

TRS-80[®]
MODEL 4/4P
TECHNICAL
REFERENCE
MANUAL

CAT. NO. 26-2119

TRSDOS[®] Version 6.2.0 Operating System:
© 1984 Logical Systems.
Licensed to Tandy Corporation.
All Rights Reserved.

Model 4 4P Technical Reference Manual; Hardware Part:
© 1985 Tandy Corporation.
All Rights Reserved.

Model 4 4P Technical Reference Manual; Software Part:
© 1985 Tandy Corporation and Logical Systems.
All Rights Reserved.

Reproduction or use, without express written permission from Tandy Corporation of any portion of this manual is prohibited. While reasonable efforts have been taken in the preparation of this manual to assure its accuracy, Tandy Corporation assumes no liability resulting from any errors or omissions in this manual, or from the use of the information contained herein.

TRSDOS is a registered trademark of Tandy Corporation.

10 9 8 7 6 5 4 3 2 1

SECTION I

4 THEORY OF OPERATION



Part 1 / Hardware

SECTION I	1
1.1 Model 4 Theory of Operation	3
1.1.1 Introduction	3
1.1.2 CPU and Timing	3
1.1.3 Buffering	3
1.1.4 Address Decoding	3
1.1.5 ROM	7
1.1.6 RAM	7
1.1.7 Keyboard	7
1.1.8 Video	7
1.1.9 Real Time Clock	9
1.1.10 Cassette Circuitry	9
1.1.11 Printer Circuitry	9
1.1.12 I/O Connectors	9
1.1.13 Sound Option	10
1.2 Model 4 I/O BUS	13
1.3 Port Bits	16
SECTION II	19
2.1 Model 4 Gate Array Theory of Operation	21
2.1.1 Introduction	21
2.1.2 Reset Circuit	21
2.1.3 CPU	21
2.1.4 System Timing and Control Register	21
2.1.5 Address Decode	28
2.1.6 ROM	36
2.1.7 RAM	36
2.1.8 Video Circuit	36
2.1.9 Keyboard	41
2.1.10 Real Time Clock	41
2.1.11 Line Printer Port	41
2.1.12 Graphics Port	41
2.1.13 Sound Port	44
2.1.14 I/O Bus	44
2.1.15 Cassette Circuit	48
2.1.16 FDC Circuit	48
2.1.17 RS-232C Circuit	51
SECTION III	55
3.1 Model 4P Theory of Operation	57
3.1.1 Introduction	57
3.1.2 Reset Circuit	57
3.1.3 CPU	57
3.1.4 System Timing	57
3.1.5 Address Decode	60
3.1.6 ROM	60
3.1.7 RAM	71
3.1.8 Video Circuit	85
3.1.9 Keyboard	87
3.1.10 Real Time Clock	87
3.1.11 Line Printer Port	87
3.1.12 Graphics Port	91
3.1.13 Sound	91
3.1.14 I/O Bus Port	91
3.1.15 FDC Circuit	93
3.1.16 RS-232C Circuit	98

SECTION IV		101
4 2	4P Gate Array Theory of Operation	103
4 2 1	Introduction	103
4 2 2	Reset Circuit	103
4 2 3	CPU	103
4 2 4	System Timing	103
4 2 5	Address Decode	105
4 2 6	ROM	105
4 2 7	RAM	116
4 2 8	Video Circuit	130
4 2 9	Keyboard	132
4 2 10	Real Time Clock	132
4 2 11	Line Printer Port	132
4 2 12	Graphics Port	136
4 2 13	Sound	136
4 2 14	I/O Bus Port	136
4 2 15	FDC Circuit	138
4 2 16	RS 232C Circuit	142
SECTION V	Chip Specifications	147
INDEX		

1.1 MODEL 4 THEORY OF OPERATION

1.1.1 Introduction

The TRS 80 Model 4 Microcomputer is a self contained desktop microcomputer designed not only to be completely software compatible with the TRS 80 Model III, but to provide many enhancements and features. System distinctions which enable the Model 4 to be Model III compatible include a Z80 CPU capable of running at a 4 MHz clock rate, BASIC operating system in ROM (14K), memory mapped keyboard, 64 character by 16 line memory mapped video display, up to 128K Random Access Memory, cassette circuitry able to operate at 500 or 1500 baud, and the ability to accept a variety of options. These options include one to four 5 1/4 inch double density floppy disk drives, one to four five megabyte hard disk drives, an RS 232 Serial Communications Interface, and a 640 by 240 pixel high resolution graphics board.

1.1.2 CPU and Timing

The central processing unit of the Model 4 microcomputer is the Z80 A microprocessor — capable of running at either a two (2 02752) or four (4 05504) MHz clock rate. The main CPU timing comes from the 20 MHz (20 2752 MHz) crystal controlled oscillator, Y1 and Q1. There is an additional 12 MHz (12 672 MHz) oscillator, Y2 and Q2, which is necessary for the 80 by 24 mode of video operation. The oscillator outputs are sent to two Programmable Array Logic (PAL) circuits, U3 and U4, for frequency division and routing of appropriate timing signals.

PAL U3 divides the 20 MHz signal by five for 4 MHz CPU operation, by ten for a 2 MHz rate, and slows the 4 MHz clock for the M1 Cycle (See Figure 1-3). U3 also divides the master clock by four to obtain a 5 MHz clock to be sent to the RS-232 option connector as a reference for the baud rate generator. PAL U4 selects an appropriate 10 MHz or 12 MHz clock for the video shift clock, and using divider U5 provides additional timing signals to the video display circuitry (See Fig 1-4).

Hex latch U18 is clocked from the 20 MHz clock, and is used to provide MUX and CAS timing for the dynamic

memory circuits. Also, with additional gates from U16, U19, U20, U31, and U32, this chip provides the wait circuitry necessary to prevent the CPU from accessing video RAM during the active portion of the display. This is done by latching the data for the video RAM and simultaneously forcing the Z80 CPU into a "WAIT" state and is necessary to eliminate undesirable "hashing" of the video display (See Fig 1-4).

1.1.3 Buffering

Low level signals from and to the CPU need to be buffered, or current amplified in order to drive many other circuits. The 16 address lines are buffered by U55 and U56, which are unidirectional buffers that are permanently enabled. The eight data lines are buffered by U71. Since data must flow both to and from the CPU, U71 is a bi directional buffer which can go into a three state condition when not in use. Both direction and enable controls come from the address decoding section.

The clock signal to the CPU (from PAL U3) is buffered by active pullup circuit Q3. RESET and WAIT inputs to the CPU are buffered by U17 and U46. Control outputs from the Z80 (M1*, RD*, WR*, MREQ*, and IORQ*) are sent to PAL U58, which combines these into other appropriate control signals consistent with Model 4's architecture. Other than MREQ*, which is buffered by part of U38, the raw control signals go to no other components, and hence require no additional buffering.

1.1.4 Address Decoding

The address decoding section is divided into two sub sections. Port address decoding and Memory address decoding.

In port address decoding, low order address lines (some combined through a portion of U32) are sent to the address and enable inputs of U48, U49, and U50. U48 is also enabled by the IN* signal, which means that it decodes port input signals, while U49 decodes port output signals. A table of the resulting port map is shown below.

Port Addr. (Hex)	Read Function	Write Function
FC FF	Cassette In, Mode Read	Cassette Out, resets cassette data latch
F8 FB	Read Printer Status	Output to Printer
(1) F4 F7	reserved	Drive Select latch
(1) F3	FDC Data Reg	FDC Data Reg
(1) F2	FDC Sector Reg	FDC Sector Reg
(1) F1	FDC Track Reg	FDC Track Reg.

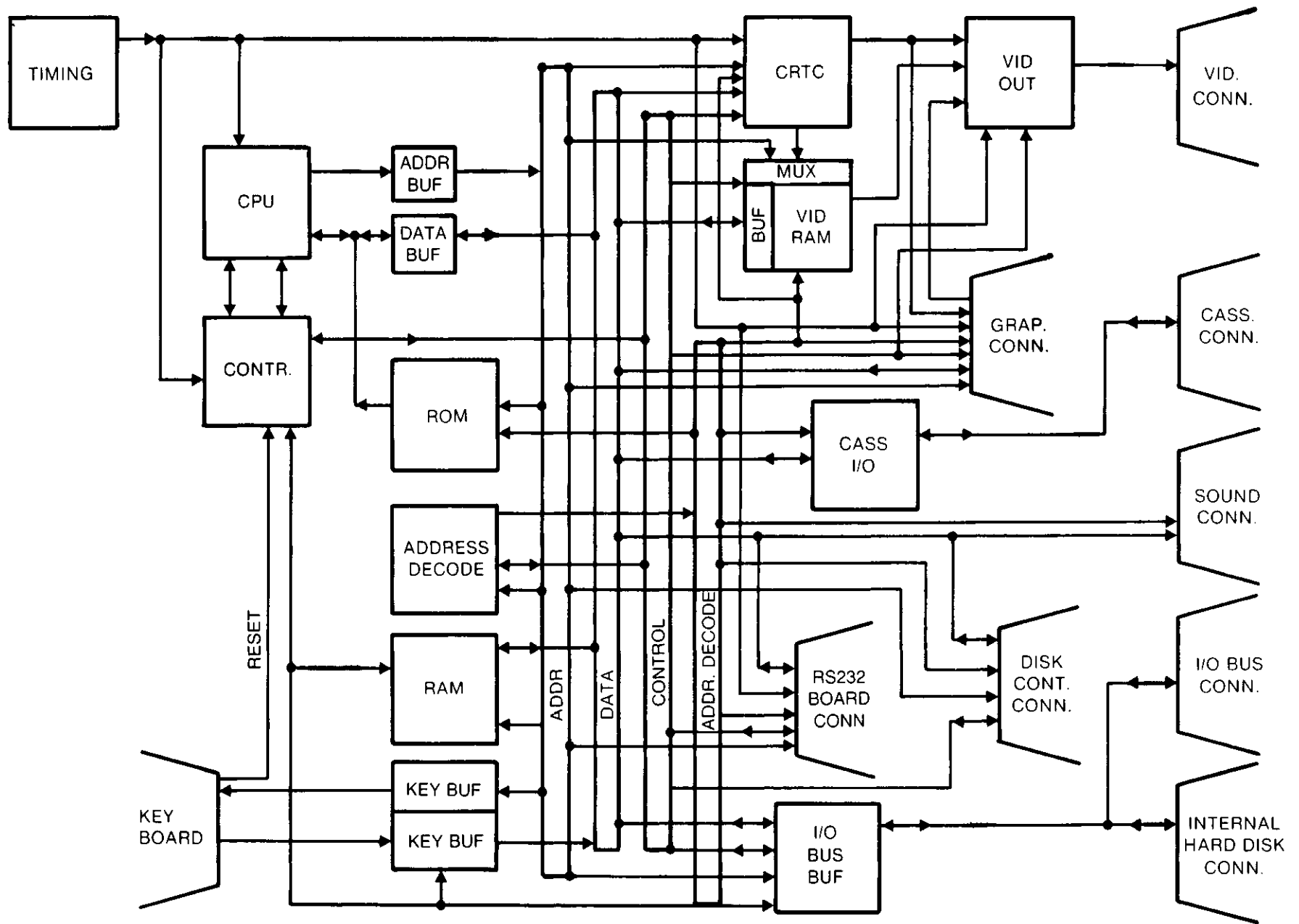


FIGURE 1-1. MODEL 4 BLOCK DIAGRAM

(1) F0	FDC Status Reg.	FDC Command Reg.
EC - EF	Resets RTC Int.	Mode Output latch
(2) EB	Rcvr Holding Reg.	Xmit Holding Reg.
(2) EA	UART Status Reg.	UART/Modem control
(2) E9	- reserved -	Baud Rate Register
(2) E8	Modem Status	Master Reset/Enable
E4 - E7	Read NMI Status	UART control reg.
E0 - E3	Read INT Status	Write NMI Mask reg.
(3) CF	HD Status	Write INT Mask reg.
(3) CE	HD Size/Drv/Hd	HD Command
(3) CD	HD Cylinder high	HD Size/Drv/Hd
(3) CC	HD Cylinder low	HD Cylinder high
(3) CB	HD Sector Number	HD Cylinder low
(3) CA	HD Sector Count	HD Sector Number
(3) C9	HD Error Reg.	HD Sector Count
(3) C8	HD Data Reg.	HD Write Precomp.
(3) C7	HD CTC channel 3	HD Data Reg.
(3) C6	HD CTC channel 2	HD CTC channel 3
(3) C5	HD CTC channel 1	HD CTC channel 2
(3) C4	HD CTC channel 0	HD CTC channel 1
(3) C2 - C3	HD Device ID Reg.	HD CTC channel 0
(3) C1	HD Control Reg.	- reserved -
(3) C0	HD Wr. Prot. Reg.	HD Control Reg.
94 - 9F	- reserved -	- reserved -
(4) 90 - 93	- reserved -	Sound Option
(5) 8C - 8F	Graphics Sel. 2	Graphics Sel. 2
8B	CRTC Data Reg.	CRTC Data Reg.
8A	CRTC Control Reg.	CRTC Control Reg.
89	CRTC Data Reg.	CRTC Data Reg.
88	CRTC Control Reg.	CRTC Control Reg.
84 - 87	- reserved -	Options Register
(5) 83	- reserved -	Gra. X Reg. Write
(5) 82	- reserved -	Gra. Y Reg. Write
(5) 81	Graphics Ram Rd.	Graphics Ram Wr.
(5) 80	- reserved -	Gra. Options Reg. Wr

