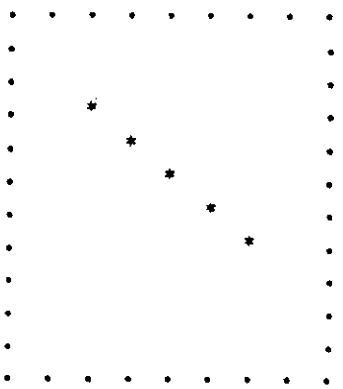
CODE 5E H



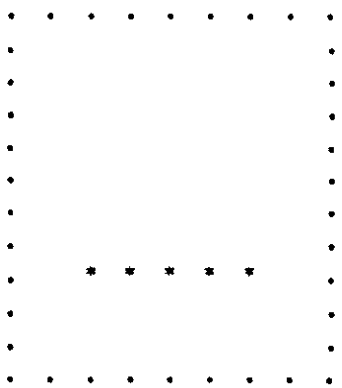
CODE 61 H



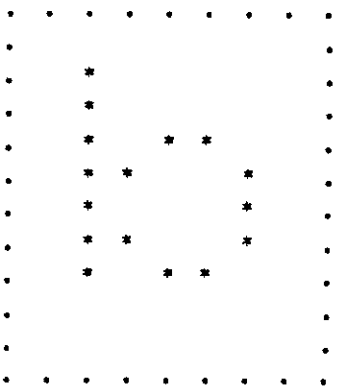
CODE 5C H



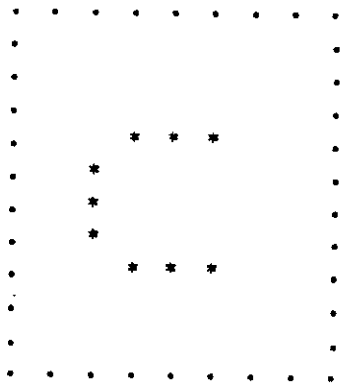
CODE 5F H



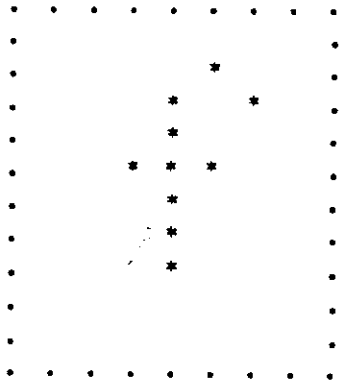
CODE 62 H



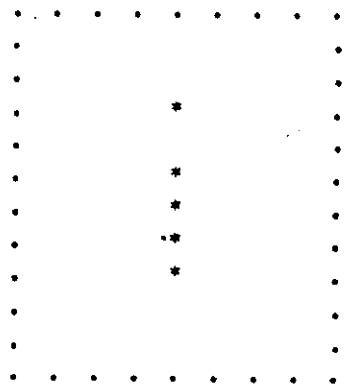
CODE 63 H



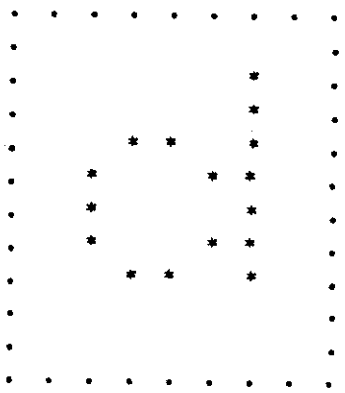
CODE 66 H



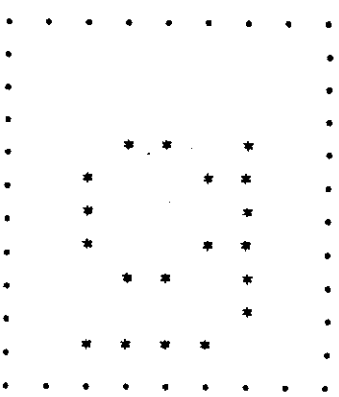
CODE 69 H



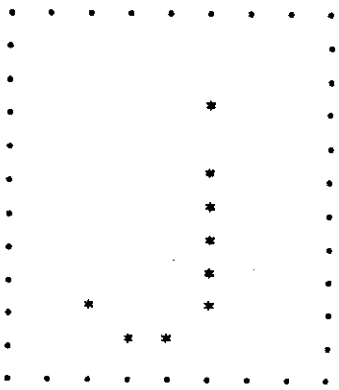
CODE 64 H



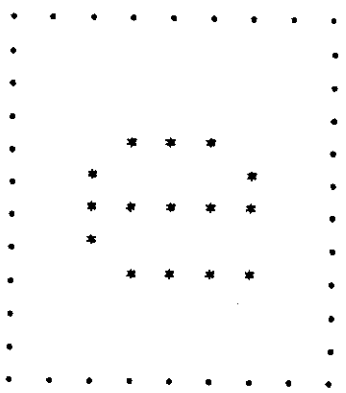
CODE 67 H



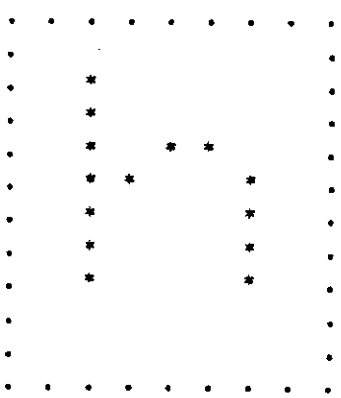
CODE 6A H



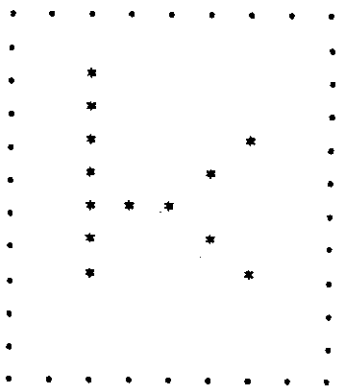
CODE 65 H



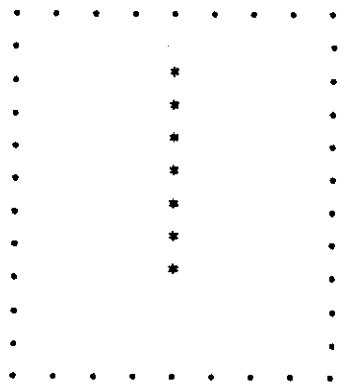
CODE 68 H



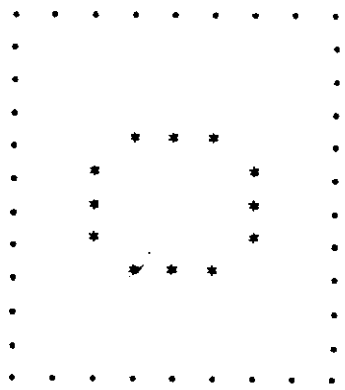
CODE 6B H



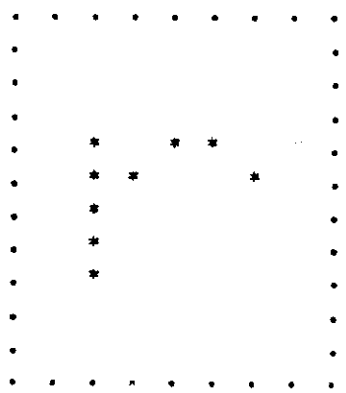
CODE 6C H



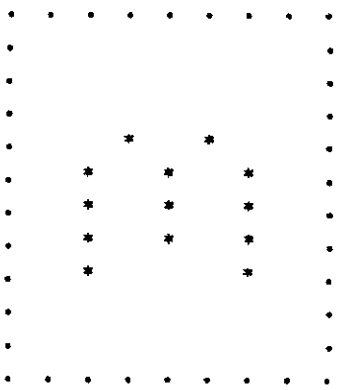
CODE 6F H



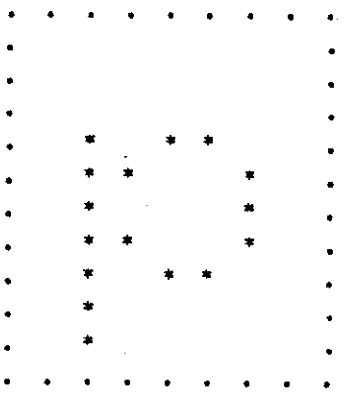
CODE 72 H



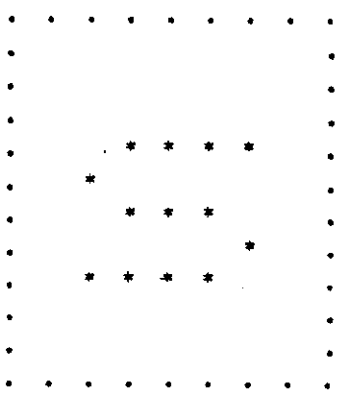
CODE 6D H



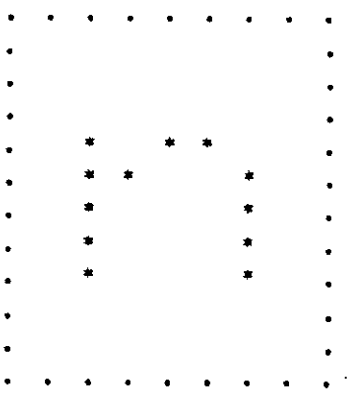
CODE 70 H



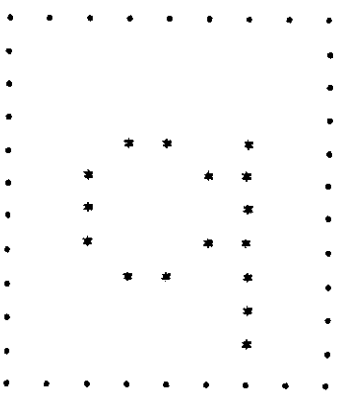
CODE 73 H



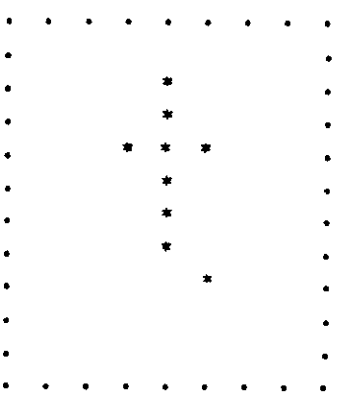
CODE 6E H



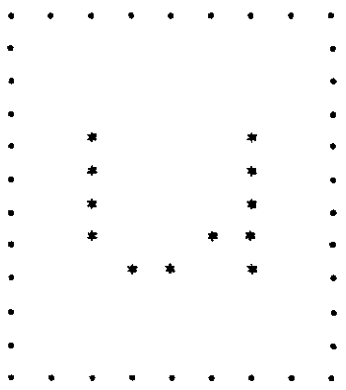
CODE 71 H



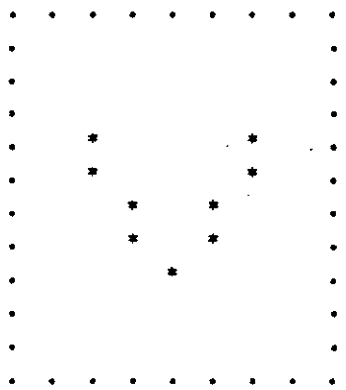
CODE 74 H



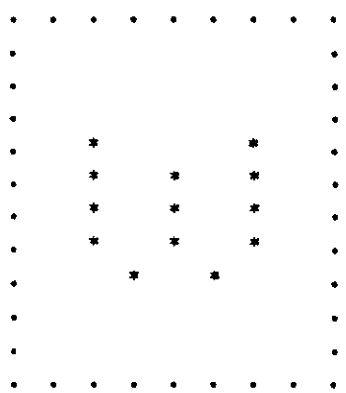
CODE 75 H



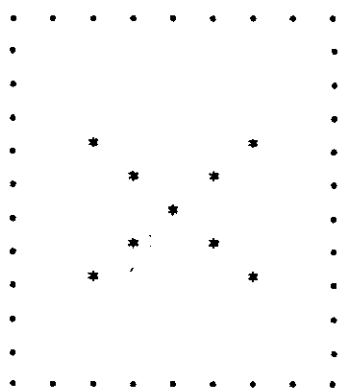
CODE 76 H



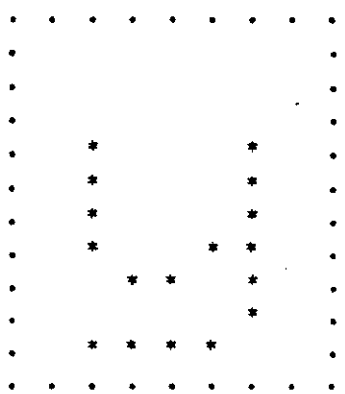
CODE 77 H



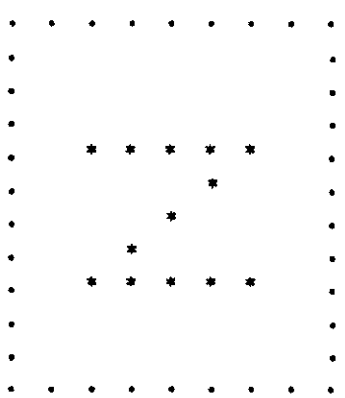
CODE 78 H



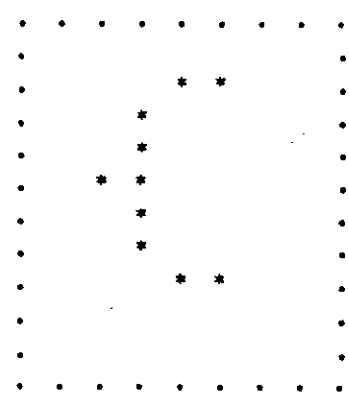
CODE 79 H



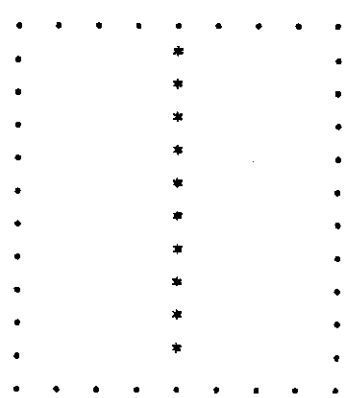
CODE 7A H



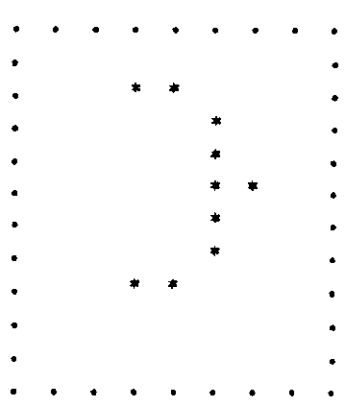
CODE 7B H



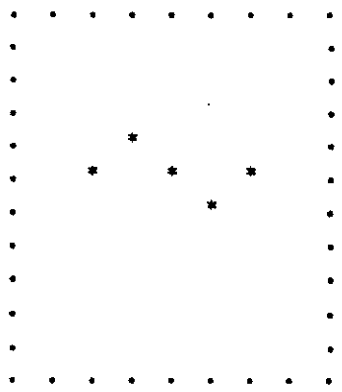
CODE 7C H



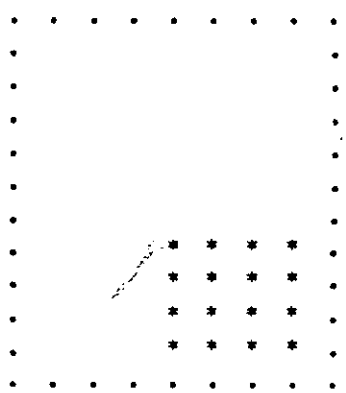
CODE 7D H



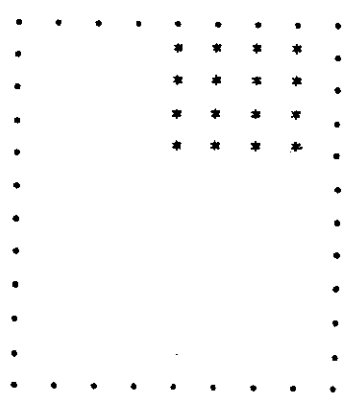
CODE 7E H



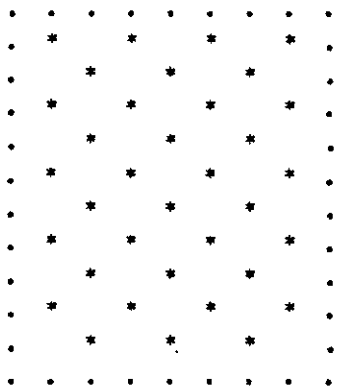
CODE 81 H



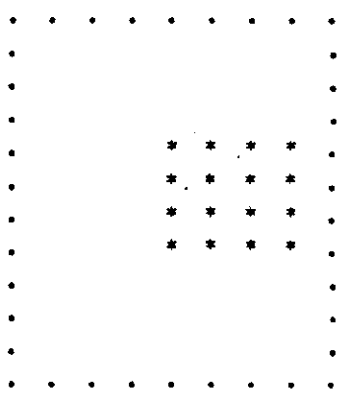
CODE 84 H



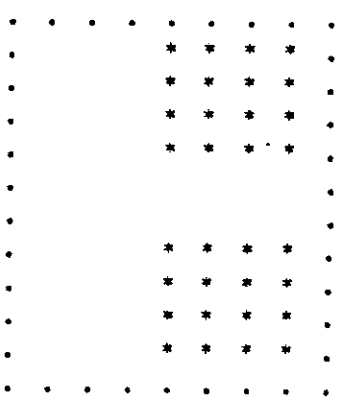
CODE 7F H



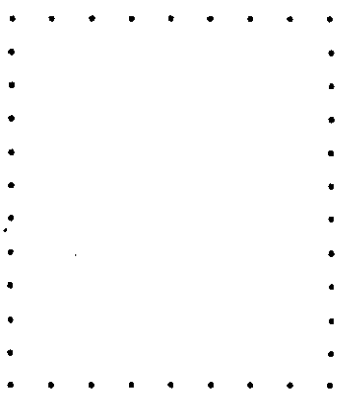
CODE 82 H



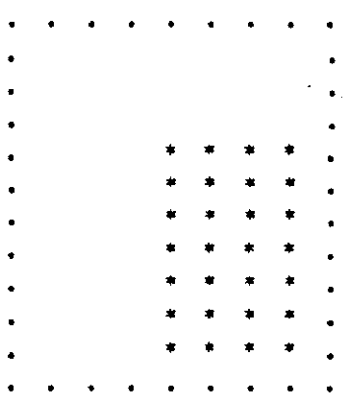
CODE 85 H



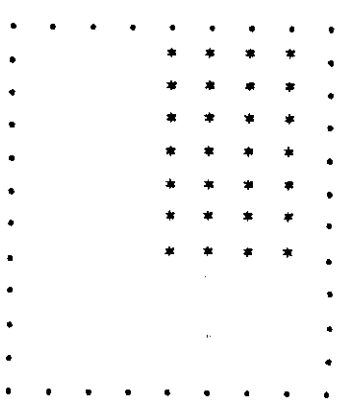
CODE 80 H



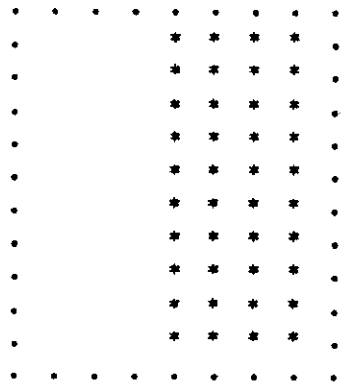
CODE 83 H



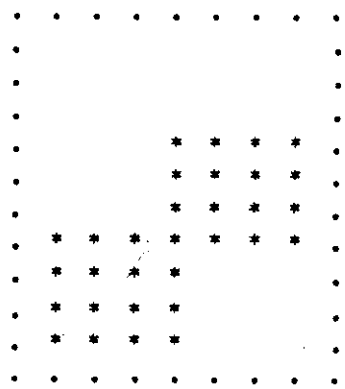
CODE 86 H



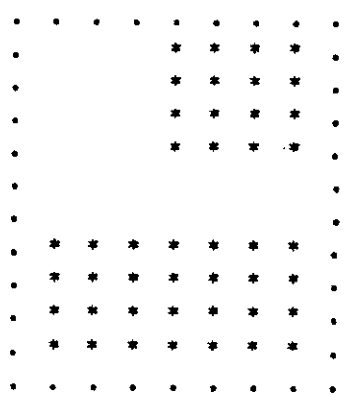
CODE 87 H



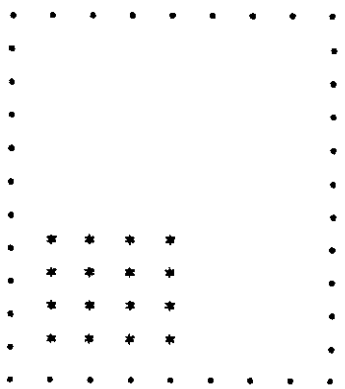
CODE 8A H



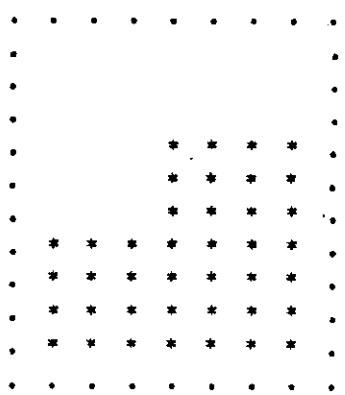
CODE 8D H



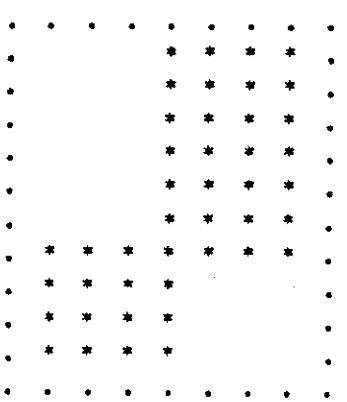
CODE 88 H



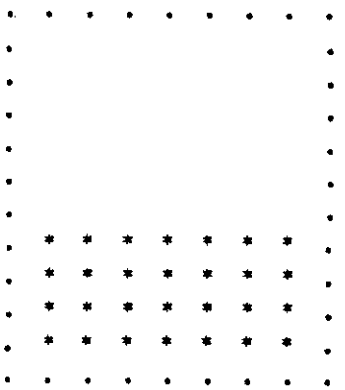
CODE 8B H



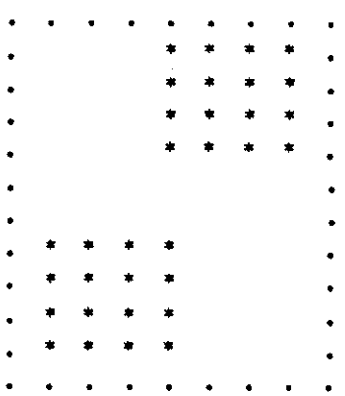
CODE 8E H



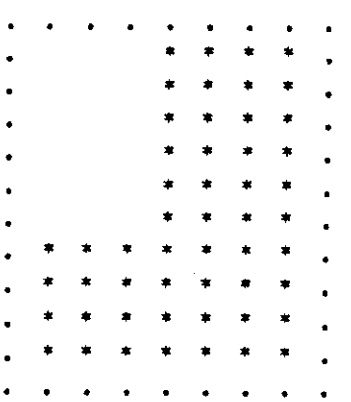
CODE 89 H



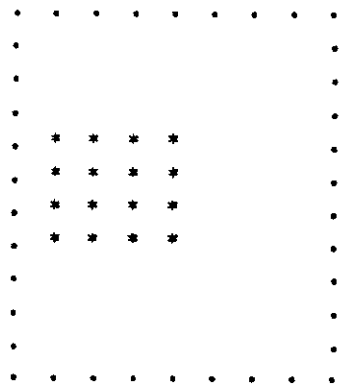
CODE 8C H



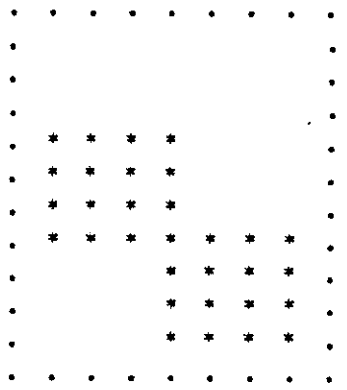
CODE 8F H



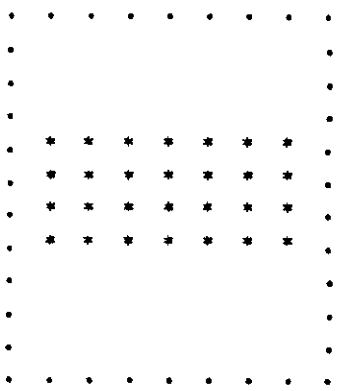
CODE 90 H



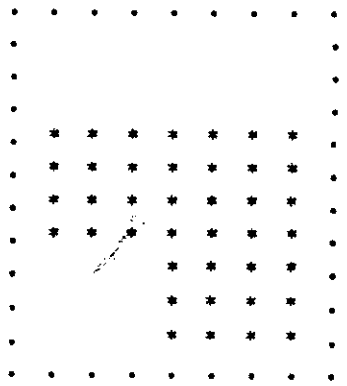
CODE 91 H



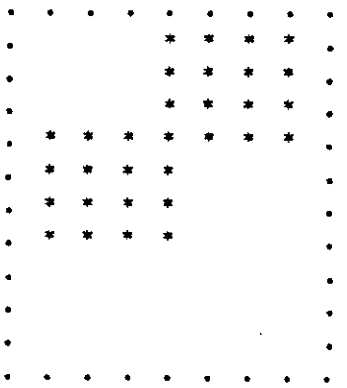
CODE 92 H



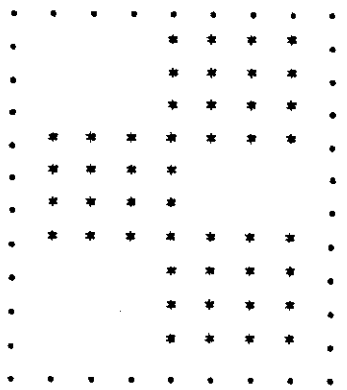
CODE 93 H



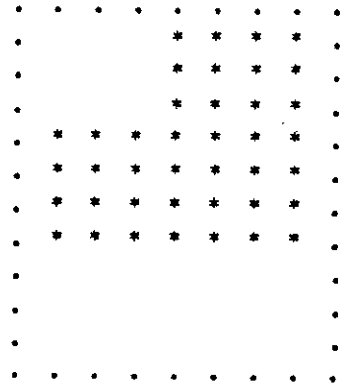
CODE 94 H



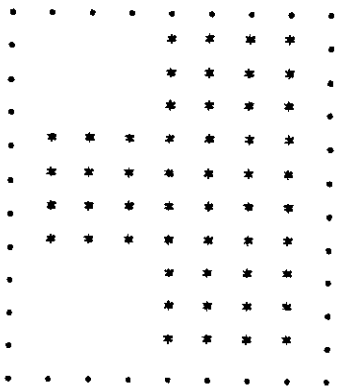
CODE 95 H



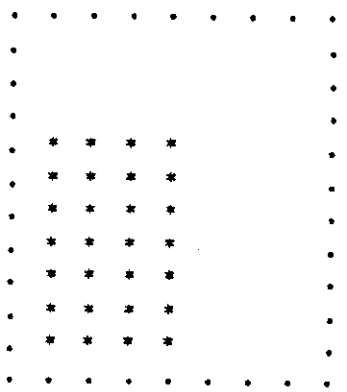
CODE 96 H



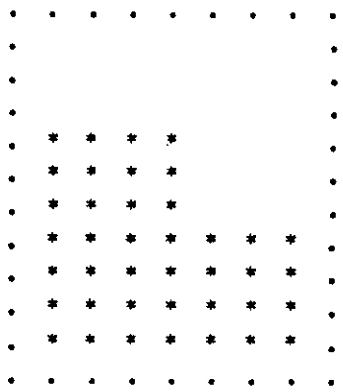
CODE 97 H



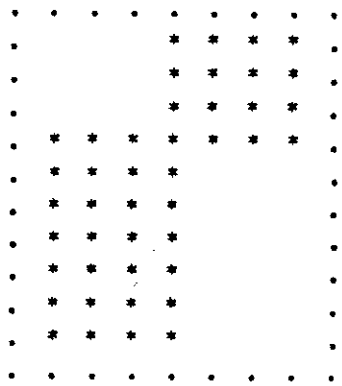
CODE 98 H



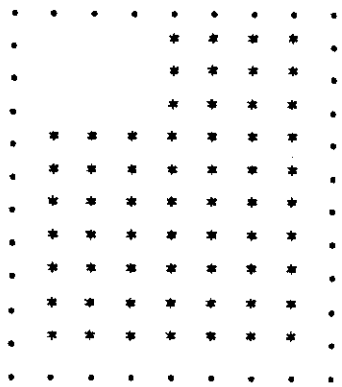
CODE 99 H



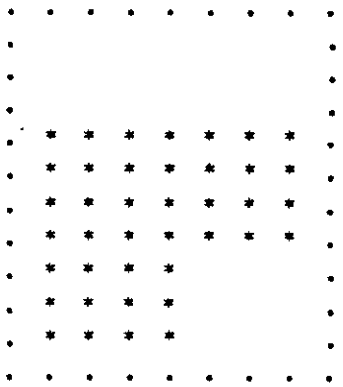
CODE 9C H



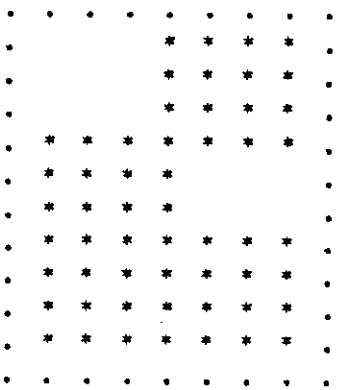
CODE 9F H



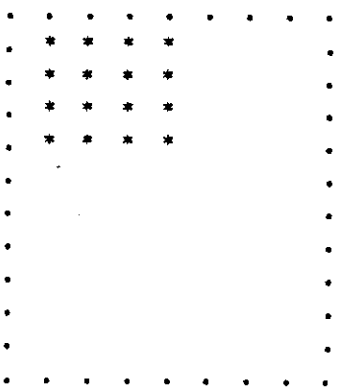
CODE 9A H



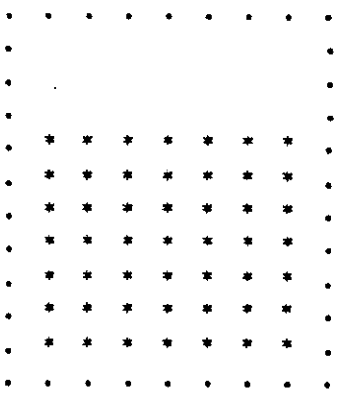
CODE 9D H



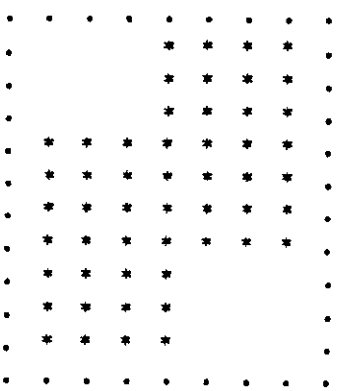
CODE A0 H



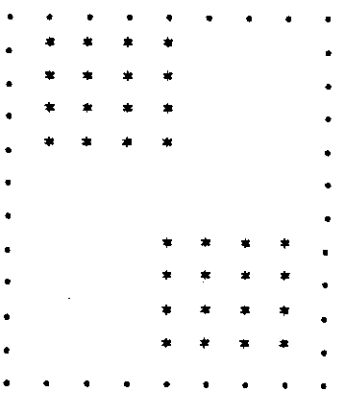
CODE 9B H



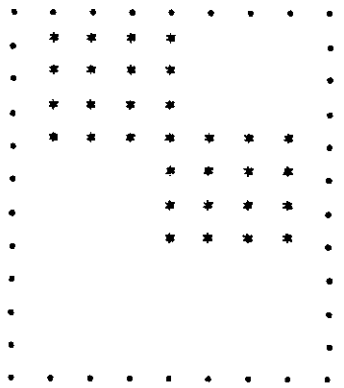
CODE 9E H



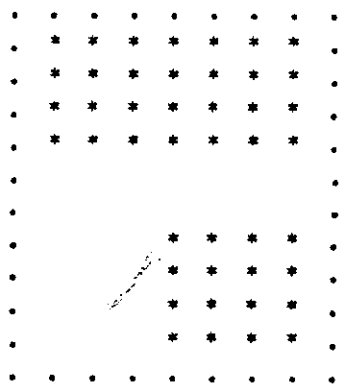
CODE A1 H



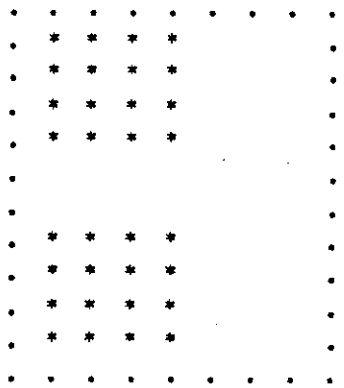
CODE A2 H



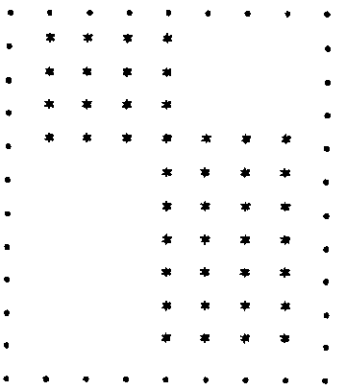
CODE A5 H



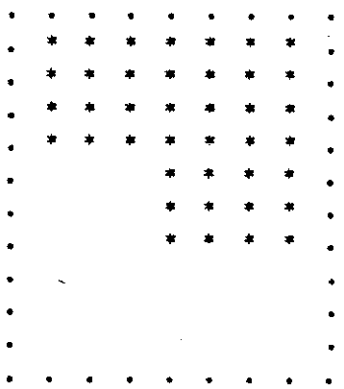
CODE A8 H



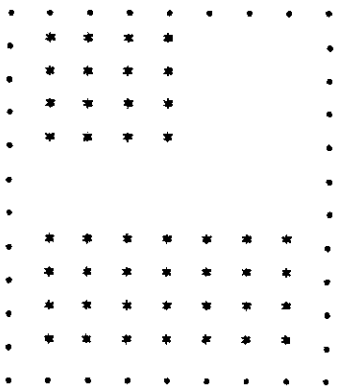
CODE A3 H



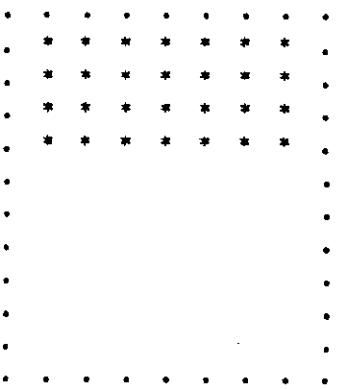
CODE A6 H



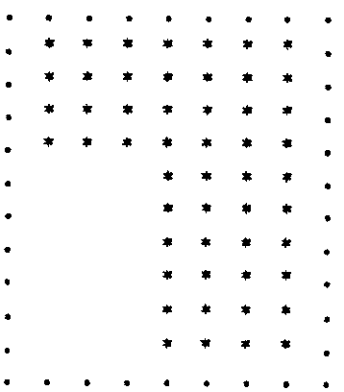
CODE A9 H



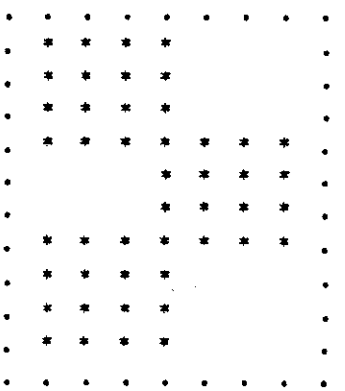
CODE A4 H



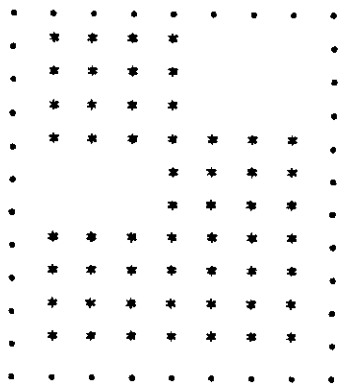