Notes: (1) Valid only if FDC option is installed
(2) Valid only if RS-232 option is installed
(3) Valid only if Hard Disk option is installed
(4) Valid only if sound option is installed
(5) Valid only if High Resolution Graphics option is installed

Following is a Bit Map of the appropriate ports in the Model 4. Note that this is an "internal" bit map only. For bit maps of the optional devices, refer to the appropriate section of the desired manual.

Model 4 Port Bit Map

Port	D7	D6	D5	D4	D3	D2	D1	D0
FC - FF	Cass							Cassette
(READ)	data 500 bd			(M I R R O R o f P O R T E C)				data 1500 bd
FC - FF			(Note, also resets cassette data latch)				cas.	cassette
(WRITE)	x	x	x	x	x	x	out	data out
F8 - FB	Prntr	Prntr	Prntr	Prntr	x	x	x	x
(READ)	BUSY	Paper	Select	Fault	x	x	x	x
F8 - FB	Prntr	Prntr	Prntr	Prntr	Prntr	Prntr	Prntr	Prntr
(WRITE)	D7	D6	D5	D4	D3	D2	D1	D0
EC - EF			(Any Read causes reset of Real Time Clock Interrupt)					
EC - EF	x	CPU	x	Enable	Enable	Mode	Cass	x
(WRITE)	x	Fast	x	EX I/O	Altset	Select	Mot On	x
E0 - E3	x	Receive	Receive	Xmit	10 Bus	RTC	C Fall	C Rise
(READ)	x	Error	Data	Empty	Int	Int	Int	Int
E0 - E3	x	Enable	Enable	Enable	Enable	Enable	Enable	Enable
(WRITE)	x	Rec Err	Rec Data	Xmit Emp	10 Int	RT Int	CF Int	CR Int
90 - 93	x	x	x	x	x	x	x	Sound
(WRITE)	x	x	x	x	x	x	x	Bit
84 - 87	Page	Fix Uptr	Memory	Memory	Invert	80/64	Select	Select
(WRITE)		Memory	Bit 1	Bit 0	Video		Bit 1	Bit 0

Memory mapping is accomplished by PAL U59 in the Basic 16K or 64K computer. In a 128K system, PAL U72, along with the select and memory bits of the options register, also enter into the memory mapping function.

Four memory maps are listed below. Memory Map I is compatible with the Model III. Note that there are two 32K banks in the 64K system, which can be interchanged with either position of the upper two banks of a 128K system. The 128K system has four moveable 32K banks. Also note, in the Model III mode, that decoding for the printer status read (37E8 and 37E9 hexadecimal) is accomplished by U93 and leftover gates from U40, U46, U51, U54, U60, and U62.

Memory Map I - Model III Mode

	0000 - 1FFF	ROM A (8K)
	2000 - 2FFF	ROM B (4K)
	3000 - 37FF	ROM C (2K) - Less 37E8 - 37E9
	37E8 - 37E9	Printer Status Port
	3800 - 3BFF	Keyboard
	3C00 - 3FFF	Video RAM (Page bit selects 1K of 2K)
*	4000 - 7FFF	RAM (16K system)
*	4000 - FFFF	RAM (64K system)

Memory Map II

0000 – 37FF	RAM (14K)	
3800 – 3BFF	Keyboard	
3C00 – 3FFF	Video RAM	
4000 – 7FFF	RAM (16K)	End of one 32K Bank
8000 – FFFF	RAM (32K)	Second 32K Bank

Memory Map III

0000 – 7FFFF	RAM (32K)	End of One 32K Bank
8000 – F3FF	RAM (29K)	Second 32K Bank
F400 – F7FF	Keyboard	
F800 – FFFF	Video RAM	

Memory Map IV

0000 – 7FFF	RAM (32K)	One 32K Bank
8000 – FFFF	RAM (32K)	Second 32K Bank

(See Figure 1-2 for 128K Maps)

1.1.5 ROM

The Model 4 Microcomputer contains 14K of Read Only Memory (ROM), which is divided into an 8K ROM (U68), a 4K ROM (U69), and a 2K ROM (U70). ROMs used have three-state outputs which are disabled if the ROMs are deselected. As a result, ROM data outputs are connected directly to the CPU data bus and do not use data buffer U71, which is disabled during a ROM access.

ROMs are Model III compatible and contain a BASIC operating system, as well as a floppy disk boot routine. The enable inputs to the ROMs are provided by the address decoding section, and are present only in the Model III mode of operation.

1.1.6 RAM

Three configurations of Random Access Memory are available on the Model 4: 16K, 64K, and 128K. The 16K option uses 4116 type, 16K by 1 dynamic RAMs, which require three supply voltages (+12 volts, +5 volts, and -5 volts). The 64K and 128K options use 6665 type, 64K by 1 dynamic RAMs, which require only a single supply voltage (+5 volts). The proper voltage for each option is provided by jumpers.

Dynamic RAMs require multiplexed incoming address lines. This is accomplished by ICs U63 and U76. Output data from RAMs is buffered by U64. With the 128K option, there are two rows of the 64K by 1 RAM ICs. The proper row is selected by the CAS* signal from PAL U72.

1.1.7 Keyboard

The Model 4 Keyboard is a 70-key sculptured keyboard, scanned by the microprocessor. Each key is identified by its column and row position. Columns are defined by address lines A0 - A7, which are buffered by open-collector drivers U29 and U30. Data lines D0 - D7 define the rows and are buffered by CMOS buffers U44 and U45. Row inputs to the buffers are pulled up by resistor pack RP 1, unless a key in the current column being scanned is depressed. Then, the row for that key goes low.

1.1.8 Video

The heart of the video display circuitry in the Model 4 is the 68045 Cathode Ray Tube Controller. The CRTC allows two screen formats: 64 by 16 and 80 by 24. Since the 80 by 24 screen requires 1,920 screen memory locations, a 2K by 8 static RAM is used for the Video RAM. The 64 by 16 mode has a two-page screen display and a bit in the options register for determining which page is active for the CPU. Offset the start address of the CRTC to gain access to the second page in the 64 by 16 mode.

Addresses to the video RAM are provided by the 68045 when refreshing the screen and by the CPU when updating the data. These two sets of addresses are multiplexed by U33, U34, and U35. Data between the CPU and Video RAM is latched by U6 for a write, and buffered by U7 for a read operation.

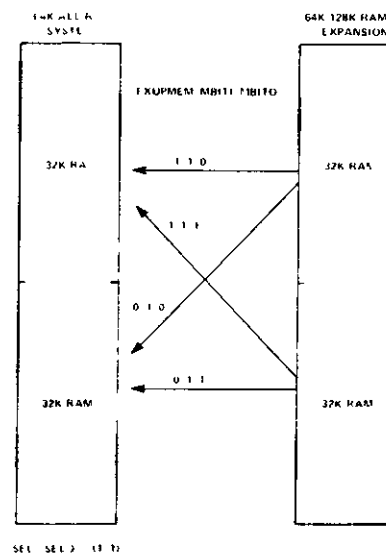
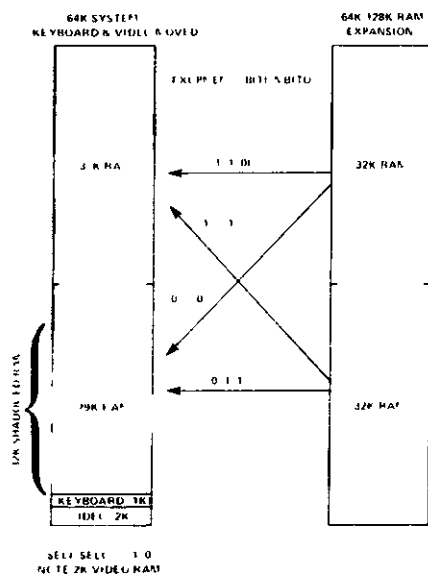
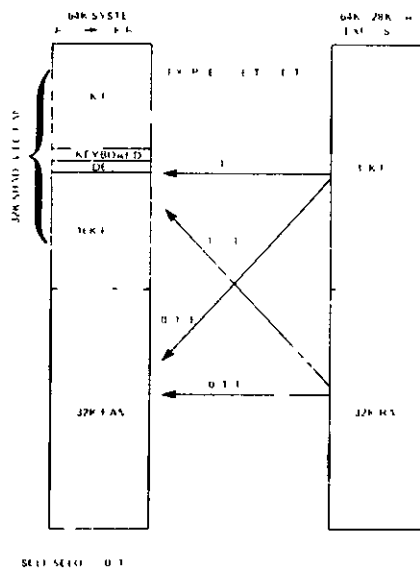
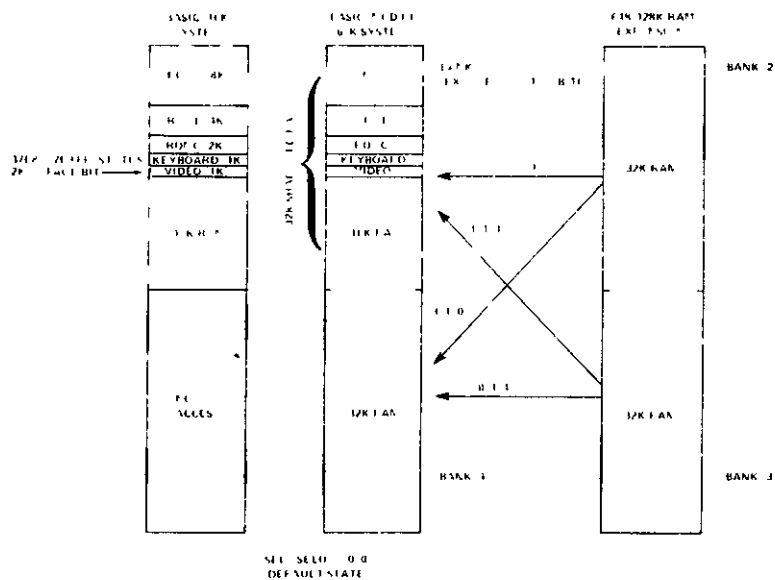


FIGURE 1-2. RAM MEMORY

During screen refresh, the data outputs of the Video RAM (ASCII character codes) are latched by U8 and become the addresses for the character generator ROM (U23). In cases of low resolution graphics, a dual 1 of 4 data selector (U9) is the cell generator with additional buffering from U10.

The shift register U11 inputs are the latched data outputs of the character or cell generator. The shift clock input comes from the PAL U4, and is 10.1376 MHz for the 64 by 16 mode and 12.672 MHz for 80 by 24 operation. The serial output from the shift register later becomes actual video dot information.

Special timing in the video circuit is handled by hex latch U2. This includes blanking (originating from CRTC) and shift register loading (originating from U4). Additional video control and timing functions, such as sync buffering, inversion selection, dot clock chopping, and graphics disable of normal video, are handled by miscellaneous gates in U12, U13, U14, U22, U24, and U26.

1.1.9 Real Time Clock

The Real Time Clock circuit in the Model 4 provides a 30 Hz (in the 2 MHz CPU Mode) or 60 Hz (in the 4 MHz CPU Mode) interrupt to the CPU. By counting the number of interrupts that have occurred, the CPU can keep track of the time. The 60 Hz vertical sync signal from the video circuitry is divided by two (2 MHz Mode) by U53, and the 30 Hz at pin 1 of U51 is used to generate the interrupts. In the 4 MHz mode, signal FAST places a logic low at pin 1 of U51, causing signal VSYNC to trigger the interrupts at the 60 Hz rate. Note that any time interrupts are disabled, the accuracy of the clock suffers.

1.1.10 Cassette Circuitry

The cassette write circuitry latches the two LSBs (D0 and D1) for any output to port FF (hex). The outputs of these latches (U27) are then resistor summed to provide three discrete voltage levels (500 Baud only). The firmware toggles the bits to provide an output signal of the desired frequency at the summing node.

There are two types of cassette Read circuits – 500 baud and 1500 baud. The 500 baud circuit is compatible with both Model 1 and III. The input signal is amplified and filtered by Op amps (U43 and U28). Part of U15 then forms a Zero Crossing Detector, the output of which sets the latch U40. A read of Port FF enables buffer U41, which allows the CPU to determine whether the latch has been set, and simultaneously resets the latch. The firmware determines by the timing between settings of the latch whether a logic "one" or "zero" was read in from the tape.

The 1500 baud cassette read circuit is compatible with the Model III cassette system. The incoming signal is compared to a threshold by part of U15. U15's output will then be either high or low and clock about one half of U39, depending on whether it is a rising edge or a falling edge. If interrupts are enabled, the setting of either latch will generate an interrupt. As in the 500 baud circuit, the firmware decodes the interrupts into the appropriate data.

For any cassette read or write operation, the cassette relay must be closed in order to start the motor of the cassette deck. A write to port EC hex with bit one set will set latch U42, which turns on transistor Q4 and energizes the relay K1. A subsequent write to this port with bit one clear will clear the latch and de-energize the relay.

1.1.11 Printer Circuitry

The printer status lines are read by the CPU by enabling buffer U67. This buffer will be enabled for any input from port F8 or F9, or any memory read from location 37E8 or 37E9 when in the Model III mode. For a listing of bit status, refer to the bit map.

After the printer driver software determines that the printer is ready to receive another character (by reading the status), the character to be printed is output to port F8. This latches the character into U66, and simultaneously fires the one shot U65 to provide the appropriate strobe to the printer.

1.1.12 I/O Connectors

Two 20 pin single inline connectors, J7 and J8, are provided for the connection of a Floppy Disk Controller and an RS 232 Communications Interface, respectively. All eight data lines and the two least significant address lines are routed to these connectors. In addition, connections are provided for device or board selection, interrupt enable, interrupt status read, interrupt acknowledge, RESET, and the CPU WAIT signal.

The graphics connector, J10, contains all of the above interface signals, plus CRTCLK, the dotclock signal, a graphics enable input, and other timing clocks which synchronize the graphics board with the CRTC.

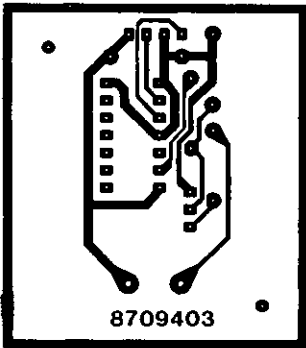
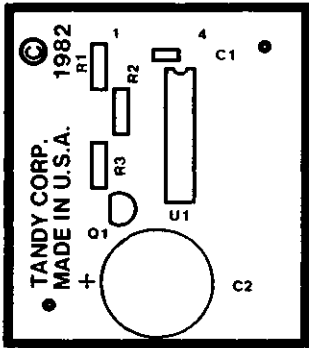
The I/O bus connector, J2, contains connections for all eight data lines (buffered by U74), the low order address lines (buffered by U73), and the control lines (buffered by U75) IN*, OUT*, RESET*, M1*, and IORQ*. In addition, the I/O bus connector has inputs to allow the device(s), connected to generate CPU WAIT states and interrupts.

The sound connector, J11, contains only four connections: sound enable (any output to port 90 hex), data bit D0, Vcc, and ground.

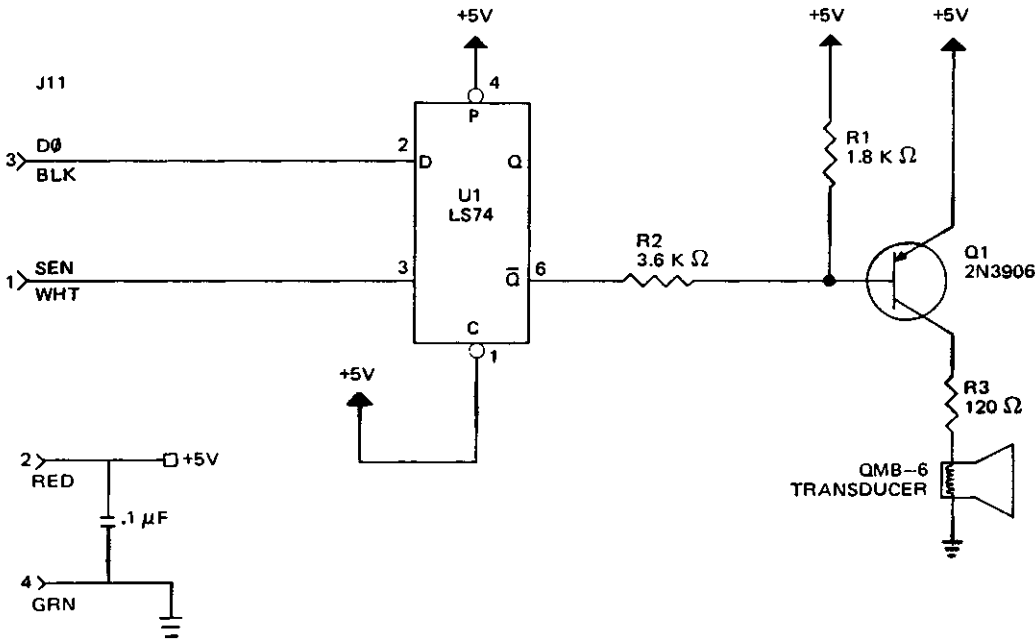
1.1.13 Sound Option

The Model 4 sound option, available as standard equipment on the disk drive versions, is a software intensive device. Data

is sent out to port 90H, alternately setting and clearing data bit D0. The state of this bit is latched by sound board U1 and amplified by sound board Q1, which drives a pie-zoelectric sound transducer. The speed of the software loop determines the frequency, and thus, the pitch of the resulting tone.