CODE A7 H



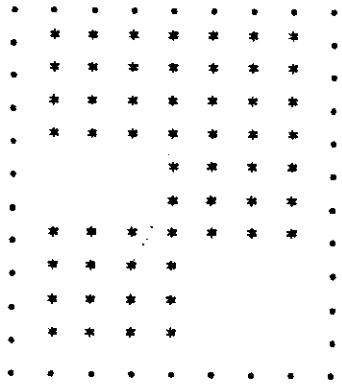
CODE AA H



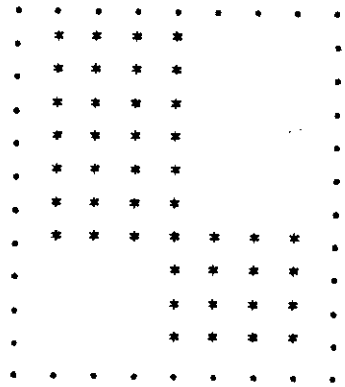
CODE A8 H



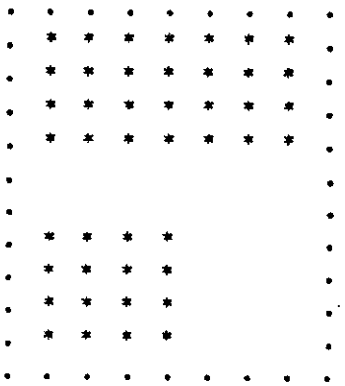
CODE AE H



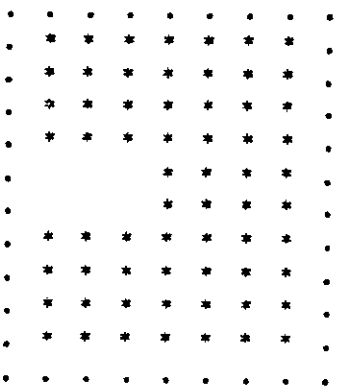
CODE B1 H



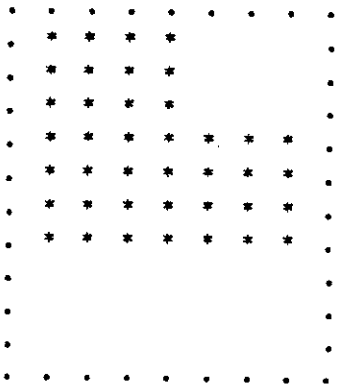
CODE AC H



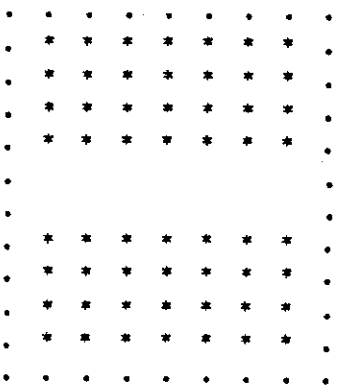
CODE AF H



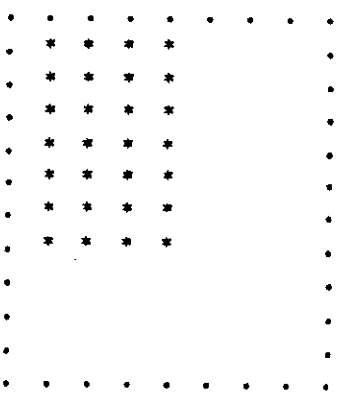
CODE B2 H



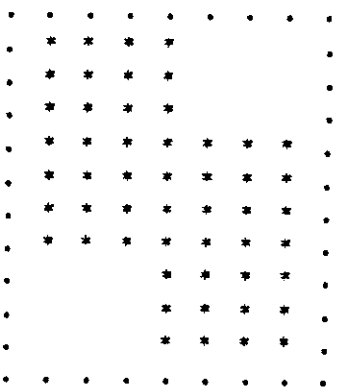
CODE AD H



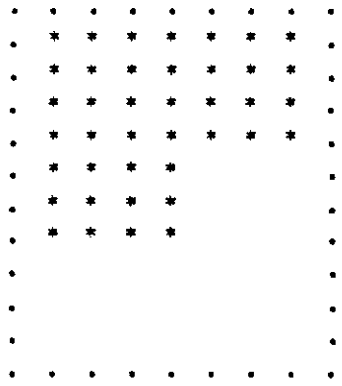
CODE B0 H



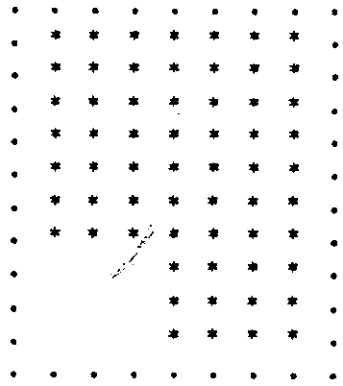
CODE B3 H



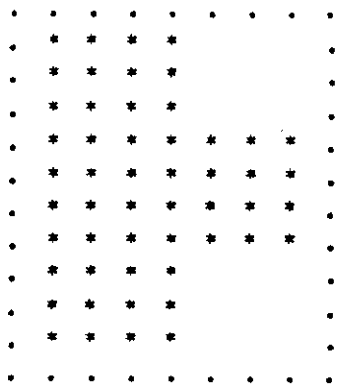
CODE B4 H



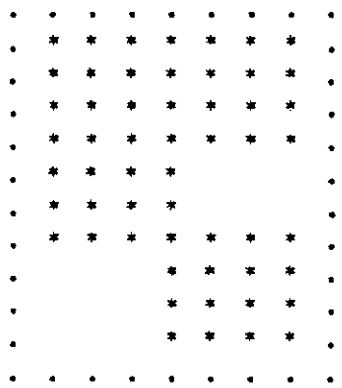
CODE B7 H



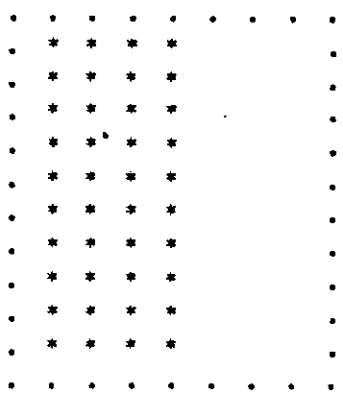
CODE BA H



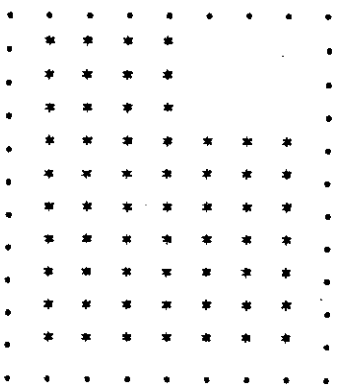
CODE B5 H



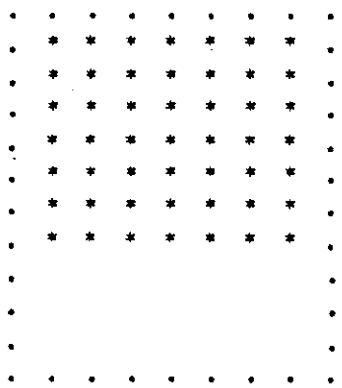
CODE B8 H



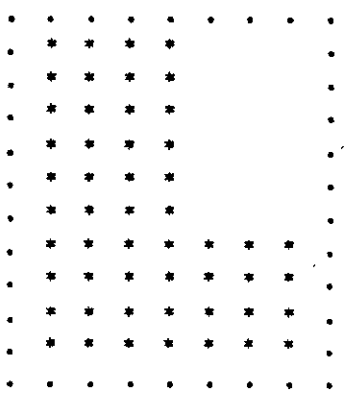
CODE BB H



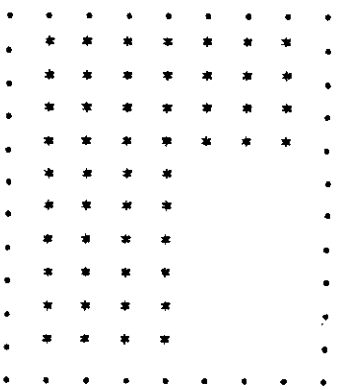
CODE B6 H



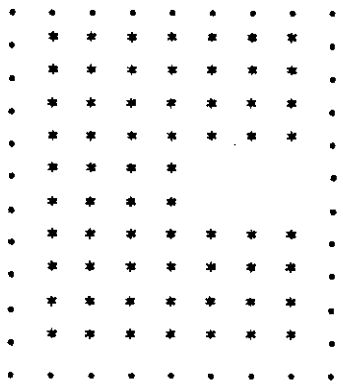
CODE B9 H



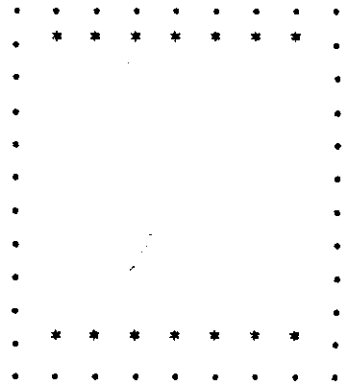
CODE BC H



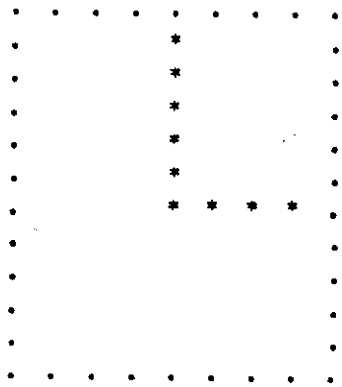
CODE BD H



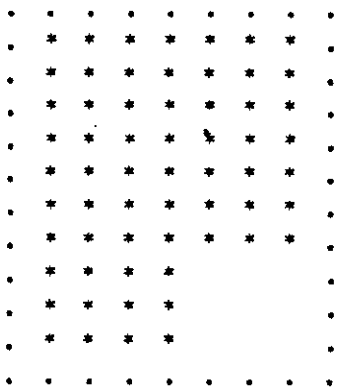
CODE C0 H



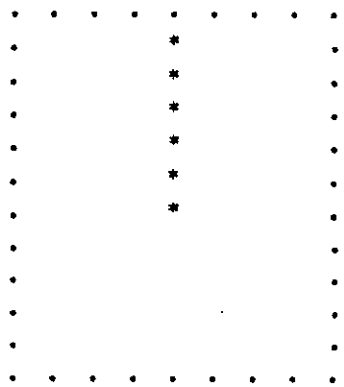
CODE C3 H



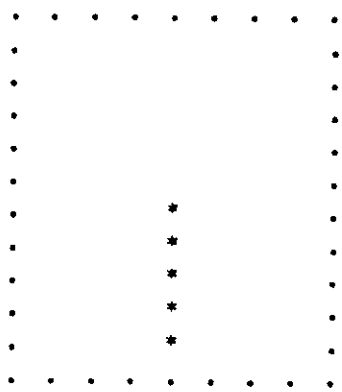
CODE BE H



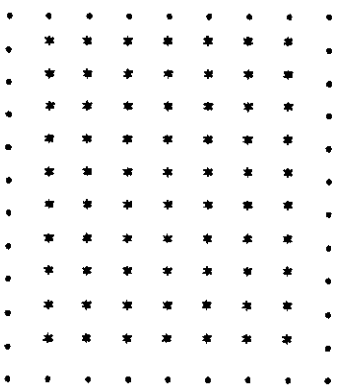
CODE C1 H



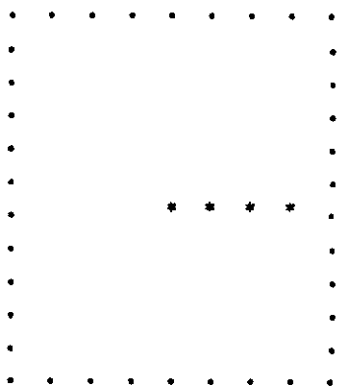
CODE C4 H



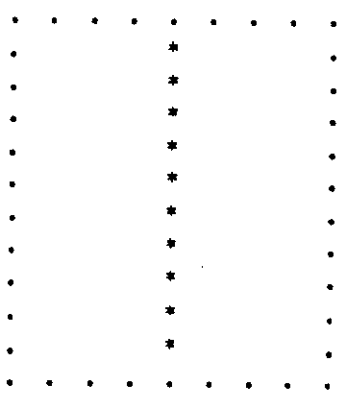
CODE BF H



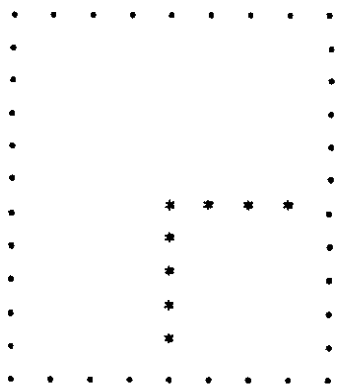
CODE C2 H



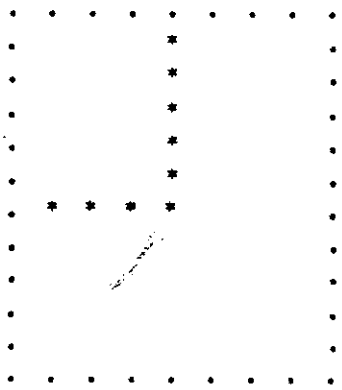
CODE C5 H



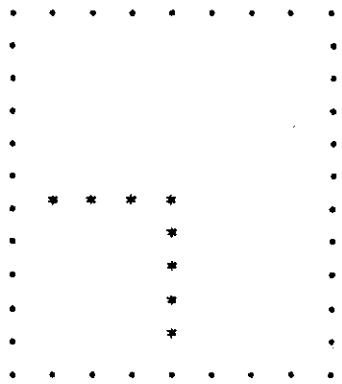
CODE C6 H



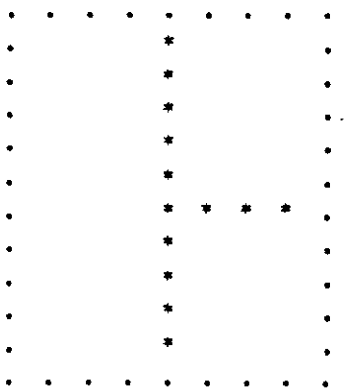
CODE C9 H



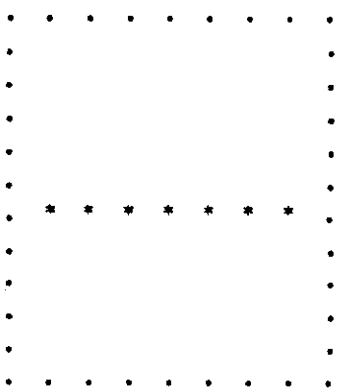
CODE CC H



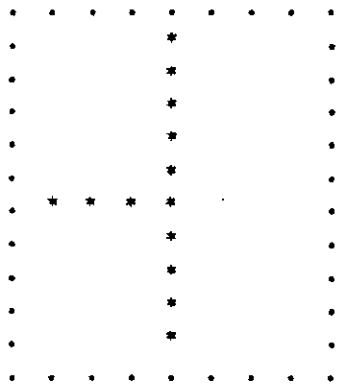
CODE C7 H



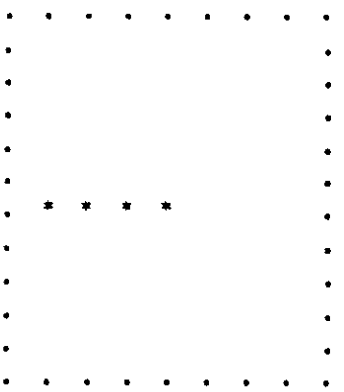
CODE CA H



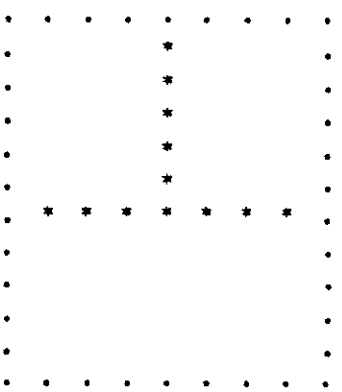
CODE CD H



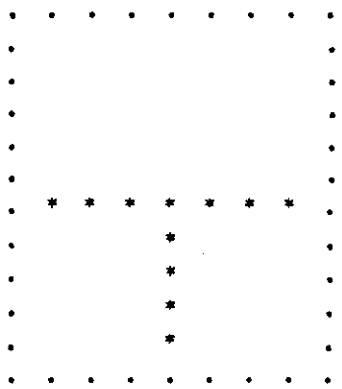
CODE C8 H



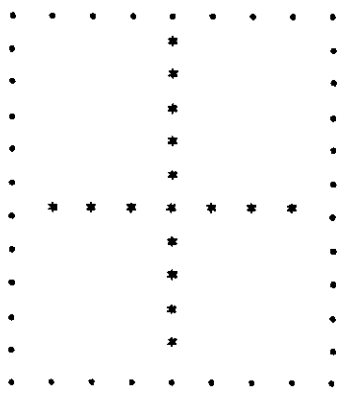
CODE CB H



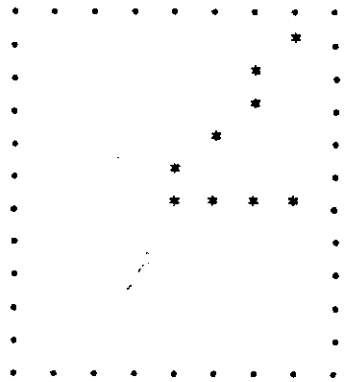
CODE CE H



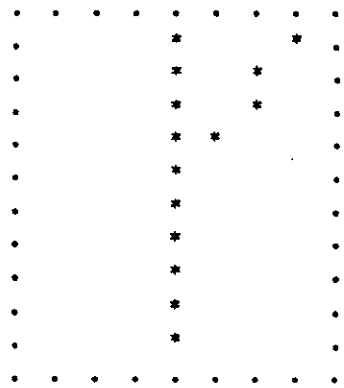
CODE CF H



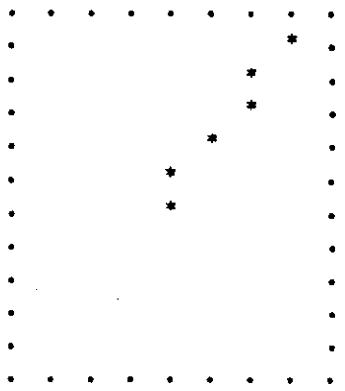
CODE D2 H



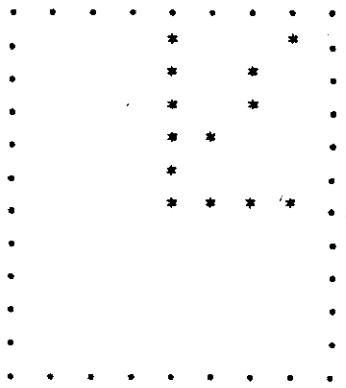
CODE D5 H



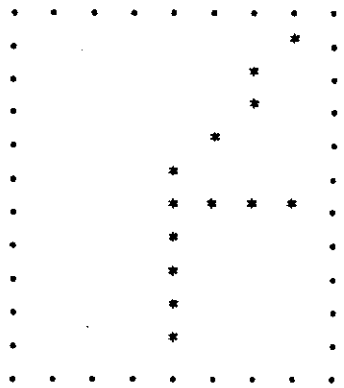
CODE D0 H



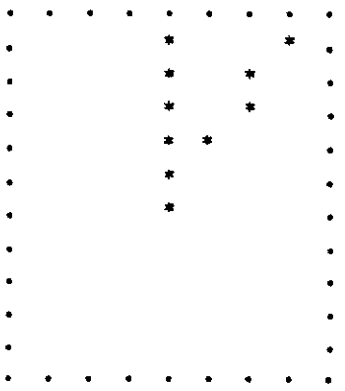
CODE D3 H



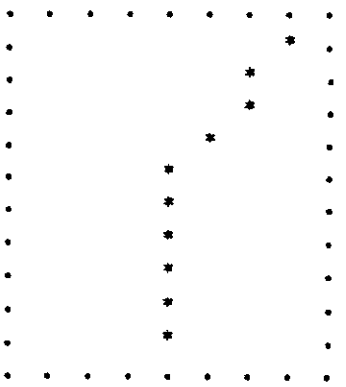
CODE D6 H



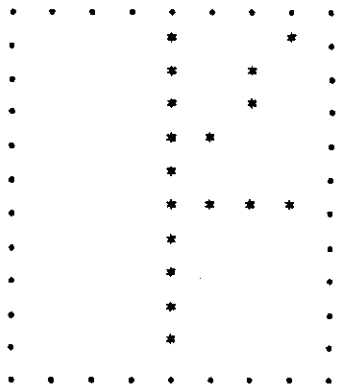
CODE D1 H



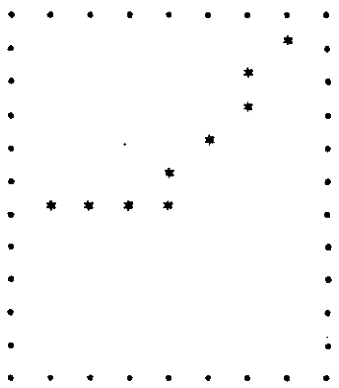
CODE D4 H



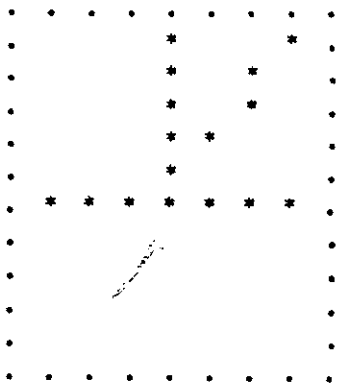
CODE D7 H



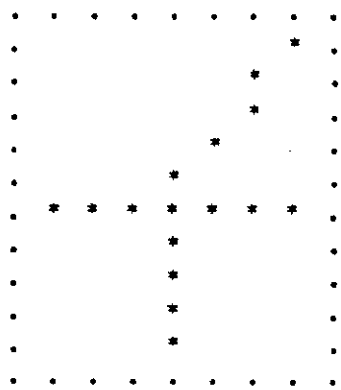
CODE D8 H



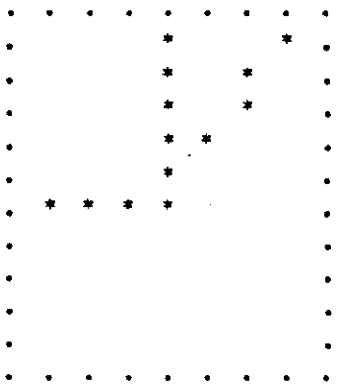
CODE DB H



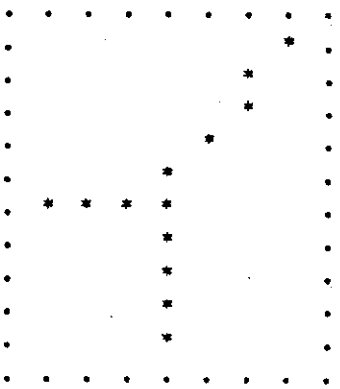
CODE DE H



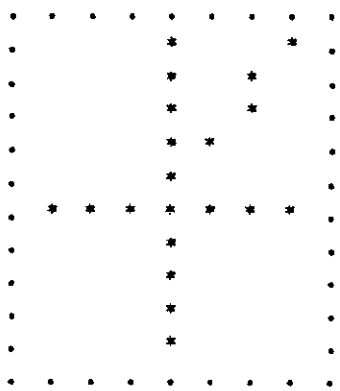
CODE D9 H



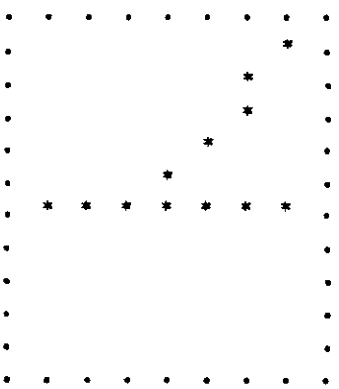
CODE DC H



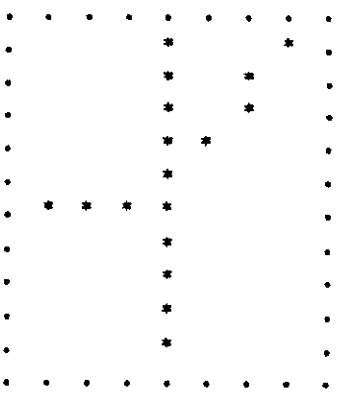
CODE DF H



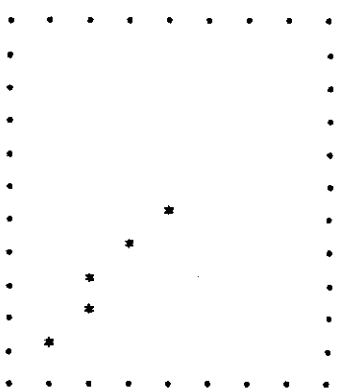
CODE DA H



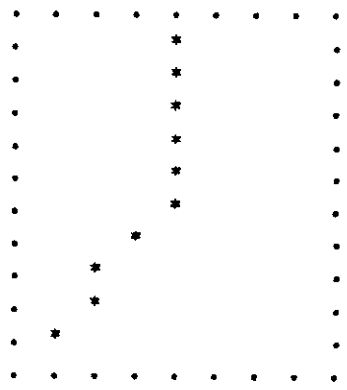
CODE DD H



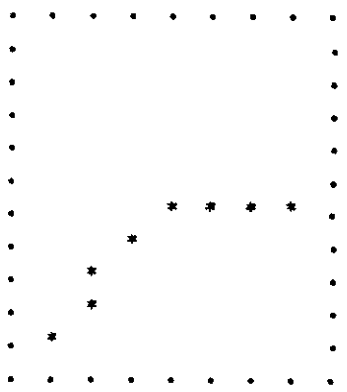
CODE E0 H



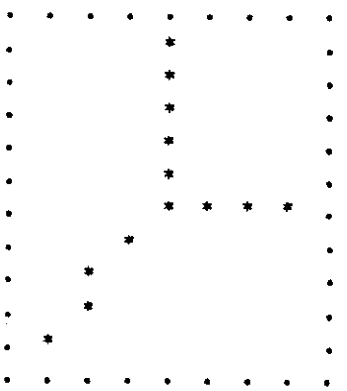
CODE E1 H



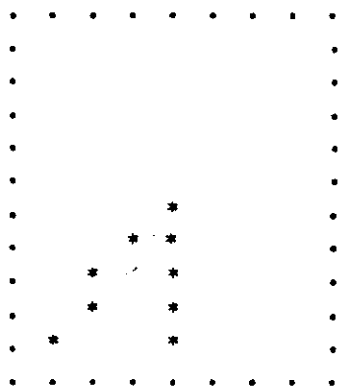
CODE E2 H



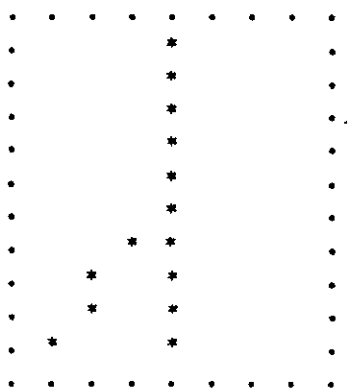
CODE E3 H



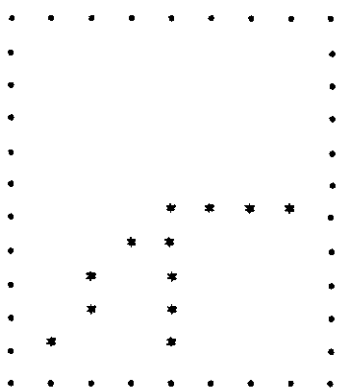
CODE E4 H



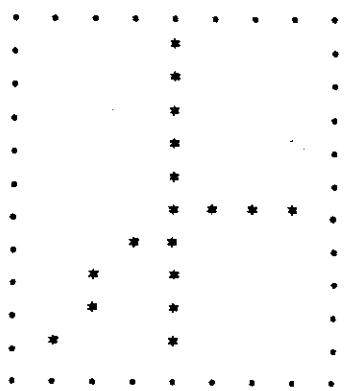
CODE E5 H



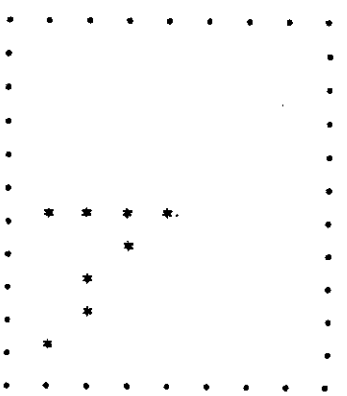
CODE E6 H



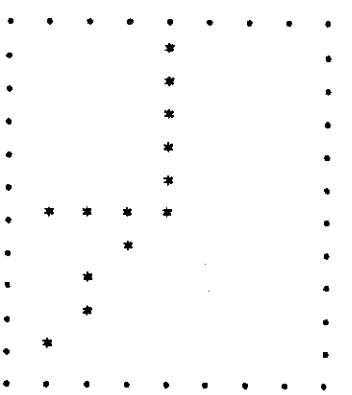
CODE E7 H



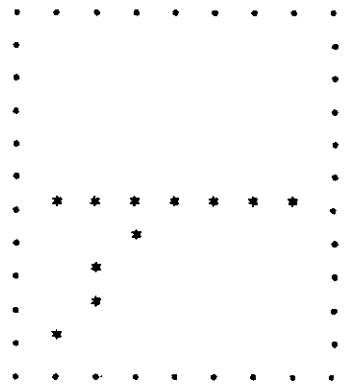
CODE E8 H



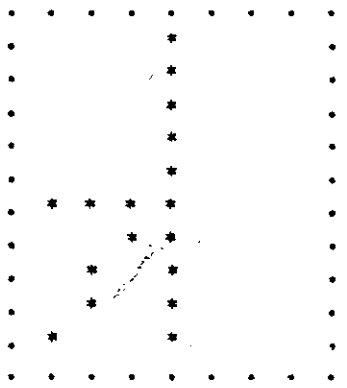
CODE E9 H



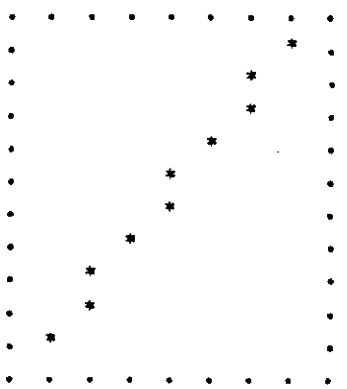
CODE EA H



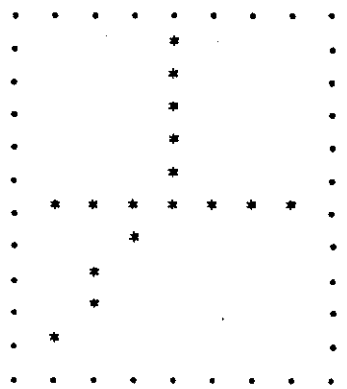
CODE ED H



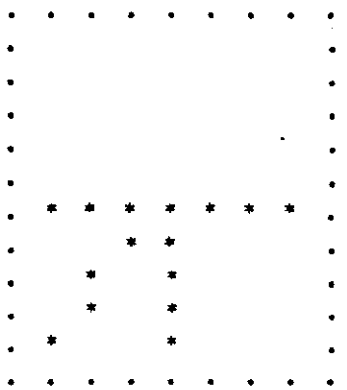
CODE F0 H



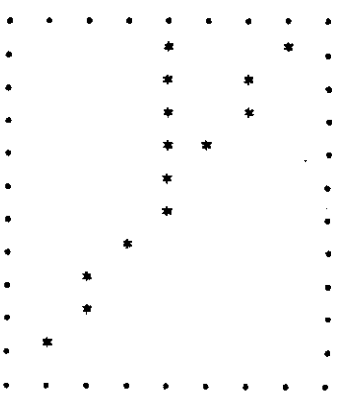
CODE E8 H



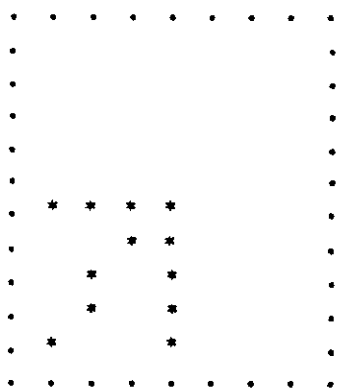
CODE EE H



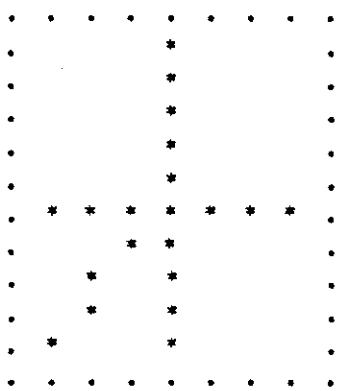
CODE F1 H



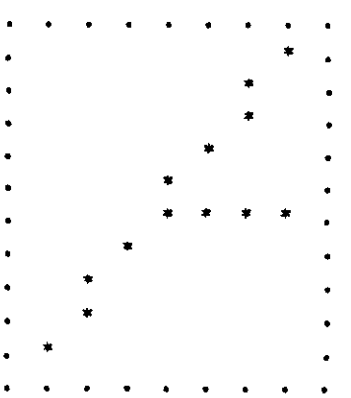
CODE EC H



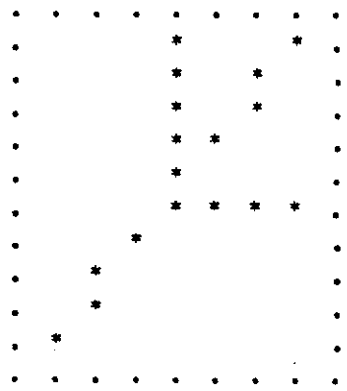
CODE EF H



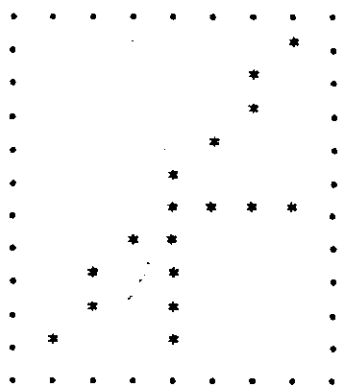
CODE F2 H



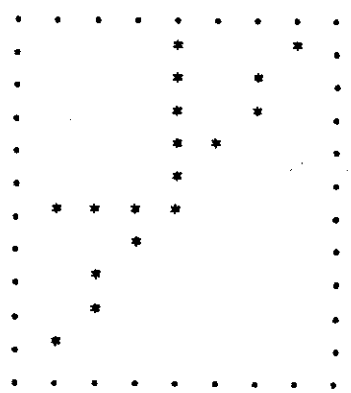
CODE F3 H



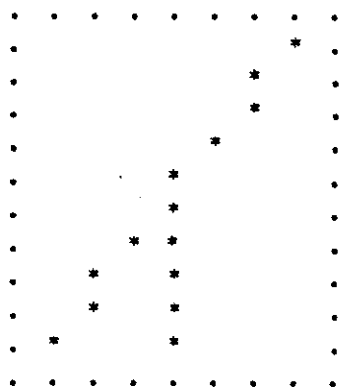
CODE F6 H