COMPONENT LOCATION/CIRCUIT TRACE, SOUND BOARD #8858121



SCHEMATIC 8000188, SOUND BOARD #8858121

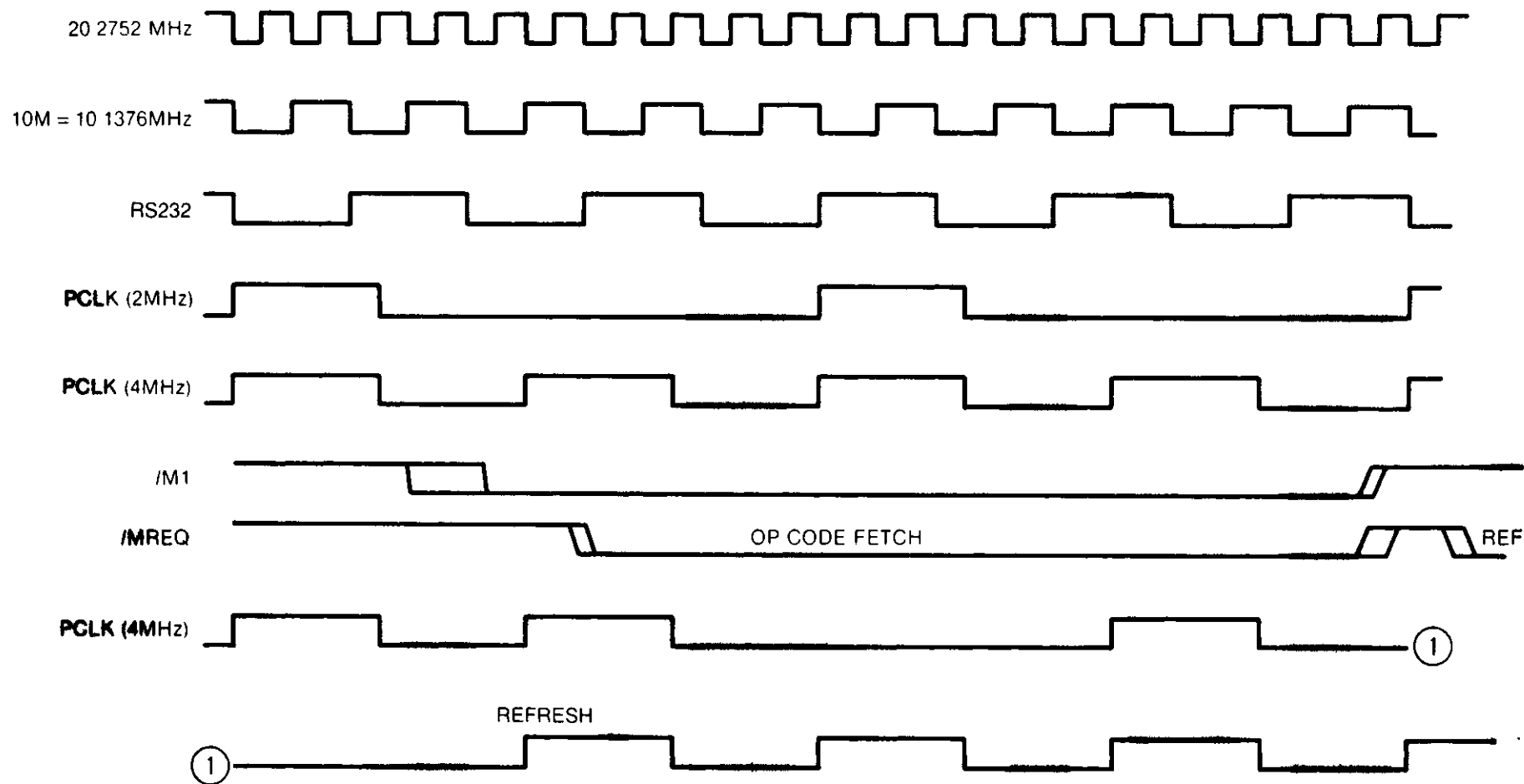


FIGURE 1-3. TIMING OF U3 & CPU

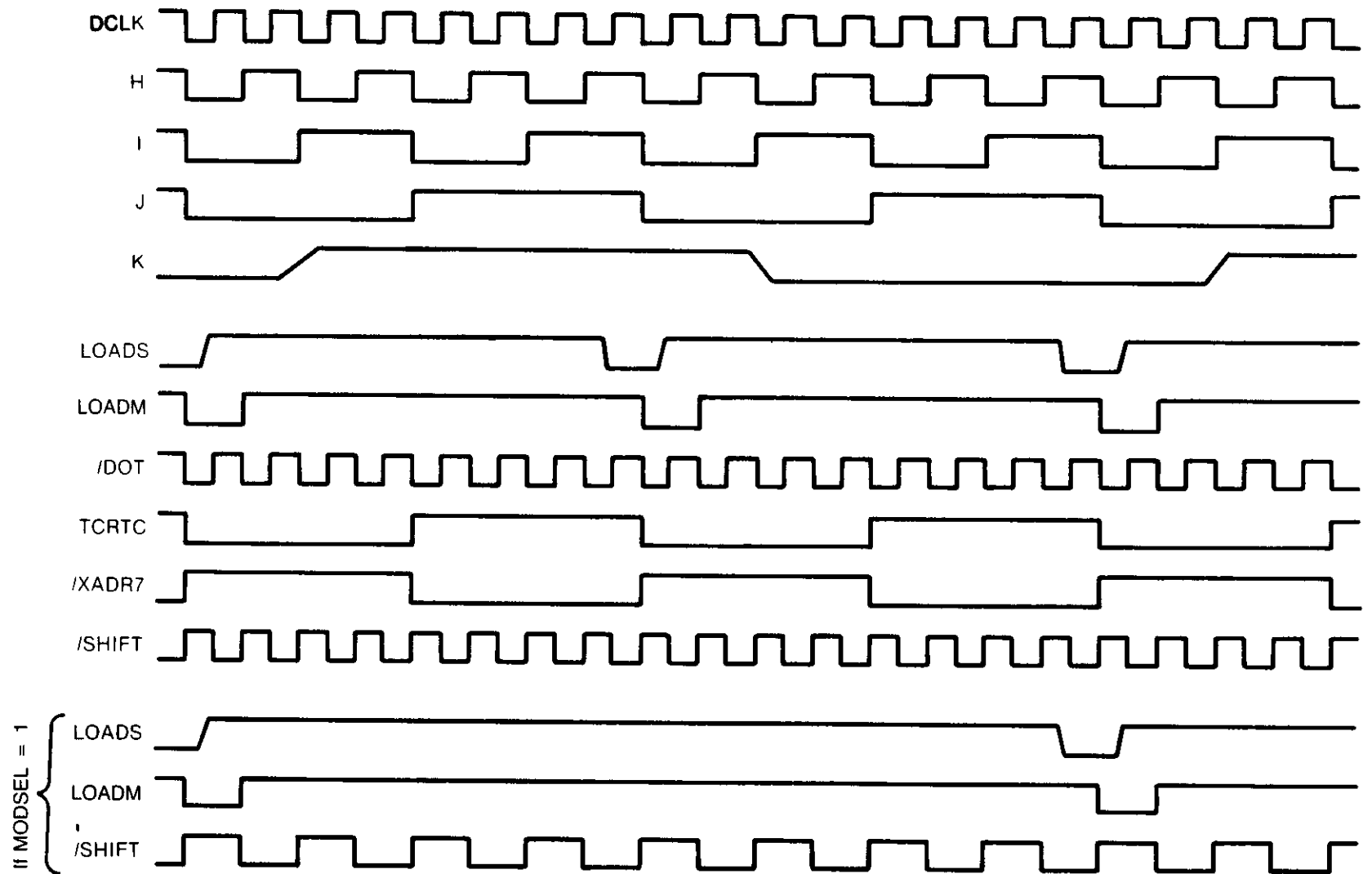


FIGURE 1-4. TIMING OF U4

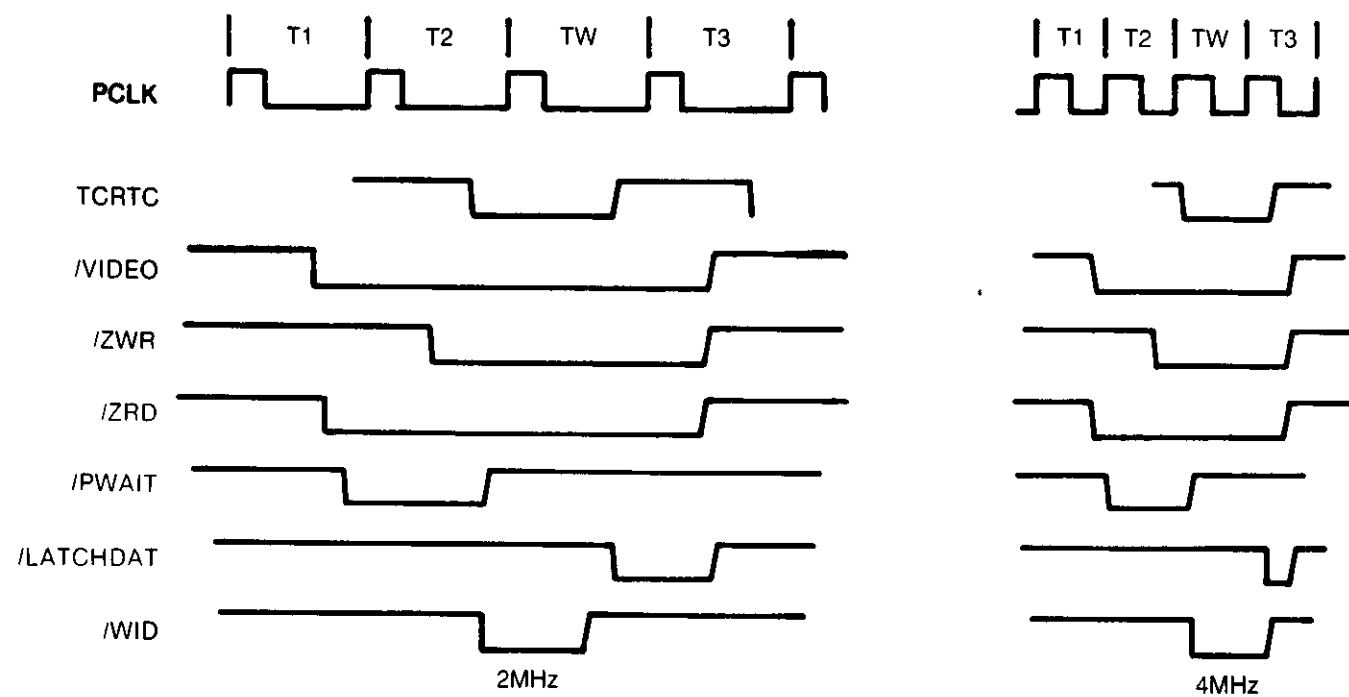


FIGURE 1-5. CPU VIDEO ACCESS TIMING

1.2 MODEL 4 I/O BUS

The Model 4 Bus is designed to allow easy and convenient interfacing of I/O devices to the Model 4. The I/O Bus supports all the signals necessary to implement a device compatible with the Z 80's I/O structure. That is:

Addresses

A0 to A7 allow selection of up to 256[†] input and 256 output devices if external I/O is enabled.

[†]Ports 80H to 0FFH are reserved for System use.

Data

DB0 to DB7 allow transfer of 8 bit data onto the processor data bus if external I/O is enabled.

Control Lines

- a IN* — Z 80 signal specifying that an input is in progress. Gated with IORQ.
- b OUT* — Z 80 signal specifying that an output is in progress. Gated with IORQ.
- c RESET* — system reset signal.
- d IOBUSINT* — input to the CPU signaling an interrupt from an I/O Bus device if I/O Bus interrupts are enabled.
- e IOBSWAIT* — input to the CPU wait line allowing I/O Bus device to force wait states on the Z 80 if external I/O is enabled.
- f EXTIOSEL* — input to CPU which switches the I/O Bus data bus transceiver and allows an INPUT instruction to read I/O Bus data.
- g M1* — and IORQ* — standard Z 80 signals.

The address line, data line, and control lines a to c and e to g are enabled only when the ENEXIO bit in EC is set to a one.

To enable I/O interrupts the ENIOBUSINT bit in the CPU IOPORT E0 (output port) must be a one. However, even if it is disabled from generating interrupts the status of the IOBUSINT* line can still read on the appropriate bit of CPU IOPORT E0 (input port).

See Model 4 Port Bit assignment for port 0FF 0EC and 0E0 on pages 14 and 15.

The Model 4 CPU board is fully protected from "foreign I/O devices" in that all the I/O Bus signals are buffered and can be disabled under software control. To attach and use an I/O device on the I/O Bus certain requirements (both hardware and software) must be met:

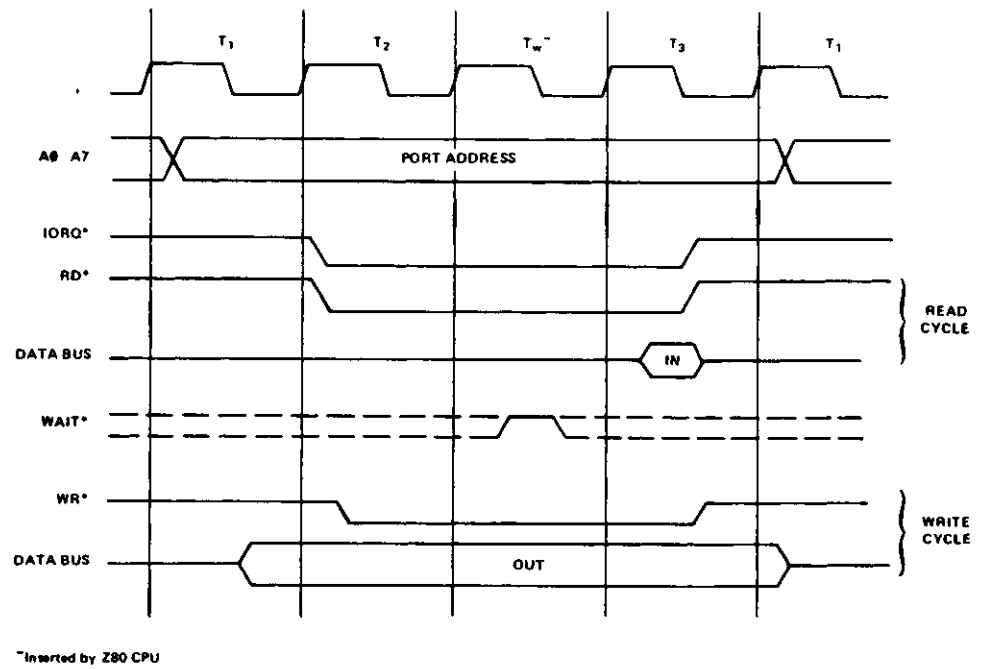
For input port device use you must enable external I/O devices by writing to port 0ECH with bit 4 on in the user software. This will enable the data bus address lines and control signals to the I/O Bus edge connector. When the input device is selected the hardware will acknowledge by asserting EXTIOSEL* low. This switches the data bus transceiver and allows the CPU to read the contents of the I/O Bus data lines. See Figure 1.6 for the timing. EXTIOSEL* can be generated by NANDing IN and the I/O port address.

Output port device use is the same as the input port device in use in that the external I/O devices must be enabled by writing to port 0ECH with bit 4 on in the user software — in the same fashion.

For either input or output devices, the IOBSWAIT* control line can be used in the normal way for synchronizing slow devices to the CPU. Note that since dynamic memories are used in the Model 4, the wait line should be used with caution. Holding the CPU in a wait state for 2 msec or more may cause loss of memory contents since refresh is inhibited during this time. It is recommended that the IOBSWAIT* line be held active no more than 500 μ sec with a 25% duty cycle.

The Model 4 will support Z 80 mode 1 interrupts. A RAM jump table is supported by the LEVEL II BASIC ROMs and the user must supply the address of his interrupt service routine by writing this address to locations 403E and 403F. When an interrupt occurs the program will be vectored to the user supplied address if I/O Bus interrupts have been enabled. To enable I/O Bus interrupts the user must set bit 3 of Port 0E0H.

Input or Output Cycles



Input or Output Cycles with Wait States.

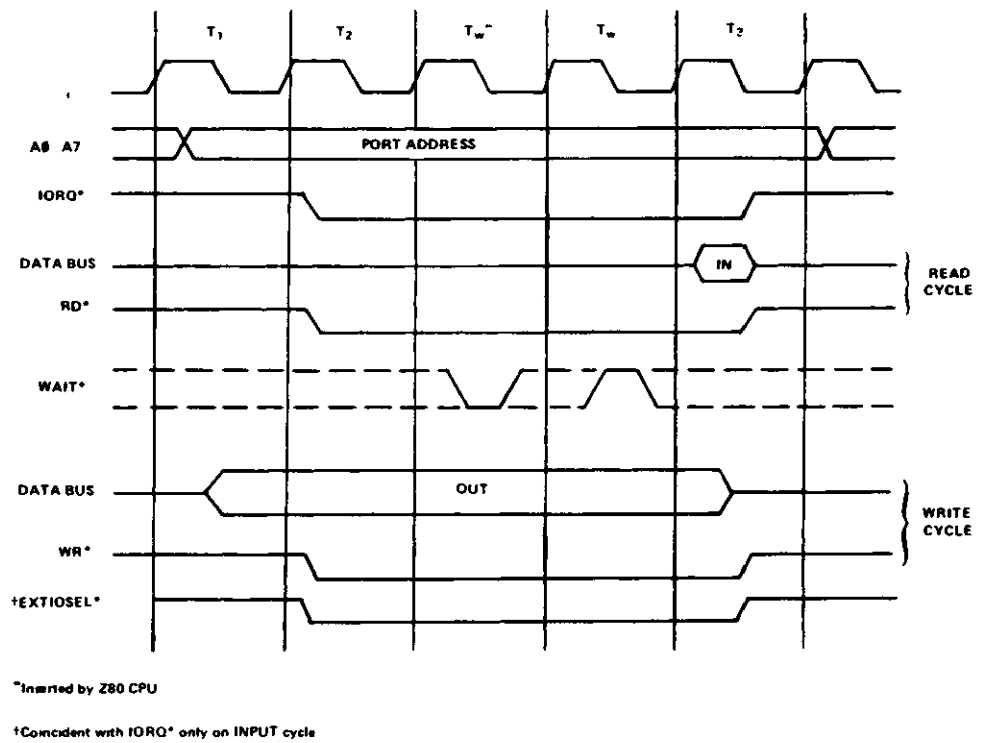


FIGURE 1-6. I/O BUS TIMING DIAGRAM

1.3 MODEL 4 PORT BITS

Name: WRNMIMASKREG*
Port Address: 0E4H
Access: WRITE ONLY

Bit 7 = ENINTRQ; 0 disables Disk INTRQ from generating an NMI.
1 enables above.

Bit 6 = ENDRQ; 0 disables Disk DRQ from generating an NMI.
1 enables above.

Name: RDNMISTATUS*
Port Address: 0E4H
Access: READ ONLY

Bit 7 = Status of Disk INTRQ; 1 = False, 0 = True

Bit 6 = Status of Disk DRQ; 1 = False, 0 = True

Bit 5 = Reset* Status; 1 = False, 0 = True

Name: MOD OUT
Port Address: 0ECH
Access: WRITE ONLY

Bit 7 = Undefined

Bit 6 = Undefined

Bit 5 = DISWAIT; 0 disables video waits, 1 enables

Bit 4 = ENEXTIO; 0 disables external IO Bus, 1 enables

Bit 3 = ENALTSET; 0 disables alternate character set,
1 enables alternate video character set.

Bit 2 = MODSEL; 0 enables 64 character mode,
1 enables 32 character mode.

Bit 1 = CASMOTORON; 0 turns cassette motor off,
1 turns cassette motor on.

Bit 0 = Undefined

Name: RDINTSTATUS*
Port Address: 0E0H
Access: READ ONLY

NOTE: A 0 indicates the device is interrupting.

Bit 7 = Undefined

Bit 6 = RS-232 ERROR INT

Bit 5 = RS-232 RCV INT

Bit 4 = RS-232 XMIT INT

Bit 3 = IOBUS INT

Bit 2 = RTC INT

Bit 1 = CASSETTE (1500 Baud) INT F

Bit 0 = CASSETTE (1500 Baud) INT R

Name: CASOUT*
Port Address: 0FFH
Access: WRITE ONLY

Bit 7 = Undefined

Bit 6 = Undefined

Bit 5 = Undefined

Bit 4 = Undefined

Bit 3 = Undefined

Bit 2 = Undefined

Bit 1 = Cassette output level

Bit 0 = Cassette output level

Name WRINTMASKREG^{*}
Port Address 0E0H
Access WRITE ONLY

Bit 7 = Undefined

Bit 6 = ENERRORINT 1 enables RS 232 interrupts on parity error, framing error, or data overrun error
 0 disable above

Bit 5 = ENRCVINT, 1 enables RS 232 receive data register full interrupts,
 0 disables above

Bit 4 = ENXMITINT 1 enables RS 232 transmitter holding register empty interrupts,
 0 disables above

Bit 3 = ENIOBUSINT, 1 enables I/O Bus interrupts,
 0 disables the above

Bit 2 = ENRTC, 1 enables real time clock interrupt,
 0 disables above

Bit 1 = ENCASINTF, 1 enables 1500 Baud falling edge interrupt,
 0 disables above

Bit 0 = ENCASINTR 1 enables 1500 Baud rising edge interrupt,
 0 disables above

Name CAS IN^{*}
Port Address 0FFH
Access READ ONLY

Bit 7 = 500 Baud Cassette bit

Bit 6 = Undefined

Bit 5 = DISWAIT (See Port 0ECH definition)

Bit 4 = ENEXTIO (See Port 0ECH definition)

Bit 3 = ENALTSET (See Port 0ECH definition)

Bit 2 = MODSEL (See Port 0ECH definition)

Bit 1 = CASMOTORON (See Port 0ECH definition)

Bit 0 = 1500 Baud Cassette bit

NOTE Reading Port 0FFH clears the 1500 Baud Cassette interrupts

Name DRVSEL^{*}
Port Address 0F4H
Access WRITE ONLY

Bit 7 = FM^{*}/MFM 0 selects single density,
 1 selects double density

Bit 6 = WSGEN, 0 – no wait states generated,
 1 = wait states generated

Bit 5 = PRECOMP, 0 = no write precompensation,
 1 = write precompensation enabled

Bit 4 = SDSEL, 0 selects side 0 of diskette,
 1 selects side 1 of diskette

Bit 3 = Drive select 4

Bit 2 = Drive select 3

Bit 1 = Drive select 2

Bit 0 = Drive select 1



SECTION II

4 GATE ARRAY THEORY OF OPERATION

2.1 MODEL 4 GATE ARRAY THEORY OF OPERATION

2.1.1 Introduction

The following discusses each element of the main board of the Model 4 Gate Array block diagram (see Figure 2-1). In each case the intent is understanding the operation on a practical level sufficient to aid in isolating a problem to the failing component.

2.1.2 Reset Circuit

Figure 2-2 shows the Reset circuit for generation of reset on power up and when the reset switch is pushed on the keyboard. The time constant determined by R8 and C25, is used to allow the system to stabilize before triggering a one shot (U63) with an approximate pulse width of 70 microseconds. When the reset switch is pushed, the input pin is brought to ground and fires the one shot when the switch is released.

A second point to be noted is the signal POWRS* which is used to reset the drive select latch in the FDC circuit.

2.1.3 CPU

The central processing unit of the Model 4 microcomputer is a Z80A microprocessor, and will run in either 2 or 4 MHz mode. All of the output lines of the Z80A are buffered. The address lines are buffered by two 74LS244s (U2 and U3 with the enable tied to ground), the control lines by a 74F04 (U27), and the data lines by a 74LS245 (U28 with the enable tied to BUSEN* and the direction control tied to BUSDIR*).

2.1.4 System Timing and Control Registers

Control Registers

The first of these registers is the WRINTMASKREG (U34). This is only part of the register as this function is shared with the Gate Array 4.5. The main register contains RTC, ENCASINTFALL, and ENCASINTRISE. The Gate Array has the interrupts for the RS232C Interface and the I/O bus interrupts and a duplicate of the RTC.

The second is the OPREG (U33) which contains the added options of the Model 4 for video and Memory mapping.

The last of the registers is MODOUT (U53) and is also readable through the CASSIN (U52) buffer. It contains the Cassette motion controls, and the FAST control for Model 4.

CPU Clock and RS232 Clock

Most of the timing generation for the board is shown in Figure 2-5. The Gate Array 4.1.1 is the basis for this timing as it produces the 20.2752 MHz clock and then divides this down to produce most of the other clocking functions used on the board.

The first clock that is produced is PCLK (pin 23) which drives the CPU. It is a divide by ten of the 20.2752 MHz in the 2 MHz mode and a divide by 5 in the 4 MHz mode. The transition from one mode to the other is without glitches and both modes are 50 percent duty cycles.

Note that the signal that controls this mode also controls the Real Time Clock circuit described later.

As a simple divide by four of the fundamental 20.2752 MHz, the RS232CLK on pin 22 of U9 provides the basic clock to the RS232C circuit.

Video and Graphics Clocking and Timing

The timing for both of these functions may be viewed as one since they must operate synchronously and the same timing must be generated for both. The additional signals sent to the Graphics Board allow it to maintain synchronization by knowing the phase relation of the signals sent to both of them. To further understand the circuit of Figure 2-5 notice the PLL Module (U8). This chip develops a 12.672 MHz signal which is phase locked to the 1.2672 MHz input on pin 5 and is a divide by 16 of the primary 20.2752 MHz clock. This provides the Gate Array 4.1.1 with two clocks to drive the video display and the graphics circuits, 10.1376 MHz for 64 character display, and a 12.672 MHz for the 80 character display.

The following discussion will consider both the 64 and 80 character displays to be the same, the difference being the primary frequency and not the phase relation or function of the signals generated.

The reference clock for the timing is DCLK (U9-15) and the other clocks that are produced for the video output are derived from this clock (DOT* at U9-17 is a phase shift of DCLK and is provided as an option for the dot clock for variations in delay paths in the video section). U9 then generates SHIFT* (pin 21), XADR7* (pin 20), CRTCLK (pin 19), LOADS* (pin 18), and LOAD* (pin 16) for the proper timing for the four video modes. In addition for the Graphics Board to synchronize with this timing H (pin 14), I (pin 13), and J (pin 11) are fed to connector J12. See Figures 2-6 and 2-7 for the timing diagrams for video clocks generated by Gate Array 4.1.1.