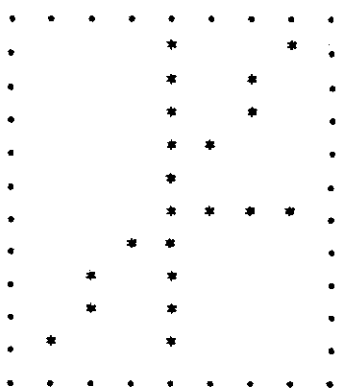
CODE F9 H



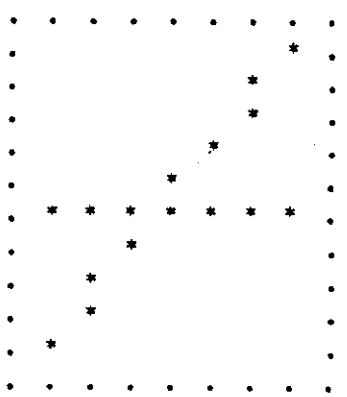
CODE F4 H



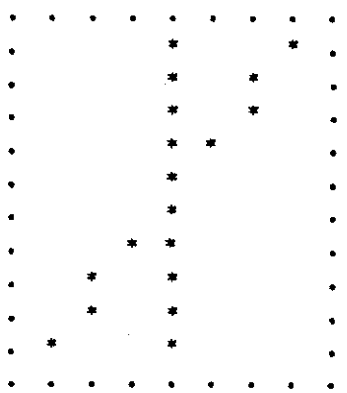
CODE F7 H



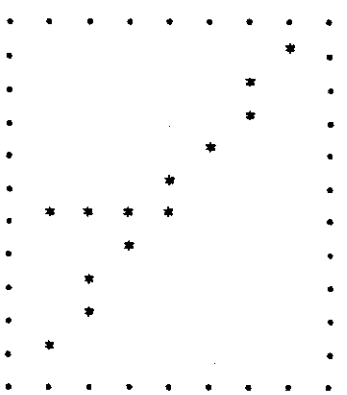
CODE FA H



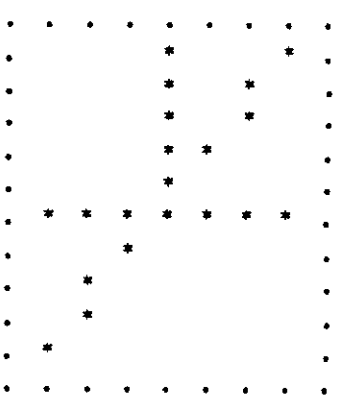
CODE F5 H



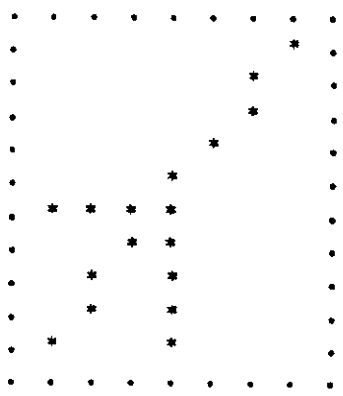
CODE F8 H



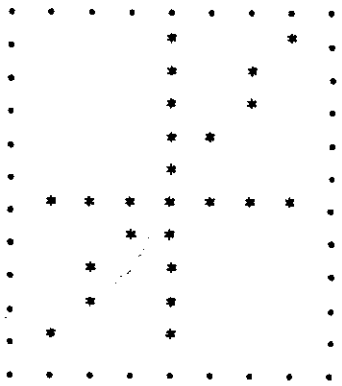
CODE FB H



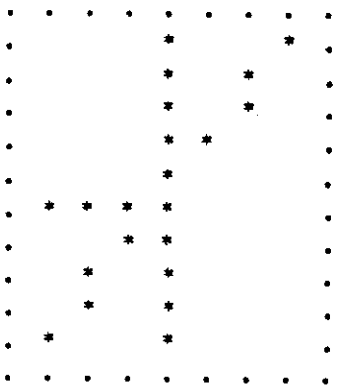
CODE FC H



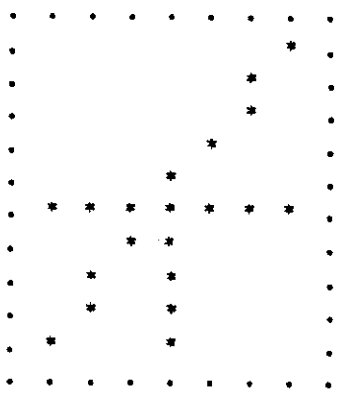
CODE FF H



CODE FD H



CODE FE H



APPENDIX 2 ----------

X,Y POSITION CODES

Y OR X	CODE	ASCII
1	20 H	SPACE
2	21 H	!
3	22 H	"
4	23 H	#
5	24 H	$
6	25 H	%
7	26 H	&
8	27 H	'
9	28 H	(
10	29 H	)
11	2A H	*
12	2B H	+
13	2C H	`
14	2D H	-
15	2E H	.
16	2F H	/
17	30 H	0
18	31 H	1
19	32 H	2
20	33 H	3
21	34 H	4
22	35 H	5
23	36 H	6
24	37 H	7
25	38 H	8
26	39 H	9
27	3A H	:

X	CODE	ASCII
28	3B H	;
29	3C H	<
30	3D H	=
31	3E H	>
32	3F H	?
33	40 H	@
34	41 H	A
35	42 H	B
36	43 H	C
37	44 H	D
38	45 H	E
39	46 H	F
40	47 H	G
41	48 H	H
42	49 H	I
43	4A H	J
44	4B H	K
45	4C H	L
46	4D H	M
47	4E H	N
48	4F H	O
49	50 H	P
50	51 H	Q
51	52 H	R
52	53 H	S
53	54 H	T
54	55 H	U

X	CODE	ASCII
55	56 H	V
56	57 H	W
57	58 H	X
58	59 H	Y
59	5A H	Z
60	5B H	[
61	5C H	\
62	5D H	]
63	5E H	^
64	5F H	
65	60 H	`
66	61 H	a
67	62 H	b
68	63 H	c
69	64 H	d
70	65 H	e
71	66 H	f
72	67 H	g
73	68 H	h
74	69 H	i
75	6A H	j
76	6B H	k
77	6C H	l
78	6D H	m
79	6E H	n
80	6F H	o

APPENDIX 3

Simple VIO Driver

This appendix lists a simple VIO driver, assembled to reside at location 0100H. The driver assumes that the refresh memory resides in location F000 - F7FF. It will set-up the display with the following parameters:

40 characters per line
24 lines per page
positive video display
Mode 3 (graphics and ASCII characters may be displayed)

When it is desired to display a character on the screen, the user program should call the driver entry point (0100H) with the character code (for the character to be displayed) in the accumulator. Note that codes 00 - FFH (Appendix 1) may be displayed with the following three exceptions.

CRTL-Z (1AH) will cause the screen to be erased
CR (0DH) will be recognized as a carriage return/line feed.
LF (0AH) will be ignored.

APPENDIX 3 ----------

SIMPLE VIO DRIVER LISTING

```
                        ;*********************************************
                        ;VIO DRIVER
                        ;PUTS SCREEN IN 40/24 FORMAT AND SCROLLS AT
                        ;CARRIAGE RETURN/LINE FEED.
                        ;RECOGNIZED 1AH AS SCREEN ERASE
                        ;RECOGNIZES 0DH AS CARRIAGE RETURN/LINE FEED
                        ;IGNORES 0AH (LINE FEED)
                        ;*********************************************
F7FF =          CTRPORT EQU 0F7FFH      ;HARDWARE CONTROL PORT
F000 =          REFRESH EQU 0F000H      ;REFRESH MEMORY LOCATION
0100                    ORG 100H
0100 E5         VIODRV  PUSH H
0101 C5                 PUSH B
0102 D5                 PUSH D
0103 F5                 PUSH PSW
0104 21FFF7             LXI H,CTRPORT
0107 36AD               MVI M,0ADH      ;24 BY 40
0109 FE1A               CPI 1AH         ;INIT?
010B C23301             JNZ VIO1
010E 01C003     ERASE   LXI B,960
0111 2100F0             LXI H,REFRESH
0114 3620       ERASE1  MVI M,' '
0116 0B                 DCX B
0117 23                 INX H
0118 78                 MOV A,B
0119 B1                 ORA C
011A C21401             JNZ ERASE1      ;CONTINUE UNTIL SCREEN ERASED
011D 21C0F3     LNER1   LXI H,REFRESH+960       ;LAST LINE
0120 0628               MVI B,40        ;40 CHARS PER LINE
0122 2B         LNER    DCX H
0123 3620               MVI M,' '
0125 05                 DCR B
0126 C22201             JNZ LNER
0129 367F       RETURN  MVI M,7FH       ;CURSOR CHAR
012B 226301             SHLD CURPTR
012E F1                 POP PSW
012F D1                 POP D
0130 C1                 POP B
0131 E1                 POP H           ;RESTORE USER STUFF
```

```
0132 C9              RET
0133 2A6301  VIO1    LHLD CURPTR
0136 3620            MVI M,' '          ;REMOVE CURSOR
0138 FE0A            CPI 0AH            ;LINE FEED?
013A CA2901          JZ RETURN          ;YES.,IGNORE IT
013D FE0D            CPI 0DH            ;CARRIAGE RETURN?
013F C25801          JNZ VIO2           ;NO,DO INDIVID CHAR
0142 2100F0  SCROLL  LXI H,REFRESH
0145 1128F0          LXI D,REFRESH+40
0148 019803          LXI B,920
014B 1A      SCRL1   LDAX D             ;GET CHAR
014C 77              MOV M,A            ;MOVE IT
014D 23              INX H
014E 13              INX D
014F 0B              DCX B
0150 78              MOV A,B
0151 B1              ORA C
0152 C24B01          JNZ SCRL1
0155 C31D01          JMP LNER1
0158 77      VIO2    MOV M,A            ;PUT CHAR ON SCREEN
0159 23              INX H              ;BUMP CURSOR
015A 7D              MOV A,L
015B FEC0            CPI 0C0H           ;END OF LINE?
015D CA4201          JZ SCROLL          ;YES
0160 C32901          JMP RETURN
0163 0000    CURPTR  DW 0               ;CURSOR PTR STORAGE
             ;
0200                 ORG 200H
0200 310002  VIOTEST LXI SP,200H
0203 DB03            IN 3
0205 E602            ANI 2
0207 CA0302          JZ VIOTEST+3
020A DB02            IN 2
020C CD0001          CALL VIODRV
020F C30302          JMP VIOTEST+3
0212                 END
```

APPENDIX 4

PROGRAMMING THE CHARACTER GENERATOR ROMS/PROMS

When it is desired to implement other character sets, the VIO's character generator ROMS/PROMS may be appropriately reprogrammed or replaced.

The three 2308/2708 ROMS/PROMs at locations U47, U48, and U49 contain the dot patterns for the entire 256 character set of the VIO. Each character consists of a 7x10 dot matrix which includes inter-character and inter-line spacing. The top eight rows of the 128 character represented by character codes 00-7F H, are stored in U49. The top eight rows of the second group of characters, (represented by codes 80-FF H), are stored in U48. U47 contains the dot patterns for the bottom two rows of the entire 256 character set.

NOTE: If only U49 is installed, the descenders of all lower case letters will be missing.

CHARACTER GENERATOR EXAMPLE

Figure III-22 shows an example of the ROM/EPROM entries for implimenting the character generator look-up tables. The figure shows a representation of the number "7" in a 5x7 dot format. The VIO uses a 7x10 dot format but this includes the inter-character and inter-line spacing. By including the spacing areas in the character matrix it is possible to form "no gap" graphic characters. Alphanumeric characters are normally limited to the 5x7 dot format.
Each row (seven dots) of each character has its own entry in the look-up table. The "dots" are stored in a "negative logic" format, i.e., a zero corresponds to a dot while a one corresponds to a blank. The left-most dot position in a row corresponds to the least significant bit in the associated look-up table entry.

The ASCII code for the number "7" is HEX 37. The correspondance between codes and ROM/PROM addresses is shown in Tables III-2 and III-3. Since 37 falls in the lower half of the character set, Table III-2 should be used. Note that for this character code, scans (rows) 0-9 correspond to locations 037, 0B7, 137, 1B7, 237, 2B7, 337, 3B7 of U49 and 037, 0B7 of U47.

There are no dots in the zero row. Because of the negative logic format this corresponds to all 7 bits being equal to one. Therefore the entry 7F is made at location 037 of ROM U49.

Row one has dots at all locations except the first and last. This requires an entry of 41 in location 0B7 of U49. Row two has a dot only in position 5. Recall that the left-most dot corresponds to the least significant bit. This pattern therefore corresponds to a 5F entry in location 137 of U49.

The remaining entries are as shown in the figure. It is very important to remember that the entries are in negative logic format and that the left-most dot position corresponds to the least significant bit in the entry.

Users who wish to do extensive character set development may wish to use the program CHARGEN included with this chapter. This program is written in IMSAI Extended BASIC.