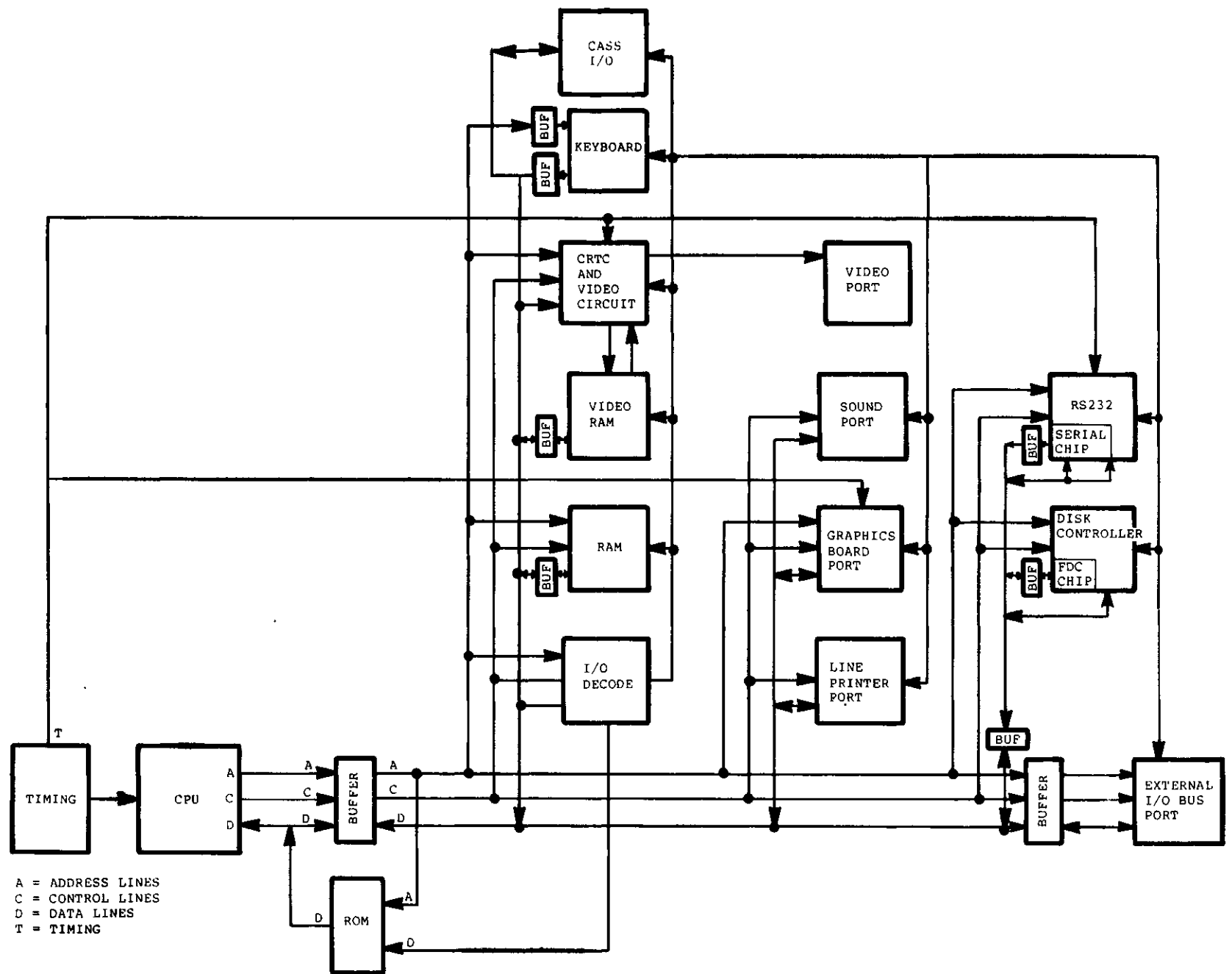
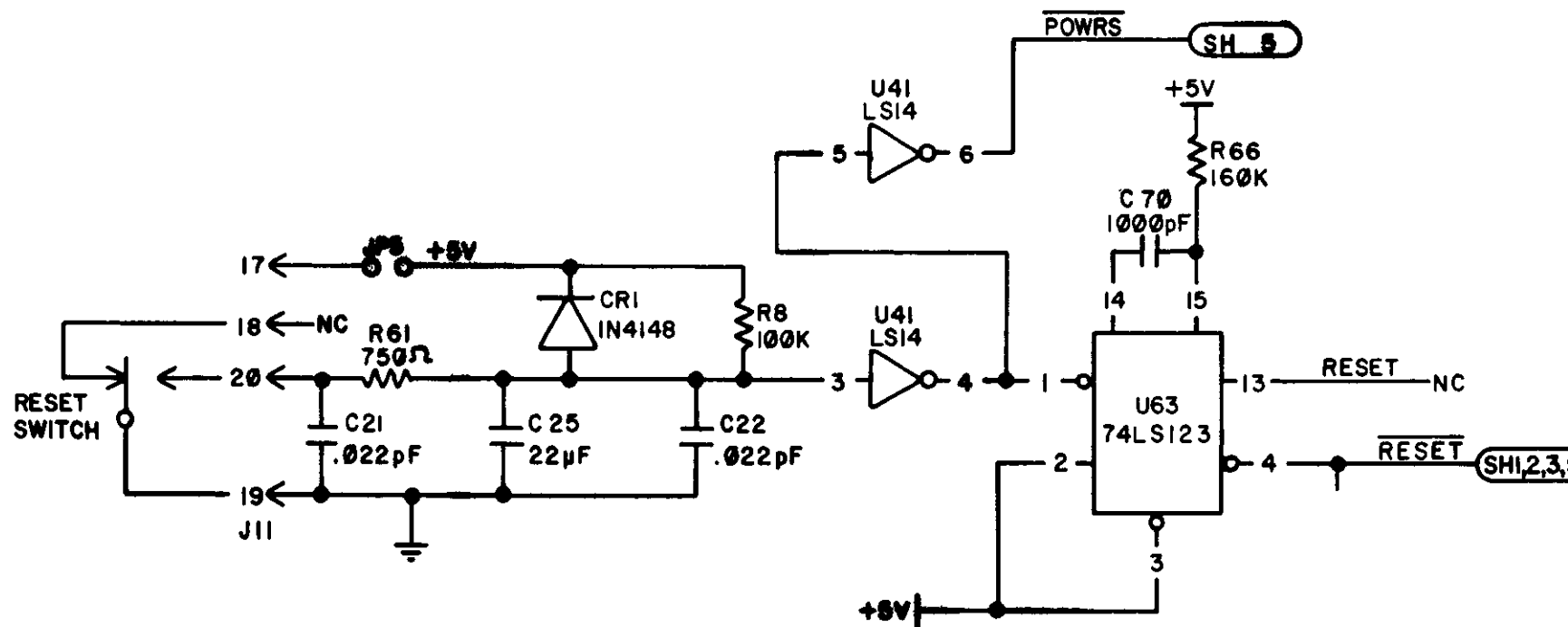


Figure 2-1. Model 4 Gate Array Functional Block



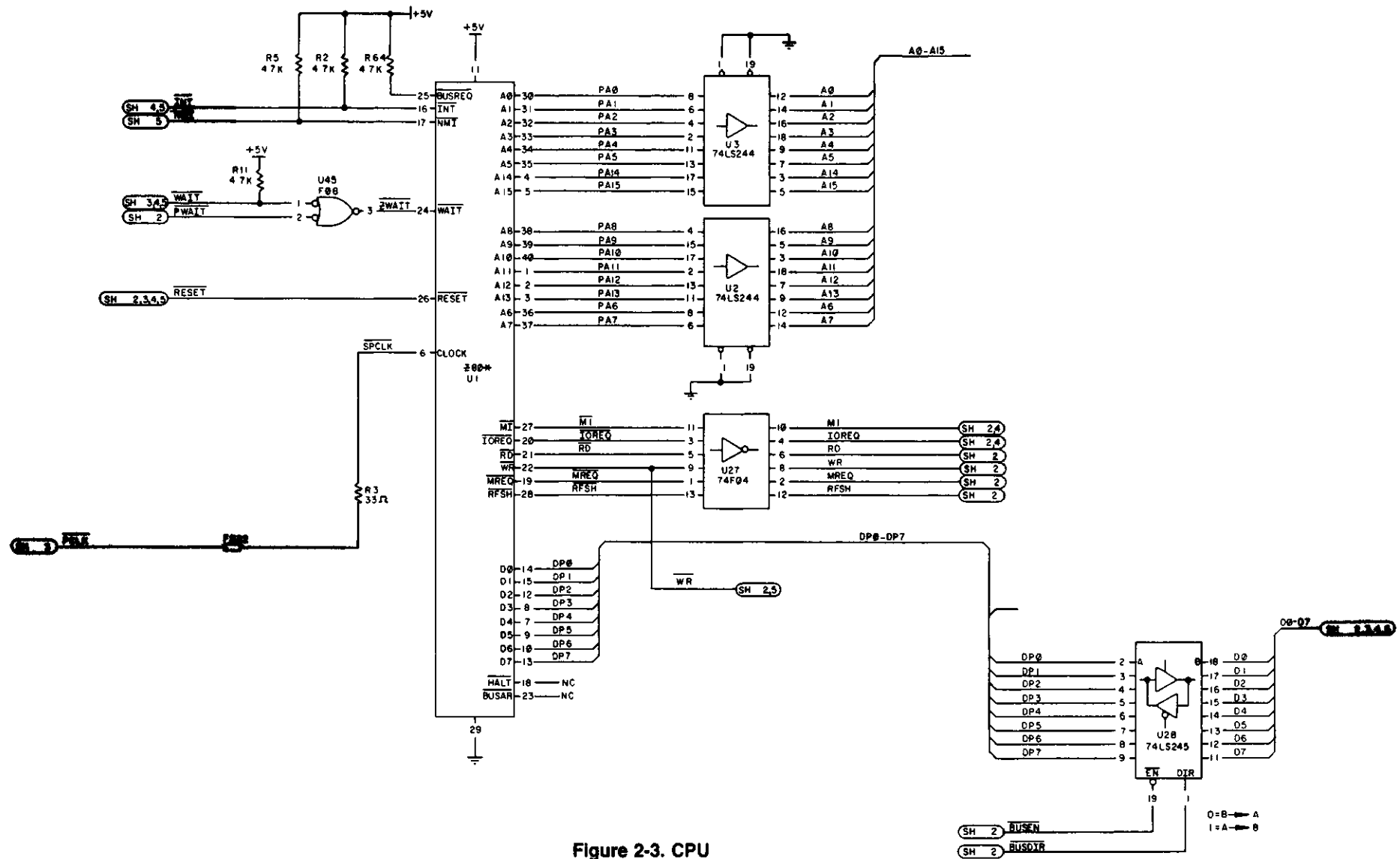


Figure 2-3. CPU
(Page 1 of Schematic)