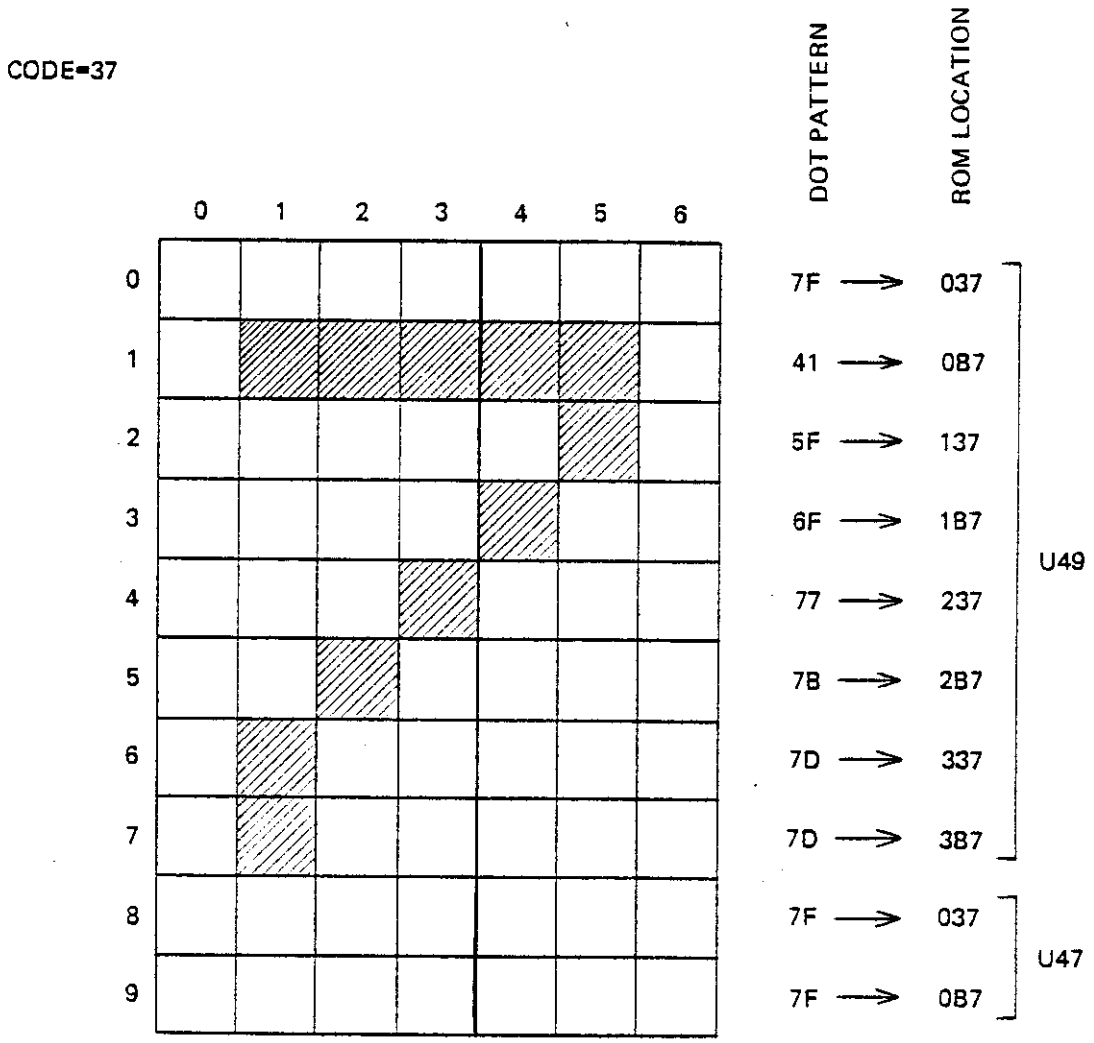


FIGURE III-22 CHARACTER GENERATOR EXAMPLE

TABLE III-2. LOW 128 FONT ROM ADDRESSES

CODE	ROM ADDRESS (U49)									(U47)		
	1	2	3	4	5	6	7	8		9	10	SCAN
00	000	080	100	180	200	280	300	380	--	000	080	
01	001	081	101	181	201	281	301	381	--	001	081	
02	002	082	102	182	202	282	302	382	--	002	082	
03	003	083	103	183	203	283	303	383	--	003	083	
04	004	084	104	184	204	284	304	384	--	004	084	
05	005	085	105	185	205	285	305	385	--	005	085	
06	006	086	106	186	206	286	306	386	--	006	086	
07	007	087	107	187	207	287	307	387	--	007	087	
08	008	088	108	188	208	288	308	388	--	008	088	
09	009	089	109	189	209	289	309	389	--	009	089	
0A	00A	08A	10A	18A	20A	28A	30A	38A	--	00A	08A	
0B	00B	08B	10B	18B	20B	28B	30B	38B	--	00B	08B	
0C	00C	08C	10C	18C	20C	28C	30C	38C	--	00C	08C	
0D	00D	08D	10D	18D	20D	28D	30D	38D	--	00D	08D	
0E	00E	08E	10E	18E	20E	28E	30E	38E	--	00E	08E	
0F	00F	08F	10F	18F	20F	28F	30F	38F	--	00F	08F	
10	010	090	110	190	210	290	310	390	--	010	090	
11	011	091	111	191	211	291	311	391	--	011	091	
12	012	092	112	192	212	292	312	392	--	012	092	
13	013	093	113	193	213	293	313	393	--	013	093	
14	014	094	114	194	214	294	314	394	--	014	094	
15	015	095	115	195	215	295	315	395	--	015	095	
16	016	096	116	196	216	296	316	396	--	016	096	
17	017	097	117	197	217	297	317	397	--	017	097	
18	018	098	118	198	218	298	318	398	--	018	098	
19	019	099	119	199	219	299	319	399	--	019	099	
1A	01A	09A	11A	19A	21A	29A	31A	39A	--	01A	09A	
1B	01B	09B	11B	19B	21B	29B	31B	39B	--	01B	09B	
1C	01C	09C	11C	19C	21C	29C	31C	39C	--	01C	09C	
1D	01D	09D	11D	19D	21D	29D	31D	39D	--	01D	09D	
1E	01E	09E	11E	19E	21E	29E	31E	39E	--	01E	09E	
1F	01F	09F	11F	19F	21F	29F	31F	39F	--	01F	09F	
20	020	0A0	120	1A0	220	2A0	320	3A0	--	020	0A0	
21	021	0A1	121	1A1	221	2A1	321	3A1	--	021	0A1	
22	022	0A2	122	1A2	222	2A2	322	3A2	--	022	0A2	
23	023	0A3	123	1A3	223	2A3	323	3A3	--	023	0A3	
24	024	0A4	124	1A4	224	2A4	324	3A4	--	024	0A4	
25	025	0A5	125	1A5	225	2A5	325	3A5	--	025	0A5	
26	026	0A6	126	1A6	226	2A6	326	3A6	--	026	0A6	
27	027	0A7	127	1A7	227	2A7	327	3A7	--	027	0A7	
28	028	0A8	128	1A8	228	2A8	328	3A8	--	028	0A8	
29	029	0A9	129	1A9	229	2A9	329	3A9	--	029	0A9	
2A	02A	0AA	12A	1AA	22A	2AA	32A	3AA	--	02A	0AA	
2B	02B	0AB	12B	1AB	22B	2AB	32B	3AB	--	02B	0AB	

2C	02C	0AC	12C	1AC	22C	2AC	32C	3AC	--	02C	0AC
2D	02D	0AD	12D	1AD	22D	2AD	32D	3AD	--	02D	0AD
2E	02E	0AE	12E	1AE	22E	2AE	32E	3AE	--	02E	0AE
2F	02F	0AF	12F	1AF	22F	2AF	32F	3AF	--	02F	0AF
30	030	0B0	130	1B0	230	2B0	330	3B0	--	030	0B0
31	031	0B1	131	1B1	231	2B1	331	3B1	--	031	0B1
32	032	0B2	132	1B2	232	2B2	332	3B2	--	032	0B2
33	033	0B3	133	1B3	233	2B3	333	3B3	--	033	0B3
34	034	0B4	134	1B4	234	2B4	334	3B4	--	034	0B4
35	035	0B5	135	1B5	235	2B5	335	3B5	--	035	0B5
36	036	0B6	136	1B6	236	2B6	336	3B6	--	036	0B6
37	037	0B7	137	1B7	237	2B7	337	3B7	--	037	0B7
38	038	0B8	138	1B8	238	2B8	338	3B8	--	038	0B8
39	039	0B9	139	1B9	239	2B9	339	3B9	--	039	0B9
3A	03A	0BA	13A	1BA	23A	2BA	33A	3BA	--	03A	0BA
3B	03B	0BB	13B	1BB	23B	2BB	33B	3BB	--	03B	0BB
3C	03C	0BC	13C	1BC	23C	2BC	33C	3BC	--	03C	0BC
3D	03D	0BD	13D	1BD	23D	2BD	33D	3BD	--	03D	0BD
3E	03E	0BE	13E	1BE	23E	2BE	33E	3BE	--	03E	0BE
3F	03F	0BF	13F	1BF	23F	2BF	33F	3BF	--	03F	0BF
40	040	0C0	140	1C0	240	2C0	340	3C0	--	040	0C0
41	041	0C1	141	1C1	241	2C1	341	3C1	--	041	0C1
42	042	0C2	142	1C2	242	2C2	342	3C2	--	042	0C2
43	043	0C3	143	1C3	243	2C3	343	3C3	--	043	0C3
44	044	0C4	144	1C4	244	2C4	344	3C4	--	044	0C4
45	045	0C5	145	1C5	245	2C5	345	3C5	--	045	0C5
46	046	0C6	146	1C6	246	2C6	346	3C6	--	046	0C6
47	047	0C7	147	1C7	247	2C7	347	3C7	--	047	0C7
48	048	0C8	148	1C8	248	2C8	348	3C8	--	048	0C8
49	049	0C9	149	1C9	249	2C9	349	3C9	--	049	0C9
4A	04A	0CA	14A	1CA	24A	2CA	34A	3CA	--	04A	0CA
4B	04B	0CB	14B	1CB	24B	2CB	34B	3CB	--	04B	0CB
4C	04C	0CC	14C	1CC	24C	2CC	34C	3CC	--	04C	0CC
4D	04D	0CD	14D	1CD	24D	2CD	34D	3CD	--	04D	0CD
4E	04E	0CE	14E	1CE	24E	2CE	34E	3CE	--	04E	0CE
4F	04F	0CF	14F	1CF	24F	2CF	34F	3CF	--	04F	0CF
50	050	0D0	150	1D0	250	2D0	350	3D0	--	050	0D0
51	051	0D1	151	1D1	251	2D1	351	3D1	--	051	0D1
52	052	0D2	152	1D2	252	2D2	352	3D2	--	052	0D2
53	053	0D3	153	1D3	253	2D3	353	3D3	--	053	0D3
54	054	0D4	154	1D4	254	2D4	354	3D4	--	054	0D4
55	055	0D5	155	1D5	255	2D5	355	3D5	--	055	0D5
56	056	0D6	156	1D6	256	2D6	356	3D6	--	056	0D6
57	057	0D7	157	1D7	257	2D7	357	3D7	--	057	0D7
58	058	0D8	158	1D8	258	2D8	358	3D8	--	058	0D8
59	059	0D9	159	1D9	259	2D9	359	3D9	--	059	0D9
5A	05A	0DA	15A	1DA	25A	2DA	35A	3DA	--	05A	0DA
5B	05B	0DB	15B	1DB	25B	2DB	35B	3DB	--	05B	0DB

5C	05C	0DC	15C	1DC	25C	2DC	35C	3DC	--	05C	0DC
5D	05D	0DD	15D	1DD	25D	2DD	35D	3DD	--	05D	0DD
5E	05E	0DE	15E	1DE	25E	2DE	35E	3DE	--	05E	0DE
5F	05F	0DF	15F	1DF	25F	2DF	35F	3DF	--	05F	0DF
60	060	0E0	160	1E0	260	2E0	360	3E0	--	060	0E0
61	061	0E1	161	1E1	261	2E1	361	3E1	--	061	0E1
62	062	0E2	162	1E2	262	2E2	362	3E2	--	062	0E2
63	063	0E3	163	1E3	263	2E3	363	3E3	--	063	0E3
64	064	0E4	164	1E4	264	2E4	364	3E4	--	064	0E4
65	065	0E5	165	1E5	265	2E5	365	3E5	--	065	0E5
66	066	0E6	166	1E6	266	2E6	366	3E6	--	066	0E6
67	067	0E7	167	1E7	267	2E7	367	3E7	--	067	0E7
68	068	0E8	168	1E8	268	2E8	368	3E8	--	068	0E8
69	069	0E9	169	1E9	269	2E9	369	3E9	--	069	0E9
6A	06A	0EA	16A	1EA	26A	2EA	36A	3EA	--	06A	0EA
6B	06B	0EB	16B	1EB	26B	2EB	36B	3EB	--	06B	0EB
6C	06C	0EC	16C	1EC	26C	2EC	36C	3EC	--	06C	0EC
6D	06D	0ED	16D	1ED	26D	2ED	36D	3ED	--	06D	0ED
6E	06E	0EE	16E	1EE	26E	2EE	36E	3EE	--	06E	0EE
6F	06F	0EF	16F	1EF	26F	2EF	36F	3EF	--	06F	0EF
70	070	0F0	170	1F0	270	2F0	370	3F0	--	070	0F0
71	071	0F1	171	1F1	271	2F1	371	3F1	--	071	0F1
72	072	0F2	172	1F2	272	2F2	372	3F2	--	072	0F2
73	073	0F3	173	1F3	273	2F3	373	3F3	--	073	0F3
74	074	0F4	174	1F4	274	2F4	374	3F4	--	074	0F4
75	075	0F5	175	1F5	275	2F5	375	3F5	--	075	0F5
76	076	0F6	176	1F6	276	2F6	376	3F6	--	076	0F6
77	077	0F7	177	1F7	277	2F7	377	3F7	--	077	0F7
78	078	0F8	178	1F8	278	2F8	378	3F8	--	078	0F8
79	079	0F9	179	1F9	279	2F9	379	3F9	--	079	0F9
7A	07A	0FA	17A	1FA	27A	2FA	37A	3FA	--	07A	0FA
7B	07B	0FB	17B	1FB	27B	2FB	37B	3FB	--	07B	0FB
7C	07C	0FC	17C	1FC	27C	2FC	37C	3FC	--	07C	0FC
7D	07D	0FD	17D	1FD	27D	2FD	37D	3FD	--	07D	0FD
7E	07E	0FE	17E	1FE	27E	2FE	37E	3FE	--	07E	0FE
7F	07F	0FF	17F	1FF	27F	2FF	37F	3FF	--	07F	0FF

TABLE III-3. HIGH 128 FONT ROM ADDRESSES

CODE	ROM ADDRESS (U48)									(U47)		
	1	2	3	4	5	6	7	8		9	10	SCAN
80	000	080	100	180	200	280	300	380	--	200	280	
81	001	081	101	181	201	281	301	381	--	201	281	
82	002	082	102	182	202	282	302	382	--	202	282	
83	003	083	103	183	203	283	303	383	--	203	283	
84	004	084	104	184	204	284	304	384	--	204	284	
85	005	085	105	185	205	285	305	385	--	205	285	
86	006	086	106	186	206	286	306	386	--	206	286	
87	007	087	107	187	207	287	307	387	--	207	287	
88	008	088	108	188	208	288	308	388	--	208	288	
89	009	089	109	189	209	289	309	389	--	209	289	
8A	00A	08A	10A	18A	20A	28A	30A	38A	--	20A	28A	
8B	00B	08B	10B	18B	20B	28B	30B	38B	--	20B	28B	
8C	00C	08C	10C	18C	20C	28C	30C	38C	--	20C	28C	
8D	00D	08D	10D	18D	20D	28D	30D	38D	--	20D	28D	
8E	00E	08E	10E	18E	20E	28E	30E	38E	--	20E	28E	
8F	00F	08F	10F	18F	20F	28F	30F	38F	--	20F	28F	
90	010	090	110	190	210	290	310	390	--	210	290	
91	011	091	111	191	211	291	311	391	--	211	291	
92	012	092	112	192	212	292	312	392	--	212	292	
93	013	093	113	193	213	293	313	393	--	213	293	
94	014	094	114	194	214	294	314	394	--	214	294	
95	015	095	115	195	215	295	315	395	--	215	295	
96	016	096	116	196	216	296	316	396	--	216	296	
97	017	097	117	197	217	297	317	397	--	217	297	
98	018	098	118	198	218	298	318	398	--	218	298	
99	019	099	119	199	219	299	319	399	--	219	299	
9A	01A	09A	11A	19A	21A	29A	31A	39A	--	21A	29A	
9B	01B	09B	11B	19B	21B	29B	31B	39B	--	21B	29B	
9C	01C	09C	11C	19C	21C	29C	31C	39C	--	21C	29C	
9D	01D	09D	11D	19D	21D	29D	31D	39D	--	21D	29D	
9E	01E	09E	11E	19E	21E	29E	31E	39E	--	21E	29E	
9F	01F	09F	11F	19F	21F	29F	31F	39F	--	21F	29F	
A0	020	0A0	120	1A0	220	2A0	320	3A0	--	220	2A0	
A1	021	0A1	121	1A1	221	2A1	321	3A1	--	221	2A1	
A2	022	0A2	122	1A2	222	2A2	322	3A2	--	222	2A2	
A3	023	0A3	123	1A3	223	2A3	323	3A3	--	223	2A3	
A4	024	0A4	124	1A4	224	2A4	324	3A4	--	224	2A4	
A5	025	0A5	125	1A5	225	2A5	325	3A5	--	225	2A5	
A6	026	0A6	126	1A6	226	2A6	326	3A6	--	226	2A6	
A7	027	0A7	127	1A7	227	2A7	327	3A7	--	227	2A7	
A8	028	0A8	128	1A8	228	2A8	328	3A8	--	228	2A8	
A9	029	0A9	129	1A9	229	2A9	329	3A9	--	229	2A9	
AA	02A	0AA	12A	1AA	22A	2AA	32A	3AA	--	22A	2AA	
AB	02B	0AB	12B	1AB	22B	2AB	32B	3AB	--	22B	2AB	

AC	02C	0AC	12C	1AC	22C	2AC	32C	3AC	--	22C	2AC
AD	02D	0AD	12D	1AD	22D	2AD	32D	3AD	--	22D	2AD
AE	02E	0AE	12E	1AE	22E	2AE	32E	3AE	--	22E	2AE
AF	02F	0AF	12F	1AF	22F	2AF	32F	3AF	--	22F	2AF
B0	030	0B0	130	1B0	230	2B0	330	3B0	--	230	2B0
B1	031	0B1	131	1B1	231	2B1	331	3B1	--	231	2B1
B2	032	0B2	132	1B2	232	2B2	332	3B2	--	232	2B2
B3	033	0B3	133	1B3	233	2B3	333	3B3	--	233	2B3
B4	034	0B4	134	1B4	234	2B4	334	3B4	--	234	2B4
B5	035	0B5	135	1B5	235	2B5	335	3B5	--	235	2B5
B6	036	0B6	136	1B6	236	2B6	336	3B6	--	236	2B6
B7	037	0B7	137	1B7	237	2B7	337	3B7	--	237	2B7
B8	038	0B8	138	1B8	238	2B8	338	3B8	--	238	2B8
B9	039	0B9	139	1B9	239	2B9	339	3B9	--	239	2B9
BA	03A	0BA	13A	1BA	23A	2BA	33A	3BA	--	23A	2BA
BB	03B	0BB	13B	1BB	23B	2BB	33B	3BB	--	23B	2BB
BC	03C	0BC	13C	1BC	23C	2BC	33C	3BC	--	23C	2BC
BD	03D	0BD	13D	1BD	23D	2BD	33D	3BD	--	23D	2BD
BE	03E	0BE	13E	1BE	23E	2BE	33E	3BE	--	23E	2BE
BF	03F	0BF	13F	1BF	23F	2BF	33F	3BF	--	23F	2BF
C0	040	0C0	140	1C0	240	2C0	340	3C0	--	240	2C0
C1	041	0C1	141	1C1	241	2C1	341	3C1	--	241	2C1
C2	042	0C2	142	1C2	242	2C2	342	3C2	--	242	2C2
C3	043	0C3	143	1C3	243	2C3	343	3C3	--	243	2C3
C4	044	0C4	144	1C4	244	2C4	344	3C4	--	244	2C4
C5	045	0C5	145	1C5	245	2C5	345	3C5	--	245	2C5
C6	046	0C6	146	1C6	246	2C6	346	3C6	--	246	2C6
C7	047	0C7	147	1C7	247	2C7	347	3C7	--	247	2C7
C8	048	0C8	148	1C8	248	2C8	348	3C8	--	248	2C8
C9	049	0C9	149	1C9	249	2C9	349	3C9	--	249	2C9
CA	04A	0CA	14A	1CA	24A	2CA	34A	3CA	--	24A	2CA
CB	04B	0CB	14B	1CB	24B	2CB	34B	3CB	--	24B	2CB
CC	04C	0CC	14C	1CC	24C	2CC	34C	3CC	--	24C	2CC
CD	04D	0CD	14D	1CD	24D	2CD	34D	3CD	--	24D	2CD
CE	04E	0CE	14E	1CE	24E	2CE	34E	3CE	--	24E	2CE
CF	04F	0CF	14F	1CF	24F	2CF	34F	3CF	--	24F	2CF
D0	050	0D0	150	1D0	250	2D0	350	3D0	--	250	2D0
D1	051	0D1	151	1D1	251	2D1	351	3D1	--	251	2D1
D2	052	0D2	152	1D2	252	2D2	352	3D2	--	252	2D2
D3	053	0D3	153	1D3	253	2D3	353	3D3	--	253	2D3
D4	054	0D4	154	1D4	254	2D4	354	3D4	--	254	2D4
D5	055	0D5	155	1D5	255	2D5	355	3D5	--	255	2D5
D6	056	0D6	156	1D6	256	2D6	356	3D6	--	256	2D6
D7	057	0D7	157	1D7	257	2D7	357	3D7	--	257	2D7
D8	058	0D8	158	1D8	258	2D8	358	3D8	--	258	2D8
D9	059	0D9	159	1D9	259	2D9	359	3D9	--	259	2D9
DA	05A	0DA	15A	1DA	25A	2DA	35A	3DA	--	25A	2DA
DB	05B	0DB	15B	1DB	25B	2DB	35B	3DB	--	25B	2DB

DC	05C	0DC	15C	1DC	25C	2DC	35C	3DC	--	25C	2DC
DD	05D	0DD	15D	1DD	25D	2DD	35D	3DD	--	25D	2DD
DE	05E	0DE	15E	1DE	25E	2DE	35E	3DE	--	25E	2DE
DF	05F	0DF	15F	1DF	25F	2DF	35F	3DF	--	25F	2DF
E0	060	0E0	160	1E0	260	2E0	360	3E0	--	260	2E0
E1	061	0E1	161	1E1	261	2E1	361	3E1	--	261	2E1
E2	062	0E2	162	1E2	262	2E2	362	3E2	--	262	2E2
E3	063	0E3	163	1E3	263	2E3	363	3E3	--	263	2E3
E4	064	0E4	164	1E4	264	2E4	364	3E4	--	264	2E4
E5	065	0E5	165	1E5	265	2E5	365	3E5	--	265	2E5
E6	066	0E6	166	1E6	266	2E6	366	3E6	--	266	2E6
E7	067	0E7	167	1E7	267	2E7	367	3E7	--	267	2E7
E8	068	0E8	168	1E8	268	2E8	368	3E8	--	268	2E8
E9	069	0E9	169	1E9	269	2E9	369	3E9	--	269	2E9
EA	06A	0EA	16A	1EA	26A	2EA	36A	3EA	--	26A	2EA
EB	06B	0EB	16B	1EB	26B	2EB	36B	3EB	--	26B	2EB
EC	06C	0EC	16C	1EC	26C	2EC	36C	3EC	--	26C	2EC
ED	06D	0ED	16D	1ED	26D	2ED	36D	3ED	--	26D	2ED
EE	06E	0EE	16E	1EE	26E	2EE	36E	3EE	--	26E	2EE
EF	06F	0EF	16F	1EF	26F	2EF	36F	3EF	--	26F	2EF
F0	070	0F0	170	1F0	270	2F0	370	3F0	--	270	2F0
F1	071	0F1	171	1F1	271	2F1	371	3F1	--	271	2F1
F2	072	0F2	172	1F2	272	2F2	372	3F2	--	272	2F2
F3	073	0F3	173	1F3	273	2F3	373	3F3	--	273	2F3
F4	074	0F4	174	1F4	274	2F4	374	3F4	--	274	2F4
F5	075	0F5	175	1F5	275	2F5	375	3F5	--	275	2F5
F6	076	0F6	176	1F6	276	2F6	376	3F6	--	276	2F6
F7	077	0F7	177	1F7	277	2F7	377	3F7	--	277	2F7
F8	078	0F8	178	1F8	278	2F8	378	3F8	--	278	2F8
F9	079	0F9	179	1F9	279	2F9	379	3F9	--	279	2F9
FA	07A	0FA	17A	1FA	27A	2FA	37A	3FA	--	27A	2FA
FB	07B	0FB	17B	1FB	27B	2FB	37B	3FB	--	27B	2FB
FC	07C	0FC	17C	1FC	27C	2FC	37C	3FC	--	27C	2FC
FD	07D	0FD	17D	1FD	27D	2FD	37D	3FD	--	27D	2FD
FE	07E	0FE	17E	1FE	27E	2FE	37E	3FE	--	27E	2FE
FF	07F	0FF	17F	1FF	27F	2FF	37F	3FF	--	27F	2FF

CHARACTER GENERATOR PROGRAM

The program CHARGEN.BAS was used to develop the standard character font for the IMSAI VIO. Users who wish to do extensive font development may wish to use this or a similar program of their own design. CHARGEN is written in IMSAI Extended BASIC and requires at least 32K of memory. It allows the user to interactively create fonts and generates HEX files for PROM programming.