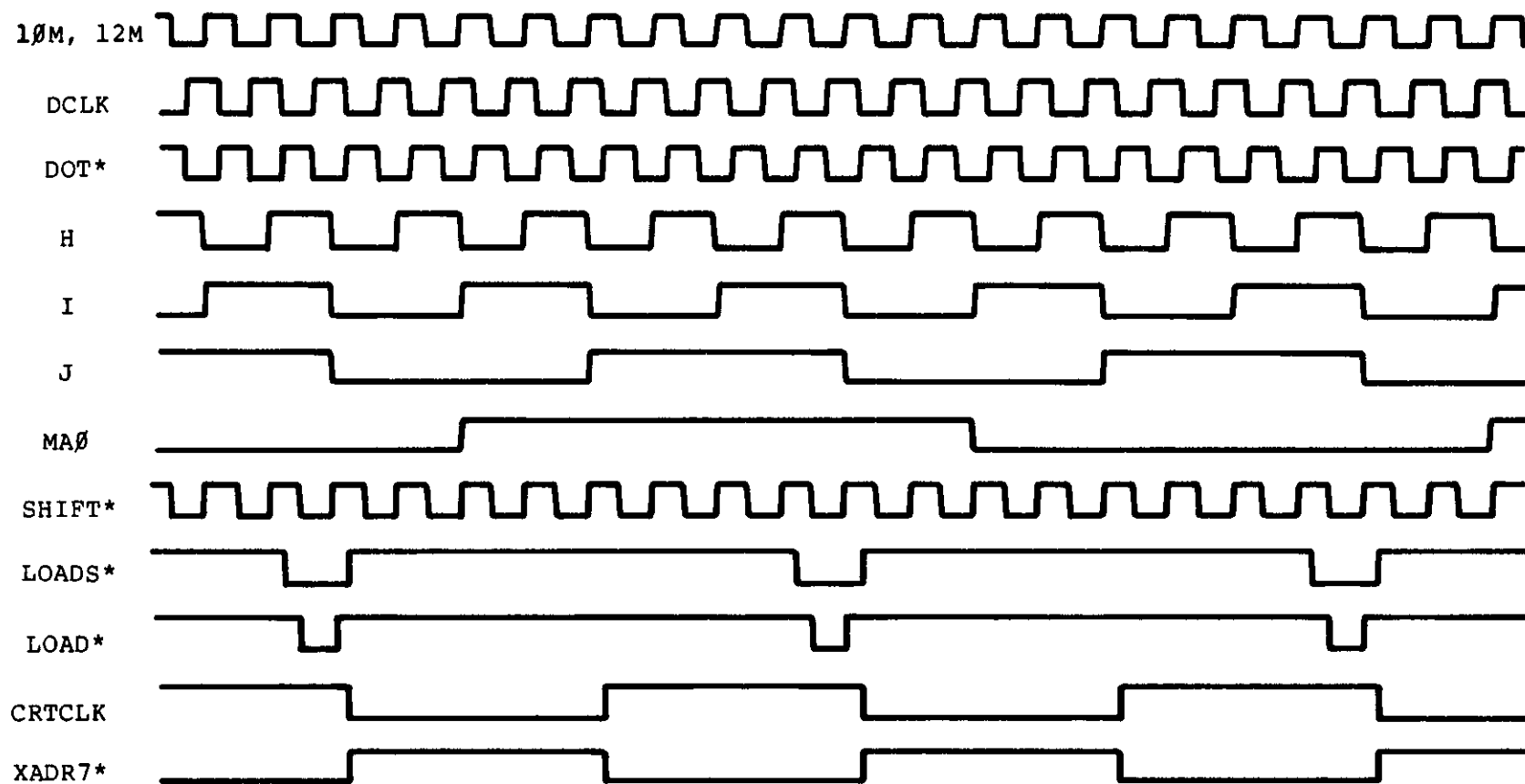


Figure 2-6. Video Timing 64 x 16 Mode 80 x 24 Mode

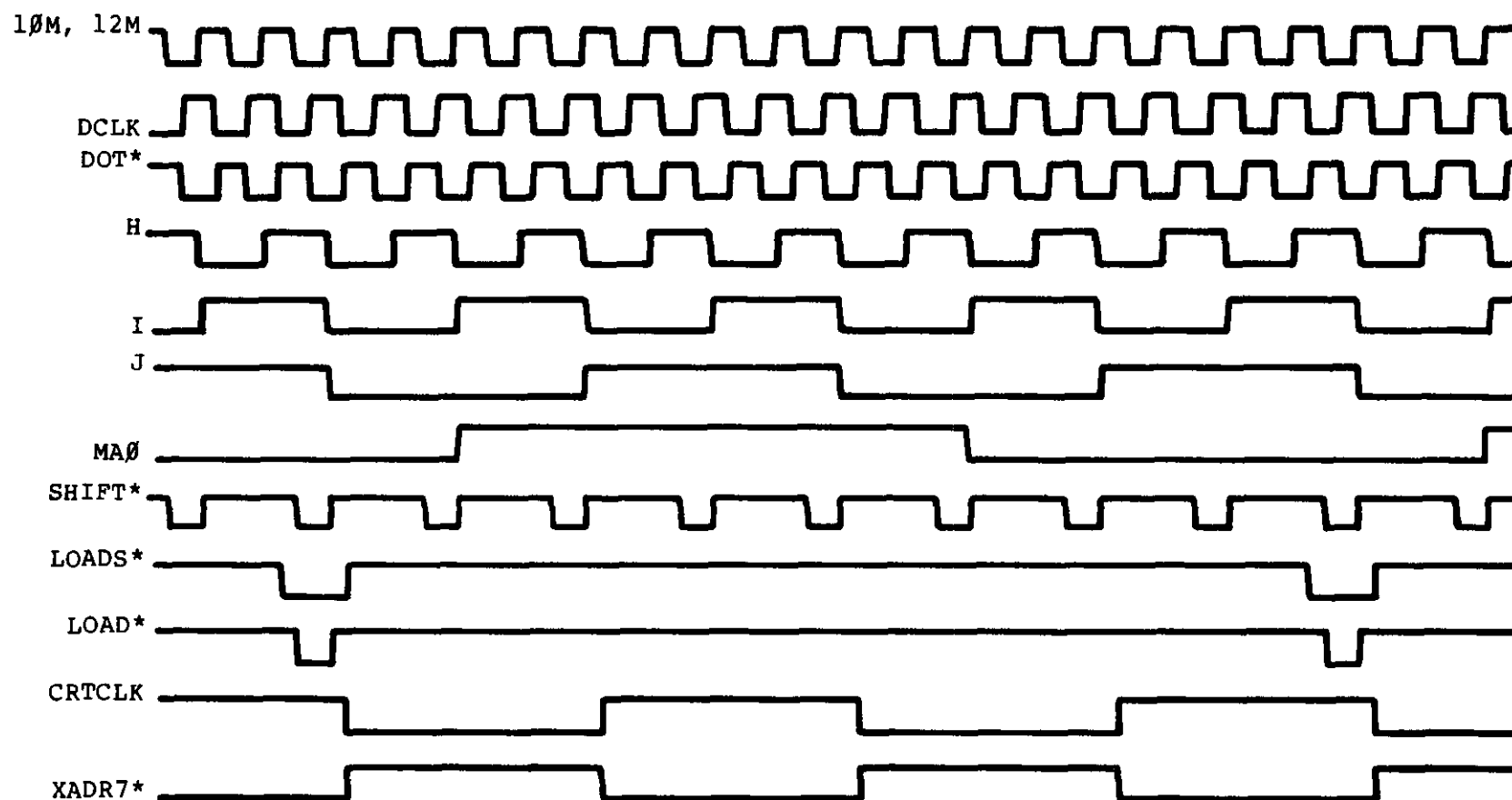


Figure 2-7.

DRAM and Video RAM Timing

The Video RAM and DRAM timing share the timing delay line (U80). This is done by 'OR'ing the two signals GRAS* and AINPRG* at U39 to get the signal STDEL*. This is possible because the signals VIDEO and MREQ or MCYCEN are gated in to mask off the signals that are not desired.

Since the CRTC and the CPU are operating independently and at different clock rates, when the CPU wants to access the Video RAM the two must synchronize with each other. This is accomplished when a video access is decoded. WAIT* is pulled low, when it is determined whether the access is a read or write and the correct cycle of the CRTC clock is present, the actual access can begin, hence AINPRG* is generated and WAIT* is released.

From this point the actual sequence depends on whether a read or a write is done. On a read the address is enabled to the RAM, the delay through U80 to VLATCH* when data is latched in the 74LS373 where the CPU can pick-up the data at the completion of this cycle. On a write the sequence is more complex. The address is enabled to the RAM, the output is disabled (VRAMDIS* at U7-12), write is delayed with respect to the address (DLYWR* at U60-6) and the buffer on the data lines is enabled (VBUFEN* at U60-8), then after a delay the write is cutoff to end the cycle for the RAM (ENDVW* at U80-10). For the timing diagram of the Video RAM CPU access see Figure 2-8.

DRAM Timing

The DRAM timing is shown in Figure 2-9. At the beginning of the CPU cycle the address lines settle-out first and are, therefore, decoded to allow maximum access speed (see Address Decode). With the generation of MREQ, U39-11 generates PMREQ and enables U42 and gates this with the type of cycle to develop GRAS* (U30-6), RAS0* (U30-3), and RAS1* (U30-11). GRAS* is then "OR"ed with AINPRG as mentioned above. The timing from this point is very straight forward. With RAS0* and RAS1* generated next MUX (U80-12) is built to switch the addresses to memory then GCAS is generated and clocks flip-flop U31 with MCYCEN on the J term. This is done to make sure this is a true memory cycle. Then if this is an M1 cycle VLATCH* clocks at U31 and cuts off PMREQ* at U39 to end the cycle. For timing diagrams of the memory interface see Figures 2-10 to 2-12.

2.1.5. Address Decode

This section is divided into two parts, the memory addressing and the I/O addressing. This separation is a reflection of the separate mapping of memory and I/O of the Z80A itself. For reference of both sections, see Figure 2-13.

Memory Address

The memory map for the Model 4 is shown in Table 2-1 and is best described as an option overlay in the sense that at each step of additional memory, the new options overlap the previous and the new options are added on. Moreover, the added options have no effect on previous levels and are invisible at those levels.

MAP I*	MAP II	Address in hex MAP III	MAP IV	Function of block
0000-37E7	0000-37FF	0000-F3FF	0000-FFFF	RAM (64K)
37E8-37E9				ROM
37EA-37FF				Printer Status
3800-3BFF	3800-3BFF	F400-F7FF		ROM
3C00-3FFF**	3C00-3FFF**	F800-FFFF		Keyboard
4000-7FFF				Video RAM
4000-FFFF	4000-FFFF			RAM (16K)
				RAM (64K)

Table 2-1

* Only map available on 16K machine

** Page bit is used to select 1K of 2K Video RAM

The decoding of the addresses for the memory map described above is done for the most part by U5. The only decode not done by U5 is the line printer memory status port at 37E8 and 37E9 hex. These needed additional address lines hence the decode LPADD as an input to U5.

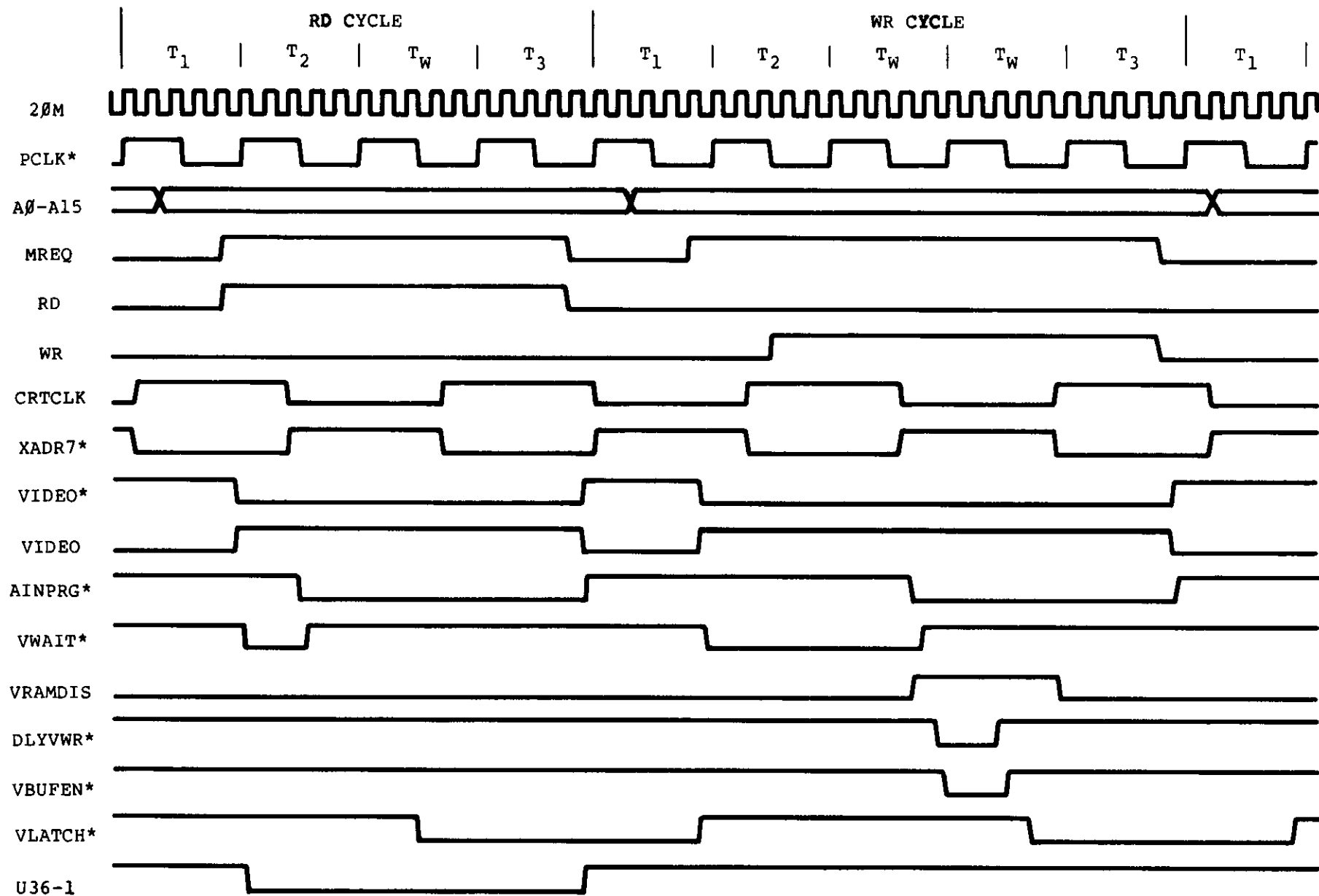


Figure 2-8. Video RAM CPU Access Timing

Waveform Symbol	Input	Output	Waveform Symbol	Input	Output
	Must Be Valid	Will Be Valid		Don't Care Any Change Permitted	Changing State Unknown
	Change From H to L	Will Change From H to L			High Impedance
	Change From L to H	Will Change From L to H			