The following is a sample session with CHARGEN:

```
        A>RUN-E CHARGEN
BASIC-E INTERPRETER - VER 2.2


NAME OF CHARACTER FILE? PFONT
NEW FILE (Y/N)? Y


ENTER CODE OR COMMAND? 00000000
INPUT CHARACTER MATRIX

LINE 1          .? "
LINE 2          .? "  *****
LINE 3          .? "  *    *
LINE 4          .? 2
ILLEGAL CHARACTER
LINE 4          .? "  *    *
LINE 5          .? "  *    *
LINE 6          .? "  *    *
LINE 7          .? "  *****
LINE 8          .? "
LINE 9          .? "
LINE 10         .? "
```

```
CODE IS 00000000

CHARACTER IS:
. . . . . .
.           .
.   * * * * *   .
.   *       *   .
.   *       *   .
.   *       *   .
.   *       *   .
.   * * * * *   .
.               .
.               .
.               .
.               .
. . . . . .

CHARACTER OK (Y/N)? Y



ENTER CODE OR COMMAND? 0000001
ENTER 8 BINARY DIGITS


ENTER CODE OR COMMAND? 00000001
INPUT CHARACTER MATRIX

LINE 1          .? "*
LINE 2          .? " *
LINE 3          .? "  *
LINE 4          .? "   *
LINE 5          .? "    *
LINE 6          .? "     *
LINE 7          .? "      *
LINE 8          .? "       *
LINE TOO LONG
LINE 8          .? "
LINE 9          .? "
LINE 10         .? "
```

```
CODE IS 00000001

CHARACTER IS:
. . . . . .
. *         .
.   *       .
.     *     .
.       *   .
.         * .
.           *
.           .
.           .
.           .
. . . . . .

CHARACTER OK (Y/N)? N
REENTER CHARACTER
INPUT CHARACTER MATRIX

LINE 1          .? "       *
LINE 2          .? "      *
LINE 3          .? "     *
LINE 4          .? "    *
LINE 5          .? "   *
LINE 6          .? "  *
LINE 7          .? " *
LINE 8          .? "
LINE 9          .? "
LINE 10         .? "

CODE IS 00000001

CHARACTER IS:
. . . . . .
.           *
.         *   .
.       *     .
.     *       .
.   *         .
.  *          .
.  *          .
.  *          .
.  *          .
. . . . . .
```

```
CHARACTER OK (Y/N)? Y



ENTER CODE OR COMMAND? E00000000

CODE IS 00000000

CHARACTER IS:
. . . . . .
.           .
.   * * * *   .
.   *       *   .
.   *       *   .
.   *       *   .
.   *       *   .
.   * * * *   .
.           .
.           .
.           .
. . . . . .

CHARACTER OK (Y/N)? Y

ENTER CODE OR COMMAND? Q

A>
```

In this example we are working on a font named PFONT. CHARGEN opens four files. The first three are hex files (PFONT1.HEX, PFONT2.HEX and PFONT3.HEX) corresponding to the character generator ROM's U49, U48 and U47. The fourth file PFONT.CHR can be used to dump a printout of the font.

The "Y" response to "NEW FILE (Y/N)?" causes the hex files to be initialized to all blank characters. After the first session with a new font you should always answer "N" to this prompt to prevent destroying the entries of previous sessions.

The characters may be entered in any order. When prompted by "ENTER CODE OR COMMAND?" type in exactly eight binary digits representing the code for the character you desire to enter. In response to the line prompt, type a double quote followed by seven or fewer spaces and *'s as appropriate for that character. After the tenth line the code and dot matrix will be echoed as a check for proper entry. If you respond "Y" to the "CHARACTER OK (Y/N)?" prompt, the character will be entered into the hex files. Any other response will require that the dot matrix be re-entered.

If you wish to examine a previously entered character, type "E" before the eight bit code. If you wish to change this character, respond "N" to the OK prompt.

There are three valid commands which may be given the program. The command "Q" terminates a session by simply closing all files. The command "DUMP" causes the complete 256 character font to be printed on the terminal and written into the type CHR file. This may take several hours if the terminal speed is low. The command "CKSUM" causes the check sums for all hex files to be updated. These computations require several minutes. The "CKSUM" command is only required when the entry of characters is completed and you are ready to use the hex files for burning PROM's.

As can be noted in the example, CHARGEN will respond with error messages when invalid entries are made. Attempting to operate on uninitiated hex files will cause program termination without any error message. The message "HEX ERROR" indicates a program malfunction and should not be encountered by the user.

The "CKSUM" and "DUMP" commands should not be used unless the user allows them to run to completion. Attempts to abort these commands may cause the file contents to be lost.

The program listing follows:

```
REM
REM       PROGRAM CHARGEN
REM
REM       THIS PROGRAM INTERACTIVELY GENERATES
REM       HEX FILES FOR THE CHARACTER GENERATOR
REM       ROMS FOR THE IMSAI VIO VIDEO BOARD
REM
REM
REM       COPYRIGHT 1977 IMSAI MANAFACTURING CORP.
REM
REM       VERS 1.0 GEE


IF END #1 THEN 9999
IF END #2 THEN 9999
IF END #3 THEN 9999

          HEXCONV$="0123456789ABCDEF"
          DIM LN$(10)
          DIM STAR$(16)
          STAR$(0)="****"
          STAR$(1)=" ***"
          STAR$(2)="* **"
          STAR$(3)="  **"
          STAR$(4)="** *"
          STAR$(5)=" * *"
          STAR$(6)="*  *"
          STAR$(7)="   *"
          STAR$(8)="*** "
          STAR$(9)=" ** "
          STAR$(10)="* * "
          STAR$(11)="  * "
          STAR$(12)="**  "
          STAR$(13)=" *  "
          STAR$(14)="*   "
          STAR$(15)="    "

          INPUT "NAME OF CHARACTER FILE";SFILE$
          DFILE$=LEFT$(SFILE$,1)
          IF LEN(SFILE$)=1 THEN 20
          FOR I=2 TO LEN(SFILE$)
          F$=MID$(SFILE$,I,1)
          IF F$="." THEN 20
          DFILE$=DFILE$+F$
          IF LEN(DFILE$)>6 THEN DFILE$=LEFT$(DFILE$,6)
          NEXT I

20
          DFILE1$=DFILE$+"1.HEX"
```

```
        DFILE2$=DFILE$+"2.HEX"
        DFILE3$=DFILE$+"3.HEX"
        DFILE4$=DFILE$+".CHR"


        FILE DFILE1$(50),DFILE2$(50),DFILE3$(50),DFILE4$

        INPUT "NEW FILE (Y/N)";A$
        IF LEFT$(A$,1)<>"Y" THEN 100

        FOR I=1 TO 64
        N=I-1
        GOSUB 200
        LINE$=":100"+HEX$+"000FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF"
        PRINT #1,I;LINE$
        PRINT #2,I;LINE$
        PRINT #3,I;LINE$
        NEXT I
        LINE$=":00000000000"
        PRINT #1,65;LINE$
        PRINT #2,65;LINE$
        PRINT #3,65;LINE$
100
        PRINT:PRINT:PRINT
        INPUT "ENTER CODE";A$
        DFLAG=0
        IF A$<>"DUMP" THEN 101
        DFLAG=1
        GOTO 700
101
        IF LEFT$(A$,1)="Q" THEN 9999
        IF A$="CKSUM" THEN 999
        EFLAG=0
        IF LEFT$(A$,1)<>"E" THEN 102
        EFLAG=1
        A$=RIGHT$(A$,LEN(A$)-1)
102
        IF LEN(A$)=8 THEN 110
        PRINT "ENTER 8 BINARY DIGITS"
        GOTO 100
110
        V=0
        F=128
        FOR I=1 TO 8
        AA$=MID$(A$,I,1)
        IF (AA$="1") OR (AA$="0") THEN 115
        PRINT "ILLEGAL CHARACTER CODE"
        GOTO 100
115     IF AA$="1" THEN V=V+F
        F=F/2
```

```
        NEXT I

        IF EFLAG=1 THEN 800
120
        PRINT "INPUT CHARACTER MATRIX"
        PRINT
        EFLAG=0
        FOR J=1 TO 10
125
        PRINT "LINE ";J,".";
        INPUT LN$(J)
        IF LEN(LN$(J))<8 THEN 130
        PRINT "LINE TOO LONG"
        GOTO 125
130
        IF LEN(LN$(J))=7 THEN 135
        LN$(J)=LN$(J)+" "
        GOTO 130
135
        FOR K=1 TO 7
        LNN$=MID$(LN$(J),K,1)
        IF (LNN$=" ") OR (LNN$="*") THEN 140
        PRINT "ILLEGAL CHARACTER"
        GOTO 125
140
        NEXT K
        NEXT J
142
        PRINT:PRINT:PRINT "CODE IS ";A$
        PRINT:PRINT "CHARACTER IS:":PRINT ". . . . . ."
        IF DFLAG<>1 THEN 104
        PRINT #4; " "
        PRINT #4; " "
        PRINT #4;"CHARACTER CODE:",A$
        PRINT #4;" "
        PRINT #4;". . . . . ."
104
        FOR J=1 TO 10
        LL$=". "
        FOR K=1 TO 7
        LL$=LL$+MID$(LN$(J),K,1)+" "
        NEXT K
        LL$=LL$+"."
        PRINT LL$
        IF DFLAG=1 THEN PRINT #4;LL$
        NEXT J
        PRINT ". . . . . ."
        IF DFLAG<>1 THEN 106
        PRINT #4;". . . . . ."
        RETURN
```

```
106
        PRINT:PRINT:INPUT "CHARACTER OK (Y/N)";B$
        IF (LEFT$(B$,1)="Y") AND (EFLAG=1) THEN 100
        IF LEFT$(B$,1)="Y" THEN 145
        PRINT "REENTER CHARACTER"
        GOTO 120
145
        FOR J=1 TO 10
        JJ=J-1
        N=0
        F=1
        FOR K=1 TO 7
        IF MID$(LN$(J),K,1)=" " THEN N=N+F
        F=F*2
        NEXT K
        GOSUB 200

        IF V>=128 THEN VV=V-128 ELSE VV=V
        IF JJ>7 THEN 150
        PPOSIT=JJ*128+VV
        LOWADR=INT(PPOSIT/16)+1
        IF V>=128 THEN READ #2,LOWADR;LINE$ ELSE READ #1,LOWADR;LINE$

        POSIT=(PPOSIT-16*LOWADR+16)*2+9
        LINE$=LEFT$(LINE$,POSIT)+HEX$+RIGHT$(LINE$,41-POSIT)
        IF V>=128 THEN PRINT #2,LOWADR;LINE$ ELSE PRINT #1,LOWADR;LINE$
        GOTO 157
150
        IF V>=128 THEN VVV=1 ELSE VVV=0
        PPOSIT=(JJ-8)*128+VVV*512+VV

        LOWADR=INT(PPOSIT/16)+1
        READ #3,LOWADR;LINE$
        POSIT=(PPOSIT-16*LOWADR+16)*2+9
        LINE$=LEFT$(LINE$,POSIT)+HEX$+RIGHT$(LINE$,41-POSIT)
        PRINT #3,LOWADR;LINE$
157
        NEXT J
        PRINT:PRINT
        GOTO 100
800
        FOR J=1 TO 10
        JJ=J-1
        IF V>=128 THEN VV=V-128 ELSE VV=V
        IF JJ>7 THEN 820
        PPOSIT=JJ*128+VV
        LOWADR=INT(PPOSIT/16)+1
        IF V>=128 THEN READ #2,LOWADR;LINE$ ELSE READ #1,LOWADR;LINE$
        GOTO 830
820
```

```
        IF V>=128 THEN VVV=1 ELSE VVV=0
        PPOSIT=(JJ-8)*128+VVV*512+VV
        LOWADR=INT(PPOSIT/16)+1
        READ #3,LOWADR;LINE$
830
        POSIT=(PPOSIT-16*LOWADR+16)*2+10
        CK$=MID$(LINE$,POSIT,1)
        GOSUB 500
        LN$(J)=LEFT$(STAR$(M),3)
        CK$=MID$(LINE$,POSIT+1,1)
        GOSUB 500
        LN$(J)=STAR$(M)+LN$(J)
        NEXT J
        GOTO 142


200     HEX16=INT(N/16)
        IF HEX16=16 THEN 250
        HEX=INT(N-HEX16*16)
        H16$=MID$(HEXCONV$,HEX16+1,1)
        H$=MID$(HEXCONV$,HEX+1,1)
        HEX$=H16$+H$
        RETURN
250
        HEX$="00"
        RETURN


400
        N=0
        L=2
410
        CK$=MID$(LINE$,L,1)
        GOSUB 500
420
        N=N+16*M
        L=L+1
        CK$=MID$(LINE$,L,1)
        GOSUB 500
430
        N=N+M
        IF N>=256 THEN N=N-256
        IF N<0 THEN N=0
        L=L+1
        IF L<42  THEN 410
        IF N>=256 THEN N=N-256
        N=ABS(256-N)
        GOSUB 200
        LINE$=LEFT$(LINE$,41)+HEX$
```

```
        RETURN

500     M=0
        IF CK$="0" THEN RETURN
        M=M+1
        IF CK$="1" THEN RETURN
        M=M+1
        IF CK$="2" THEN RETURN
        M=M+1
        IF CK$="3" THEN RETURN
        M=M+1
        IF CK$="4" THEN RETURN
        M=M+1
        IF CK$="5" THEN RETURN
        M=M+1
        IF CK$="6" THEN RETURN
        M=M+1
        IF CK$="7" THEN RETURN
        M=M+1
        IF CK$="8" THEN RETURN
        M=M+1
        IF CK$="9" THEN RETURN
        M=M+1
        IF CK$="A" THEN RETURN
        M=M+1
        IF CK$="B" THEN RETURN
        M=M+1
        IF CK$="C" THEN RETURN
        M=M+1
        IF CK$="D" THEN RETURN
        M=M+1
        IF CK$="E" THEN RETURN
        M=M+1
        IF CK$="F" THEN RETURN
        PRINT "HEX ERROR"
        GOTO 9999

999
        FOR I=1 TO 64
        READ #1,I;LINE$
        GOSUB 400
        PRINT #1,I;LINE$
        READ #2,I;LINE$
        GOSUB 400
        PRINT #2,I;LINE$
        READ #3,I;LINE$
        GOSUB 400
        PRINT #3,I;LINE$
        NEXT I
```

```
        GOTO 9999

700
        P$="0"
        FOR P=0 TO 1
        Q$="0"
        FOR Q=0 TO 1
        R$="0"
        FOR R=0 TO 1
        S$="0"
        FOR S=0 TO 1
        T$="0"
        FOR T=0 TO 1
        U$="0"
        FOR U=0 TO 1
        W$="0"
        FOR W=0 TO 1
        X$="0"
        FOR X=0 TO 1
        A$=P$+Q$+R$+S$+T$+U$+W$+X$
        V=X+2*W+4*U+8*T+16*S+32*R+64*Q+128*P
        GOSUB 800
        X$="1"
        NEXT X
        W$="1"
        NEXT W
        U$="1"
        NEXT U
        T$="1"
        NEXT T
        S$="1"
        NEXT S
        R$="1"
        NEXT R
        Q$="1"
        NEXT Q
        P$="1"
        NEXT P
9999 END
```

APPENDIX 5

Using CP/M (DOS-A) with the IMSAI VIO Video Display Board

The IMSAI CP/M Floppy Disk Operating System (DOS-A) Version 1.33 may be used with the IMSAI VIO Video Display Board and a television monitor.

Version 1.33 CP/M has been in existence since July 20, 1977. Registered owners of CP/M who do not have Ver. 1.33 may receive an updated version through IMSAI Customer Service for a nominal duplication charge.

Hardware Requirements

A keyboard must be used with the VIO to form a console device. This may be a separate keyboard or the keyboard of a CRT or other terminal. The keyboard must be interfaced to port 2, with status on port 3. This is the standard console port assignment in IMSAI software, and one of the two console port options already supported by CP/M.

Presence of the VIO in the system does not eliminate any other device support options; all of the other CP/M drivers may still be used. For instance, a terminal with a keyboard and printer (such as a teletype) might be connected to port 2. The operator could use the keyboard for console input and select between either the video display or the printer for output.

Operation

Cold Start

To select the VIO keyboard on port 2 as the console device when cold-starting (bootstrapping) the system, raise the programmed input switches 0 and 1 before hitting the RUN switch. To select the VIO as the list device (LST:), raise switches 6 and 7.

Reassignment in Running System

Once the CP/M system is running, I/O device assignments can be changed with the STAT console command, as described on page 21 of the System User's Guide for Version 1.33. To select the VIO and keyboard on port 2 as the console, use the command

```
STAT CON:=UC1
```

To select the VIO as the List device, use the command

```
STAT LST:=UL1
```

The VIO may be used for list output whether or not there is a keyboard on

port 2 and irrespective of what device is in use as a console.

VIO Mode Control

There are a number of special character sequences which, when output to the VIO, perform control functions such as clearing the screen or changing the number of characters displayed on a line.

These sequences may be typed in from the console when CP/M is awaiting command input if each character is rubbed out after it is typed. The rubout is necessary to eliminate the character from the next CP/M command line, and also because CP/M Version 1.33 echoes control characters as control characters only when they are rubbed out.

The control sequences are documented in detail in the VIO User Manual, Section 3.3. Descriptions of the ones that are most useful with CP/M follow. When typing these in from the keyboard, remember to type a rubout after each character. For example, "escape C" is typed as "escape, rubout, C, rubout". When output by a program, only the escape and the C should be output. On some keyboards that do not have an escape key, escape may be typed as shift-control-K.

escape C

Switch between 40 and 80 characters per line. The system comes up set to 80 characters per line; changing to 40 produces larger, more readable characters.

escape L

Switch between 12 and 24 lines of characters on the screen. System comes up set to 24 lines.

^Z (Control-Z)

Clear screen, move cursor to upper left. Should be given after switch from 24 to 12 lines, or from 80 to 40 characters.

escape V

Switch between white-on-black and black-on-white display.

escape U

Switch between upper-lower case display and upper case only. System comes up set for upper case display only; lower case characters output are displayed in upper case. NOTE: lower case characters are displayed without "tails" below the line unless the extended (256 character) set is installed in the VIO.

Limitations

The version of the system diagnostic, SHAKDOWN, supplied with CP/M Version 1.33 cannot output to the VIO. It is necessary to connect a console device other than the VIO to the system in order to run SHAKDOWN.

Control characters typed, or rubbed out, can sometimes perform undesired control functions. For instance, control-Z's are commonly used in command lines to CP/M text editor (ED). If a control-Z in a partly typed line is rubbed out, the screen will be cleared. To correct line containing a control-Z without clearing the screen, it is necessary to erase the entire line with control-U and type it again from the beginning.

IV THE INTELLIGENT KEYBOARD (IKB-1)

IV. THE INTELLIGENT KEYBOARD (IKB-1)

A. THEORY OF OPERATION

1. KEYBOARD HARDWARE

The IMSAI keyboard uses an on-board 8035 processor to handle all keyboard control functions. The keyboard control firmware is resident in the 4751 ROM at U7.

The control sequence is completely dependent on the firmware program, described in Section IV-A,2. The 8212 at U6 serves as an address latch during instruction fetch cycles from the 4751 ROM.

The two 74154, 4-line to 16-line decoders, at locations U1 and U2 allow the processor to scan the keyboard array and optional external keypad. Keyboard data is output to the parallel interface through the 8212 latch at U4.

a) Instruction Fetch

The 8035 processor executes those instructions stored in the 4751 PROM at U7. During an instruction fetch cycle, the 8035 outputs the 12 bit address of the next instruction to be executed on its Bus Lines (D0 - D7), and Port 2 (bits 0 - 3). The high order address bits A8 - A11 are internally latched and remain stable on the output lines of Port 2 (bits 0 - 3). The low order address bits are latched into the 8212 at U6 using the ALE strobe from the 8035 processor.

Once the address is present and stable at the address inputs of the 4751 ROM, the processor will drive its /PSEN strobe active low to enable the outputs of the 4751 ROM. Data is then input to the processor through its Bus lines D0 - D7 on the rising edge of /PSEN.

b) Keyboard Scan

The keyboard array consists of a diode matrix, 16 columns wide and 4 key positions (rows) to a column. To identify a key closure, the 8035 processor will output a column number to the 74154 at U1 through port P1, bits 0 - 3. The column number is decoded by the 74154 and will drive one of its 16 output lines active to place a low logic level along the selected column line of the keyboard array.

The 8035 processor then reads the state of the 4 row lines through port 2 bits 4 - 7. Any key closure in the selected column will appear as a "0" in one of the four bit positions (rows).

By successively scanning each column in this manner, the 8035 processor can key closures in the entire keyboard array.

If an external keypad is used, the same scanning technique is used with the 74154 at U2.

c) Display LED's

Each of the three display LED's are turned on by the 8035 processor by outputting one of three column select codes to the 74154 at U1. The corresponding output of the 74154 are used to drive the three LED's. Note that for a LED to appear lit to the user, it only has to be on for a brief interval since the period between successive "on" times cannot be detected by the user.

d) Speaker

An audible speaker beep is created by the 8035 processor by successively outputting a series of 1's and 0's to the speaker through output port one, bit 5. The pitch or frequency of the tone is determined by the rate of repetition.

e) Character Output

Characters to be output from the keyboard are latched into the 8212 at U4 from the 8035's Bus Lines on the rising edge of the processor's /WR strobe. The outputs of the 8212 directly interface to the external input port when the keyboard is operating in parallel mode. Parallel handshaking is achieved using the 8035 port 1, bits 6 and 7, as /ACK and /RDY. The 8035 outputs a "1" to bit 7 of port 1 when a character is ready to be input to the computer. Bit 6 of Port 1 is then polled to determine when the computer has read the character.

When the keyboard is used in the serial mode, the serial data appears on bit D0 of the 8035's Bus and is latched into the 8212 at U4. The DO0 output of the 8212 is then used to provide the TTL serial data line out to the interface. The DO0 line of the 8212 2ll also drives the base of Q1 so that EIA level serial data appears at the collector of Q1 (J1 pin 2).

f) Program Lines

When an output port is used to program the keyboard, the data appears at the T0 input of the 8035. T0 is a general purpose test input and in this case allows the 8035 to input the sequence of tests which comprises the command strings.

Handshaking is achieved by using the T1 input of the 8035 as a ready line from the interface. An acknowledge signal is generated with an output line from the 74154 at U2. The processor will place a 1FH on Port 1 bits 0 – 4, then, on the next cycle, will place a 00 H on these same bits. This causes U2 – 17 to toggle resulting in the desired acknowledge signal (J1 – 11).

2. KEYBOARD SOFTWARE

a) Initialization

On startup of the 8035, all interrupts are disabled and the chip is started at location 0. The program then clears most of RAM, initializes the timer, turns on the timer interrupt, and enters a wait loop which performs various functions, such as turning on the LED's, while waiting for a signal from the interrupt routine to initiate keyboard scans. I/O routines handling communication with the computer are called from the main loop.

b) Timer

The timer interrupt is based on a resettable 134.1 microsecond timer clock which is derived from the 3.58 MHz system crystal. Three processes are timed by the interrupt routine: (1) keyboard scans, which are initiated every 10 to 14 milliseconds after the interrupt routine sets the byte SCANFLAG; (2) serial output, which is handled directly by the interrupt routine; and (3) the loudspeaker beeps, which are also handled directly by the interrupt routine. The principal function of the interrupt routine is to time serial output accurately; for this reason, each interrupt begins with a delay loop (FUGLUP) which precisely times the resetting of the timer clock. The length of the delay is looked up in a table CKFUDGE which is indexed by the baud rate index RATE. Likewise, the timer is set to a value looked up in the table CKDIV, again indexed by RATE. This sets up the basic interrupt rate. Serial output is initiated by the HANDSHAKE routine by putting a data byte in RSERCHAR and setting RBITNUMBER to 11. The interrupt routine, upon seeing a non-zero value in RBITNUMBER, starts counting down from the main interrupt rate to the baud rate (by a value in CKBIT). Each time the counter goes to zero, the rightmost bit of RSERCHAR, shifted right, or a stop or start bit, is placed on the BUS outputs of the 8035, RBITNUMBER is decremented, and the clock divider is re-initialized.

c) Speaker

The loudspeaker beep is initialized from several places in the program by setting RBEEP to some nonzero value. Each time the initial timer routine sees a nonzero value of RBEEP, it complements bit 5 of P1 and decrements RBEEP. Since the basic timing rate varies for higher baud rates, the beep is higher in pitch when the baud rate is 1800 or 2400 baud, and slightly lower when for parallel output.

d) Main Loop

The main loop of the program alternately checks various conditions which turn the indicator lights on, and calls a series of routines which handle keyboard encoding and data communications. It should be noted that whenever a LED is flashed, the program enters a wait loop so that the light is long enough that it can be seen. This has an effect on the timing of non-interrupt driven processes, such as unencoded scans and RDY-ACK handshaking with the computer. The other processes in the main loop are HANDSHAKE, which initiates transmission of a byte to the computer, CHECKT1, which handles control (programming) information coming from the computer, CHECKPROG, which looks at the PROG/END-ENTRY keys to set or clear PROGFLAG, and CHECKSCAN, which will initiate a scan of the keyboard when the interrupt routine has set SCANFLAG.

e) Handshake

HANDSHAKE functions entirely differently in serial and parallel mode. In serial mode, it looks for the FULL flag, which is set by SCAN or DOUNENC to indicated that a byte to be transmitted is in BUFF. If FULL is set, it clears FULL, transfers the data byte from FUFF to RSERCHAR, and sets RBITNUMBER to 11 to signal the interrupt routine to start transmitting the character. In parallel mode, on the other hand, the routine places the data byte on the BUS, sets P1 bit 7, which is the RDY line going to the computer, and waits for P1 bit 6, the ACK line, to become true. When ACK becomes true, RDY is turned off. In the current routine, the situation is actually a little more complicated; after a timeout delay, RDY is turned off whether ACK appears or not. Also, an extra check for ACK going away immediately after RDY is turned off speeds up handshaking if the computer responds quickly or if RDY is jumpered to ACK.

f) CHECKT1

CHECKT1 checks for programming signals coming from the computer. The T1 input to the 8035 chip is used as a RDY input, T0 is used as a data bit, and an acknowledge is generated using one of the multiplexer outputs. The program works by counting 1's on the T0 line when T1 goes true. It exits without doing anything if too many (more than 3) 1's are seen in a row or if more than about 3 milliseconds elapse between 1's. A normal control message ends with 0. The meanings of the various legal sequences are defined in the programming page of the manual.

g) CHECKPROG

CHECKPROG sets PROGFLAG to 1 when the PROG key is depressed, if PROGFLAG is currently 0. The effect of this is to cause a branch from the scan routine to PROGRAMMIT, which uses input characters to setup other values for PROGFLAG. On subsequent entries to PROGRAMMIT, a 4-way branch on the value of PROGFLAG will be used to interpret subsequent characters in a command string. Note that at the point of entry to PROGRAMMIT, a character is freshly encoded, that is, shift, control and flag do not effect its value. If PROGFLAG is 4, the current mode is "enter character which will be substituted for some key", which has to be handled by a special routine later in SCAN. When a particular command sequence is completed, PROGFLAG is reset to 1 and the speaker is beeped. Note that some of the programming routines are also used by the external-programming routine CHECKT1, which calls REMOTESEL. In this case, if PROGFLAG is already 0, it is not set to 1.

h) CHECKSCAN

CHECKSCAN works by evoking the N-key rollover encoding routine SCAN whenever SCANFLAG is set and VERBFLAG is false. SCAN works as follows: A map of keys which are currently believed to be depressed is maintained in the array OLDKEYS. Scanning consists of reading in succession all 22 4-bit scan rows of the keyboard matrix (including the external keypad), packing the nibbles into 8-bit bytes, and comparing the input bytes with the corresponding entry in OLDKEYS. If a transition is seen, scanning stops, the row of the matrix, the bit number of the rightmost different bit, and whether the change is up or down are recorded. On the next scan, the resulting flags ONBIT and OFFBIT are checked. If the corresponding key is STILL down, or still up, respectively, the transition is considered debounced. In the case of a downward transition, the bit for the corresponding key is cleared (down keys are 0's) and encoding begins. In the case of an upward transition, the bit for the corresponding key is set in OLDKEYS. In this way, only transitions in the keyboard matrix initiate encoding and transmission of characters — any number of keys can be down, and a new depression will transmit a new character.

i) Encoding

Encoding is straightforward. An index to an ASCII table ASCTABLE on page 3 of the program is computed, and the basic ASCII code corresponding to the key is looked up. At this point PROGFLAG is checked to see if the character is to be input to the programming routine rather than to be transmitted. After that, if it is to be transmitted, the shift key and the shiftlock bit of MODE (set or cleared earlier) are checked to determine if the character is to be shifted. Likewise, the control key is checked if appropriate to determine if the character is to be turned into a control character. At this point, if PROGFLAG is equal to 4, the character is saved as a substitute character, and not transmitted. Otherwise, the character is placed in BUFF, FULL is set to tell HANDSHAKE to send the character, and REPCOUNT is set to -50 to time the delay to the first repeat of the character. Subsequent scans

check to see if the last character to be encoded has its corresponding key still down. If so, REPCOUNT is incremented; if it goes to 0, the character is re-transmitted and REPCOUNT set to -5. This results in a delay of about .8 second before the first repeat and about .08 seconds between subsequent repeats.

j) Unencoded Output

Unencoded output is handled as follows: When the CHECKSCAN routine sees that VERBFLAG is set, it calls the routine DOUNENC. This routine reads a row (4 bits in the left half of the input byte) of the keyboard matrix, packs the row number into the right half of the byte, and transmits the byte to the computer, incrementing the row number for the next call to DOUNENC. Note that (1) 1 means a key is UP, 0 means the key is DOWN, and (2) only the keys on the basic keyboard can be read in this mode — the external keypad is ignored, because only 16 rows can be encoded in this manner.

B. USER GUIDE

1. BOARD CONFIGURATION

Since the keyboard may be used as a serial or parallel device, it is important that the circuit board jumpers be properly configured for the type of hardware interface to be used. The jumper areas are located on the printed circuit board above the keyboard assembly. Each area is clearly marked on the board.

a) Parallel Port Configuration

If the keyboard is to be used as a parallel input device, the following jumper configuration instructions apply. Figure IV-1 shows the proper configuration when using the IMSAI MIO parallel port. Assembled units are shipped already jumpered for an MIO parallel port.

1) Install a jumper in each of the position JG, JH, JI, JK, JL, JM, JN, and JO, to set-up the eight parallel data lines.

2) If an IMSAI MIO parallel port is used, install a jumper at locations JE and JF to configure the handshake lines.

   When other types of parallel interface boards are used, it will be necessary to determine the type of handshake signals required. Refer to Appendix 1 for the configuration of the handshake lines before continuing.

3) Install a jumper in location JA from the bottom, center pad to the pad marked "P" (left).

4) Install a jumper in location JR from the center pad to the pad marked "+5V" (top).

5) If an IMSAI MIO parallel port is to be used, install jumpers in locations JB and JC.

   When other types of parallel interface boards are used and it is desired to allow the keyboard to be programmed from an output port, refer to Appendix 2 to configure the keyboard's control port lines before continuing.

6) Make certain that all jumpers are correctly configured and proceed with Section IV-B,2.

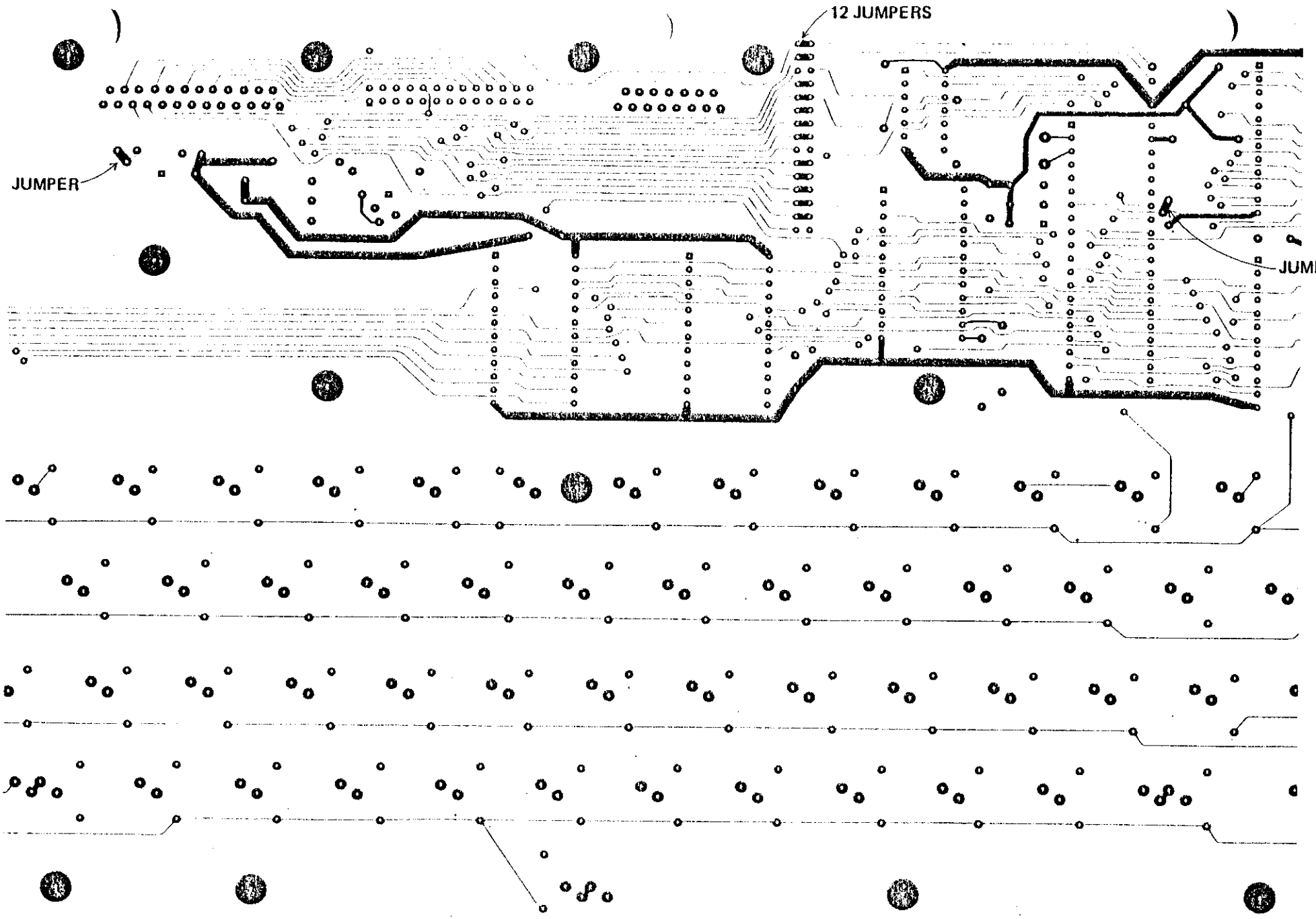


FIGURE IV - 1

b) Serial Port Configuration

If the keyboard is to be used as a serial input device, the following jumper configuration instructions apply. Figure IV-2 shows the proper configuration when using the IMSAI SIO board or the IMSAI MIO serial port.

1) Install a jumper in location JQ.

2) Install a jumper in location JR from the center pad to the pad marked "+5V" (top).

3) Install a jumper in location JA from the bottom, center pad to the pad marked "S" (right).

4) Install two jumpers as shown in Figure IV-3.

5) If the IMSAI MIO serial port is used, no other jumpers are required. Make certain all jumpers are correctly configured and proceed with Section IV-B,2. If the SIO is used, a jumper must be installed and a trace cut on the SIO as shown in Figure IV-4.

   If the TTL level serial output is to be used, install a jumper at location J1.

2. EXTERNAL INTERFACE CONNECTION

Once the circuit board has been properly configured for serial or parallel operation, the external interface connections can be made.

NOTE: Do not attach the keyboard to an interface for which it has not been configured or damage to the circuit may result.

a) IMSAI Parallel/Serial Interface (IMSAI MIO, SIO)

If an IMSAI MIO or SIO interface board is used, an IMSAI cable A and an IMSAI cable C will make interface connection straightforward.

1) Attach one 25 pin male end of Cable C to the 25 pin J1 connector at the rear of the keyboard cabinet.

2) Attach the remaining end of Cable C to the 25 pin connector of Cable A.

3) If the keyboard is configured for parallel operation, attach the remaining end of Cable A to the PIO1 or PIO2 connector on the MIO board to complete the parallel interface connection.

   If the keyboard is configured for serial operation, attach the remaining end of Cable A to the serial connector of the MIO or SIO board to complete the serial interface connection.

4) The keyboard is now ready to be used. Proceed with Section IV-B,2.

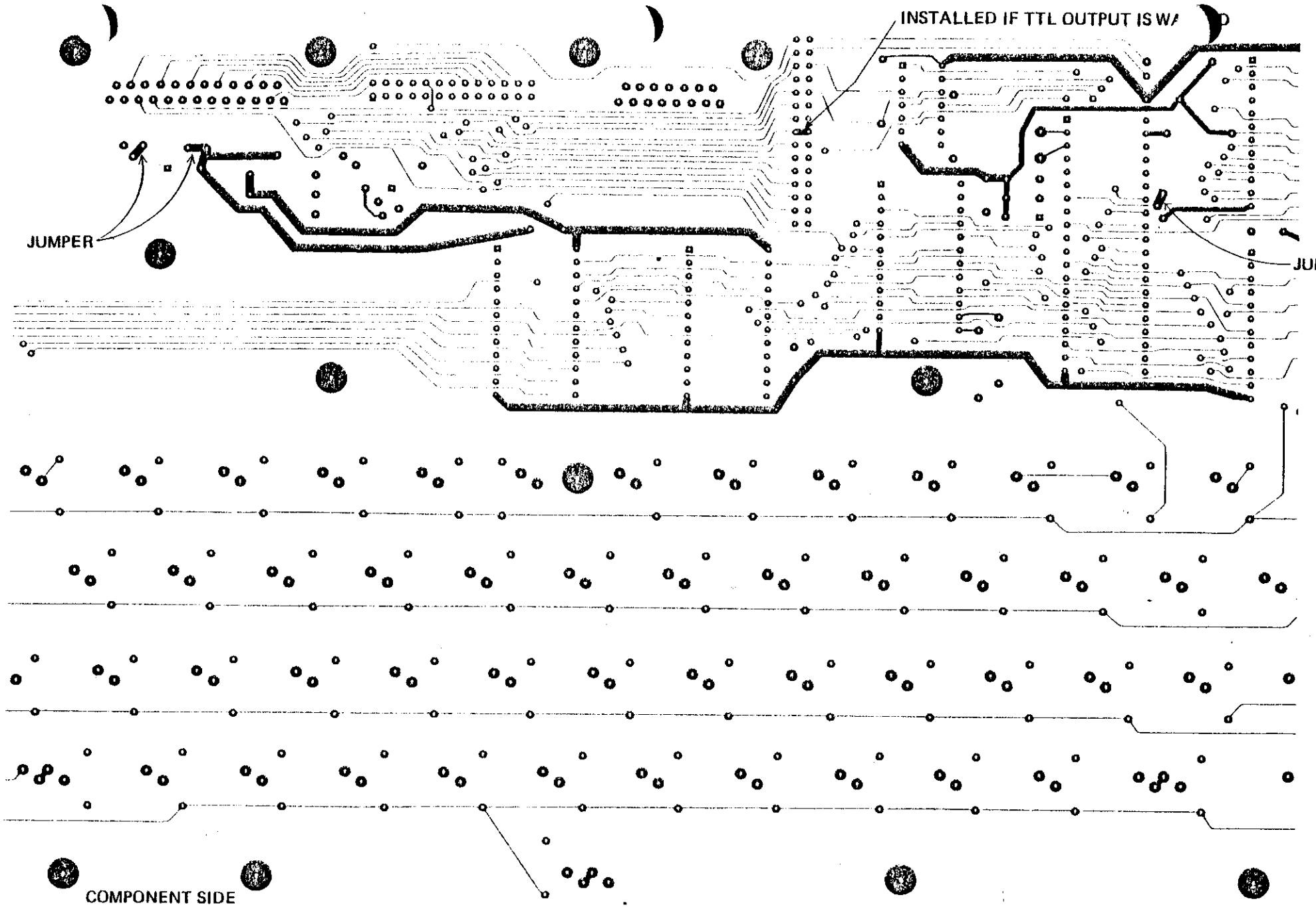


FIGURE IV - 2

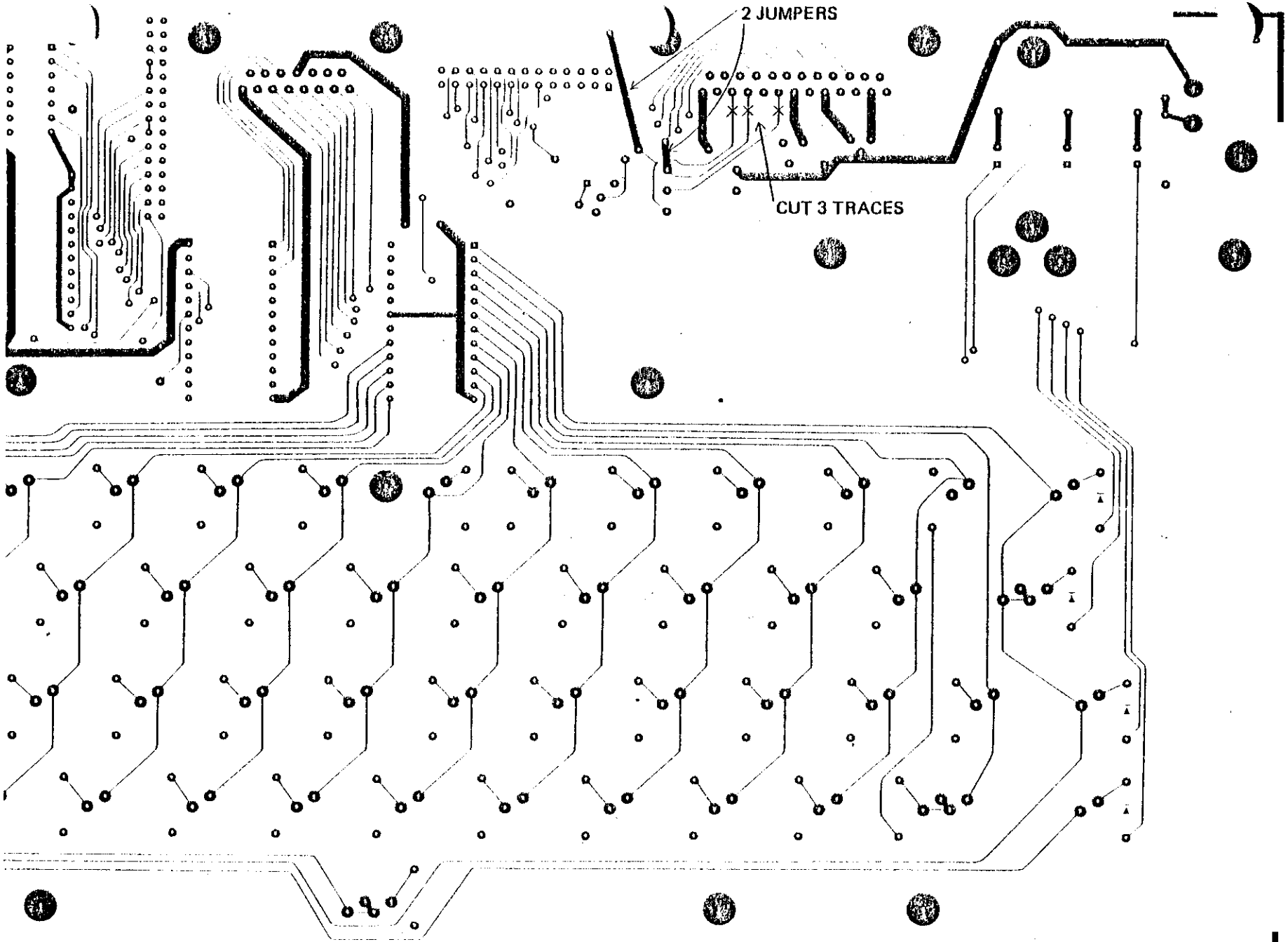


FIGURE IV - 3

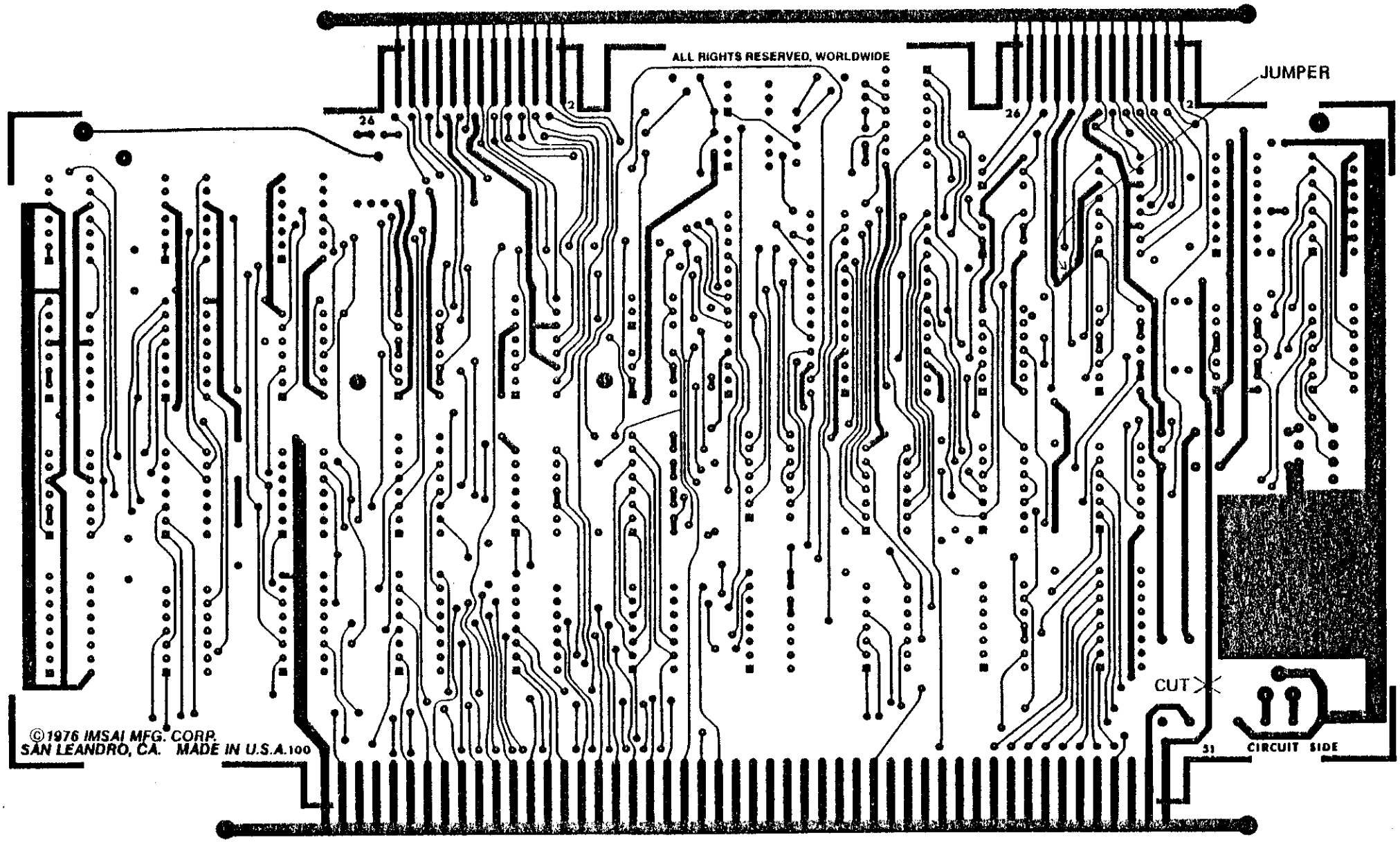

ALL RIGHTS RESERVED, WORLDWIDE
JUMPER
CUT
CIRCUIT SIDE
©1976 IMSAI MFG. CORP.
SAN LEANDRO, CA. MADE IN U.S.A.100

FIGURE IV - 4

b) Parallel Interface

If the IMSAI MIO parallel port is not used, the keyboard's parallel interface signals will have to be mapped to the appropriate lines of the interface board. Table IV-1 lists the signal lines used in the parallel mode. All signals are available at the 25 pin J1 connector at the rear of the keyboard cabinet.

TABLE IV-1
Parallel Mode Signals (J1 Connector)

Pin #	Signal Name	Description
22	DO7	Data bit 7
21	DO6	Data bit 6
20	DO5	Data bit 5
19	DO4	Data bit 4
18	DO3	Data bit 3
17	D02	Data bit 2
16	DO1	Data bit 1
15	DO0	Data bit 0
24	/RDY	Ready (active low)
23	/ACK	Acknowledge (active low)

c) Serial Interface (General)

If the IMSAI SIO or MIO serial port is not used, the keyboard's serial interface signals will have to be mapped to the appropriate lines of the interface board. Table IV-2 lists the signal lines used in the serial mode. All signals are available at the 25 pin J1 connector at the rear of the keyboard cabinet.

TABLE IV-2
Serial Mode Signals (J1 Connector)

Pin #	Signal Name	Description
16	DO1	TTL Data Out
2	EIA Data Out	Data Out EIA level
20	+5V Pullup	

3. KEYBOARD OPERATION

a) Data Entry Mode

Once the keyboard options have been selected and the END MODE key has been depressed, the keyboard will operate in the Data ENTRY MODE.

ENCODED: If the "E", encoded, option is selected, any key or sequence of keys typed on the keyboard or external keypad (optional) will cause the correspondign ASCII code to be sent to the computer.

In the encoded mode, continuous depression of a key will invoke the AUTO REPEAT function, causing the character to be output continuously until the key is released.

Selection of the "U", upper case only option, will cause the SHIFTLOCK LED to light.

UNENCODED: If the "V" unencoded option is selected, the keyboard will output a continuous series of bytes which will provide the user with a continuously updated map, indicating the state of the keyboard array. (The state of the external keypad is excluded from the map.)

Each byte which is output from the keyboard is in the following form:

D7 D6 D5 D4 D3 D2 D1 D0

Row State Row Number

The low order four bits contain a binary encoded row number (0 - 15 decimal), and the high order four bits indicate the state of that row.

A "0" in any of the bit positions D7 - D4 indicate that the corresponding key(s) of that row are depressed. Likewise, a "1" indicates that a key is in the up position.

Table IV-3 lists the keyboard array by row number and bit position (D7 - D4).

The keyboard will successively output the state of each row in the above format as long as the "V" unencoded mode is selected.

TABLE IV-3
Unencoded Mode

Row #	Row State D7	D6	D5	D4
0	RSHFT	CTL	TAB	ESC
1	FLAG1	SHFTLK	BRK	LSHFT
2	2	A	Q	!
3	R	S	W	"
4	C	D	E	#
5	V	F	R	$
6	B	G	t	%
7	N	HI	Y	&
8	M	J	U	'
9	<	K	I	(
10	>	L	O	)
11	?	+	P	0
12	=	†	§	.
13	SPACE	BS	?	LF
14		DEL	CR	
15		MODE		ENDENTRY

b) Program Mode (from Computer)

A number of the basic keyboard options may be programmed from an 8212-type parallel output port.

When the keyboard is configured for parallel operation with the IMSAI MIO parallel port, the keyboard may be programmed from the MIO's PIO1 or PIO2 output ports.

Programming is achieved by outputting a series of bytes to the keyboard. Only one bit per byte is taken as part of the command string. When an MIO parallel port is used, only output bit 0 is taken as the valid command bit.

A command string consists of from one to four bits and always terminates with a 0. Valid command strings for use with the MIO output port are (X's are "don't cares"):

XXXX XXX0	beep the speaker
XXXX XXX1 XXXX XXX0	place keyboard in default mode
XXXX XXX1 XXXX XXX1 XXXX XXX0	set lower case mode
XXXX XXX1 XXXX XXX1 XXXX XXX1 XXXX XXX0	select unencoded (verbatim mode)

Once a byte is output to the keyboard, the next byte of sequence must be output within 4 milliseconds, or the input routine will time-out and return, making no programming changes.

4. EXTERNAL KEYPAD

Provision is made for an external cursor control keypad and a numeric entry keypad. The scan lines for these keypads are brought out to connector J3. The connections for each key are shown in Table IV-4. To retain N-key rollover, it is necessary to place a diode in series with each switch with the cathode conecting to the scan line (/S10 through /S15). Therefore, each key will consist of a connection from one of the scan lines, /S10 through /S15, then to the cathode of a small signal diode. The anode is then connected to one side of the swithc and the other side of the switch connects to the appropriate bit line, bit 0 through bit 3.

TABLE IV-4

	Bit 0	Bit 1	Bit 2	Bit 3
/S10	9	8	7	^K=Cursor Up
/S11	6	5	4	^H=BS (Cursor Left)
/S12	3	2	1	^L=Cursor Right
/S13	0	Tab	^(home)	Line Feed (Cursor Down)
/S14	.	^U	^T	^E
/S15	,	^N(XMIT)	Del	^D

APPENDIX 1: PARALLEL HANDSHAKING

When the keyboard is configured for parallel operation, the /RDY and /ACK lines are used for handshaking.

The keyboard will assert /RDY (active low) when data is stable and is ready to be latched by the interface port. If it is desired to use /RDY, install a jumper in location JF. This timing is shown in Figure IV-5 (Case 1).

If it is desired to use an external acknowledge signal /ACK from the parallel input port, also install a jumper at location JD. In this configuration, once /RDY is asserted, the interface should respond by driving /ACK active low to indicate the reception of data.

This timing is shown in Figure IV-5 (Case 3). Note that if /ACK is not received within 11 ms following the leading edge of /RDY, /RDY will time-out and return to its cleared (high) state.

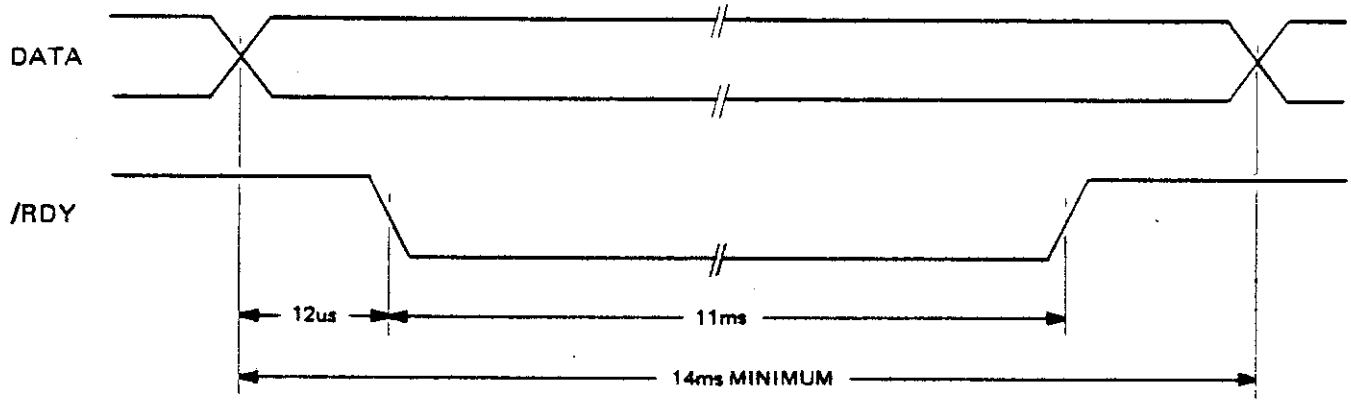
If it is desired to jumper the keyboard's /RDY output to the /ACK input, install a jumper at location JE. This configuration will result in a short /RDY pulse and does not require an external acknowledge signal from the interface. This timing is shown in Figure IV-5 (Case 2).

FIGURE IV - 5

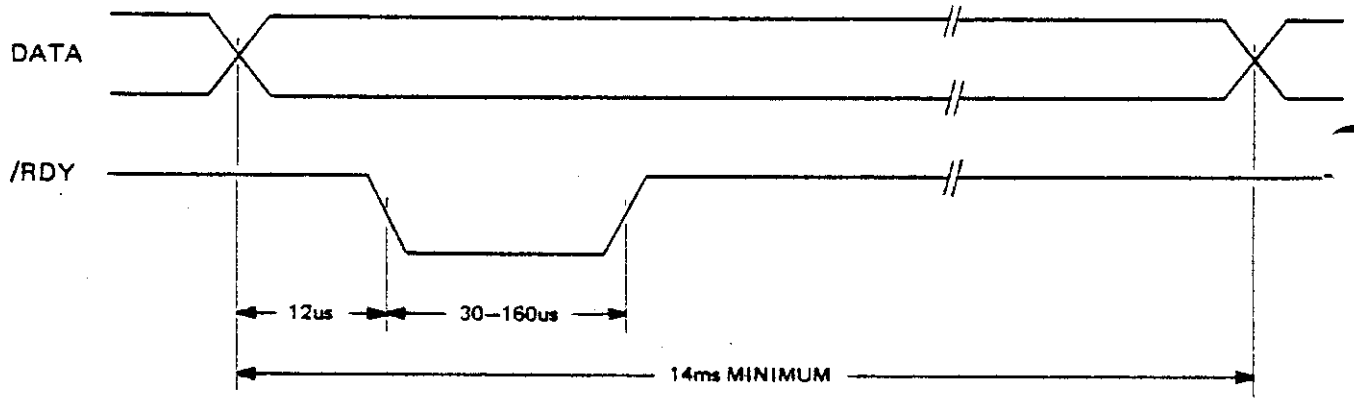
/RDY – /ACK TIMING

THE LEADING EDGE OF /RDY IS ALWAYS 12us AFTER DATA IS VALID

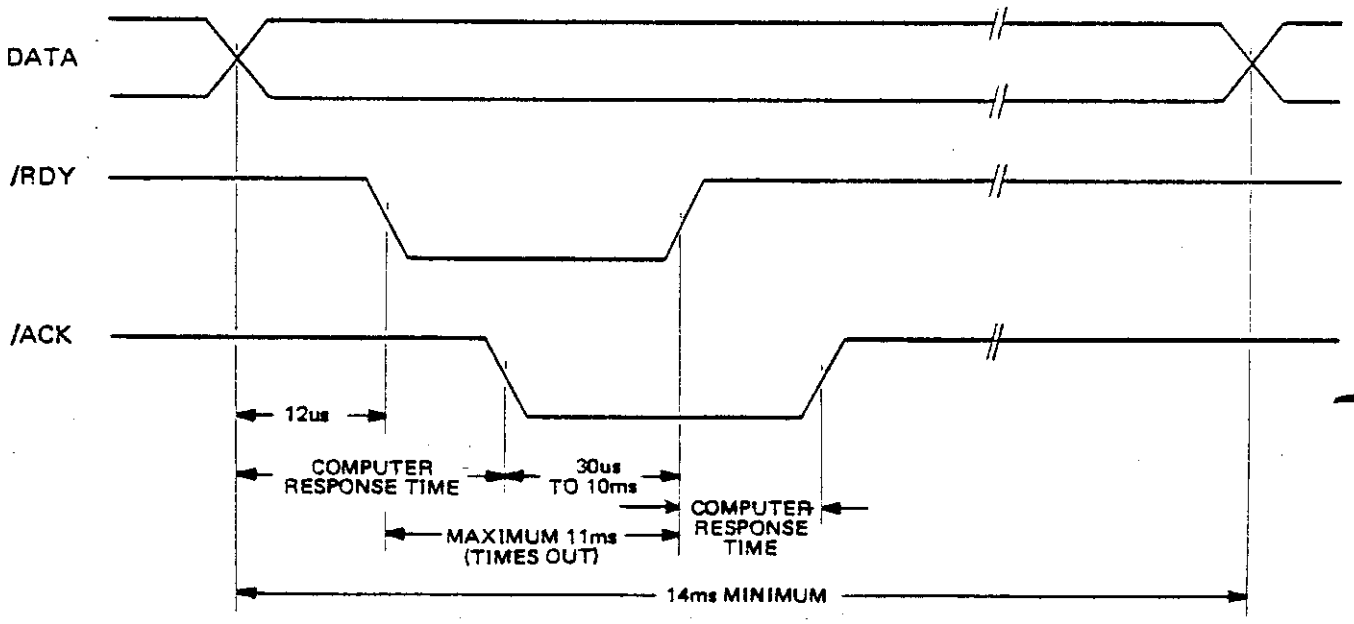
CASE 1: JUMPER F ONLY (RDY TIMES OUT)



CASE 2: /RDY JUMPERED TO /ACK (JUMPER "E") (SHORT /RDY PULSE)



CASE 3: EXTERNAL /ACK (JUMPER "D") . . . CLEARS /RDY



APPENDIX 2: CONTROL (OUTPUT) DATA AND HANDSHAKING

If a parallel output port is used to program the keyboard, the data and handshake lines need to be configured for the type of output port used.

A jumper should be installed in location JB to enable the use of the data input line (J1, pin 2). This line will be driven by one bit of the output port used for keyboard programming.

Jumpers JC and JP allow the READY and ACKNOWLEDGE handshake lines to be used.

When it is desired to use a READY output from the interface, install a jumper in location JC. The interface should assert READY (J1, pin 10) when data is stable and is ready to be READ by the keyboard processor.

When it is desired to use an ACKNOWLEDGE output from the keyboard, install a jumper in location JP. The keyboard will assert ACKNOWLEDGE (J1, pin 11) following the receipt of the READY signal.

The READY and ACKNOWLEDGE handshaking is suitable for use with an output port having hardware handshake line., e.g. 8212 type ports. The use of software implemented handshake lines is not recommended due to timing constraints.

V POWER SUPPLY (PS–28U)

V POWER SUPPLY (PS-28U)

A. THEORY OF OPERATION

The PS-28U is an unregulated power supply that provides the basic +8, +16 and -16 voltages for the PCS-80/30 system. It is comprised of four major component assemblies: line filter, transformer, rectifiers and filters.

LINE FILTERS: The line filter is a triple PI L-C filter designed to remove high frequency noise present on the AC line. This filter attenuates line noise above 1 MHz in frequency.

TRANSFORMER: The transformer is primarily designed for a number of AC input voltages: 92, 103.5, 115, 126.5, 184, 207, 230, and 253 VAC, 50/60 Hz, single phase input. The transformer secondary is connected as three series windings with a center tap. Four MR 1121 diodes full-wave rectify the +8 volts, while a full-wave bridge of four MR 501 diodes rectify the +16 and -16 volts.

FILTERING: The +16 and -16 volt supplies are each filtered by a 10K uF capacitor to ground, providing the +15 and -15 average volts at 4.0 amps. The +8 volts is filtered by two 95K uF capacitors to ground, providing 7.3 average volts at the 28 amp rated current.

.1uF capacitors high frequency bypass each voltage supply and bleeder resistors discharge the filter capacitors when power is turned off.