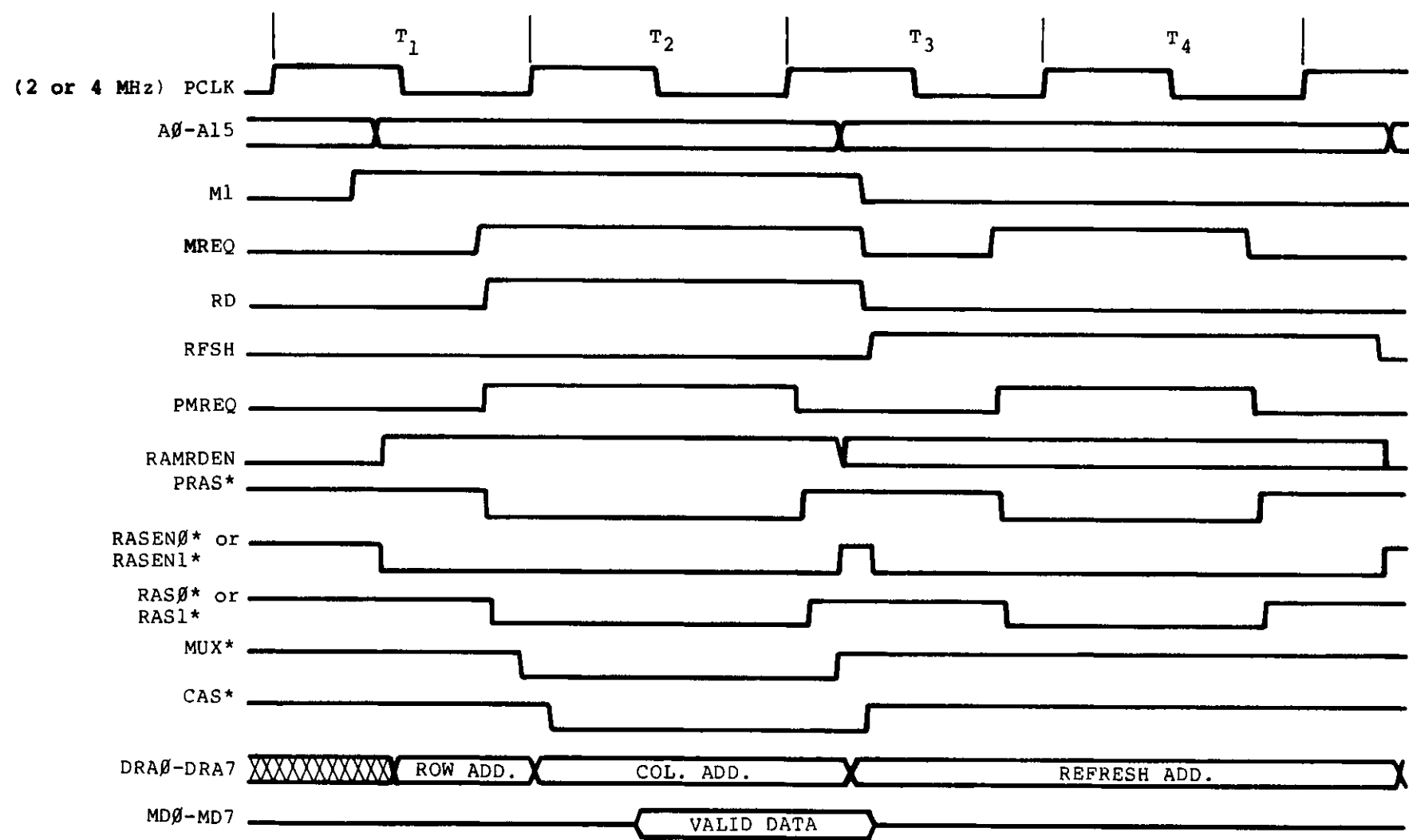


Figure 2-10. M1 Cycle Timing

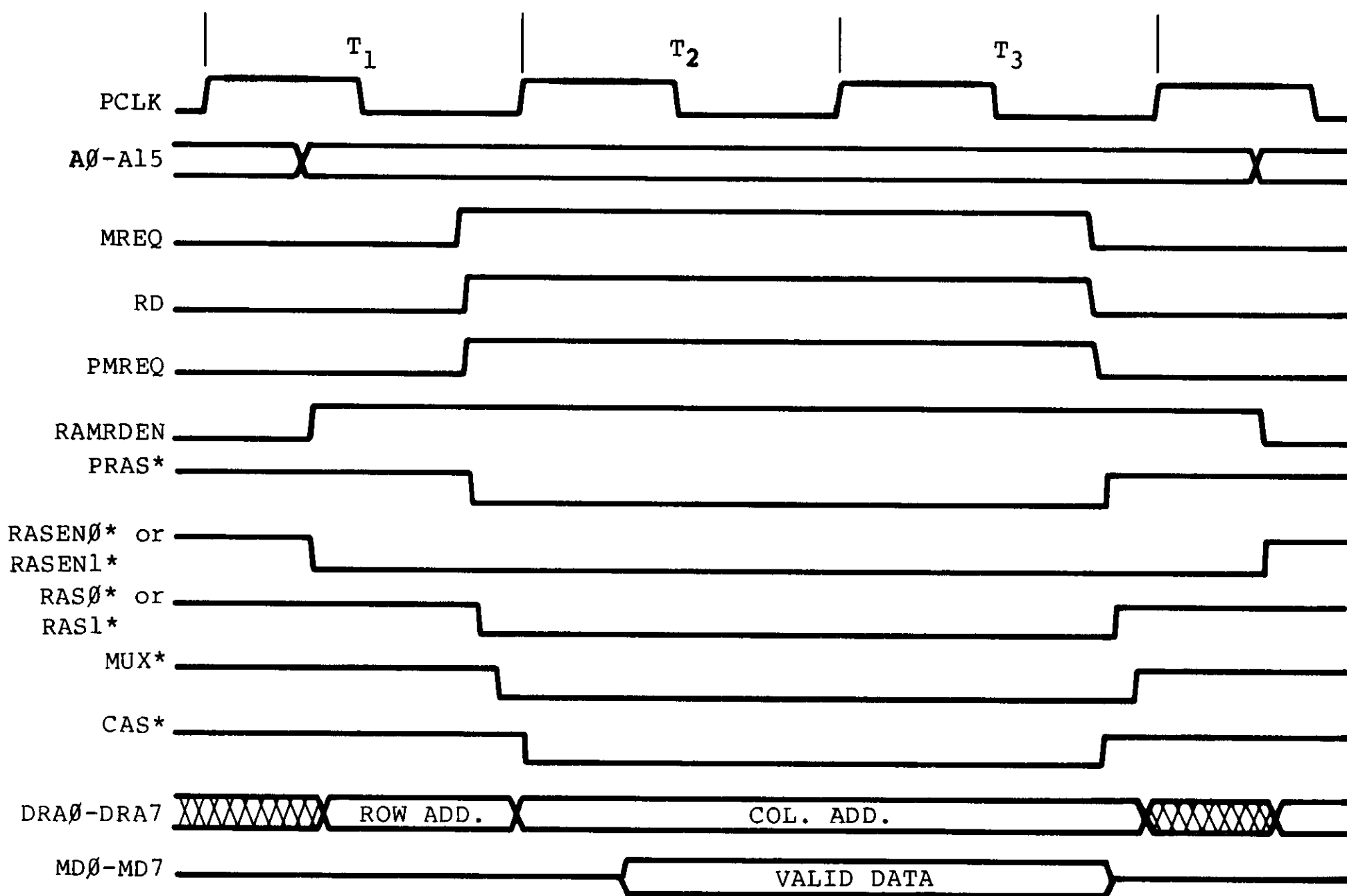


Figure 2-11. Memory Read Cycle Timing

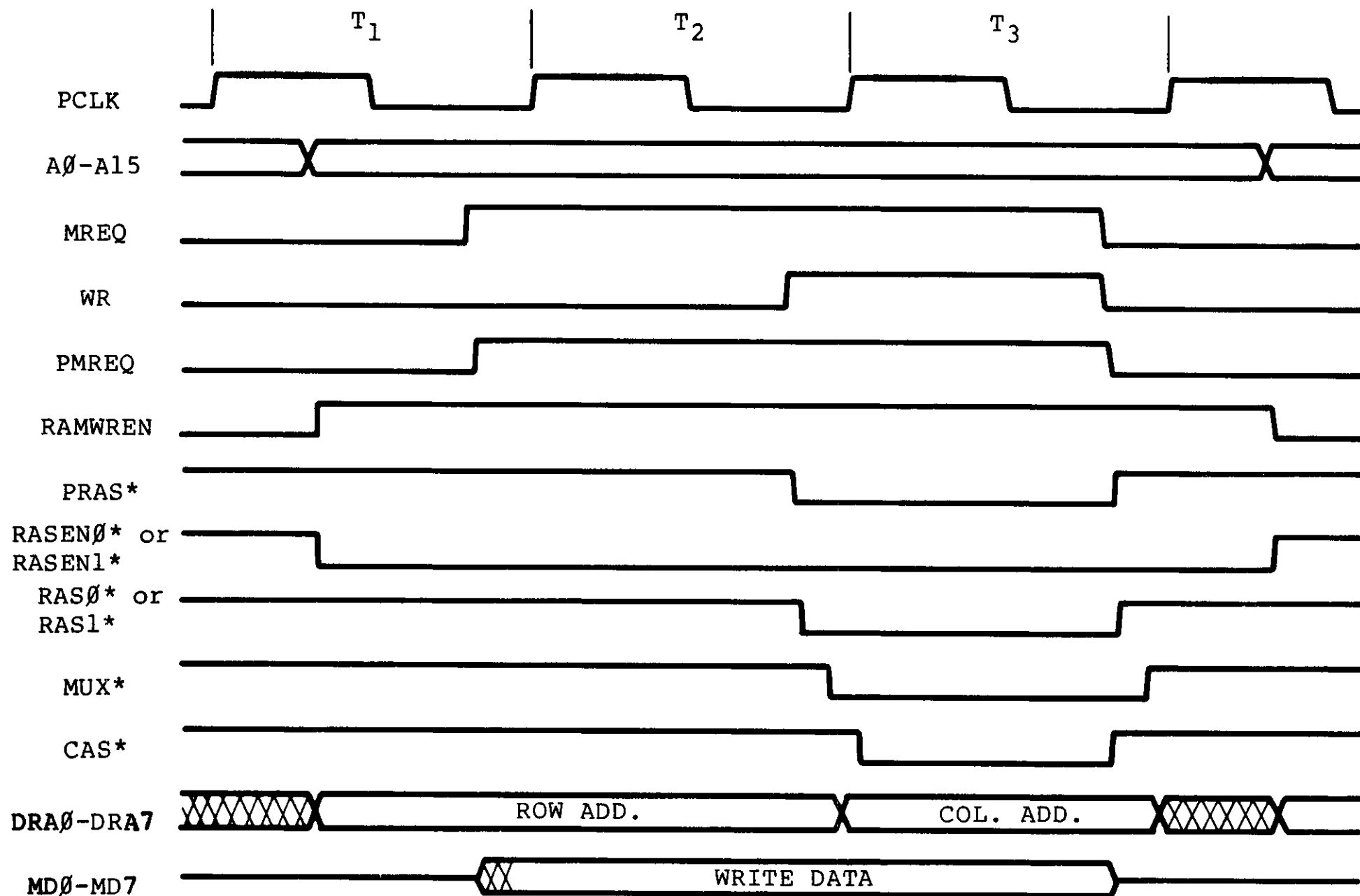


Figure 2-12. Memory Write Cycle Timing

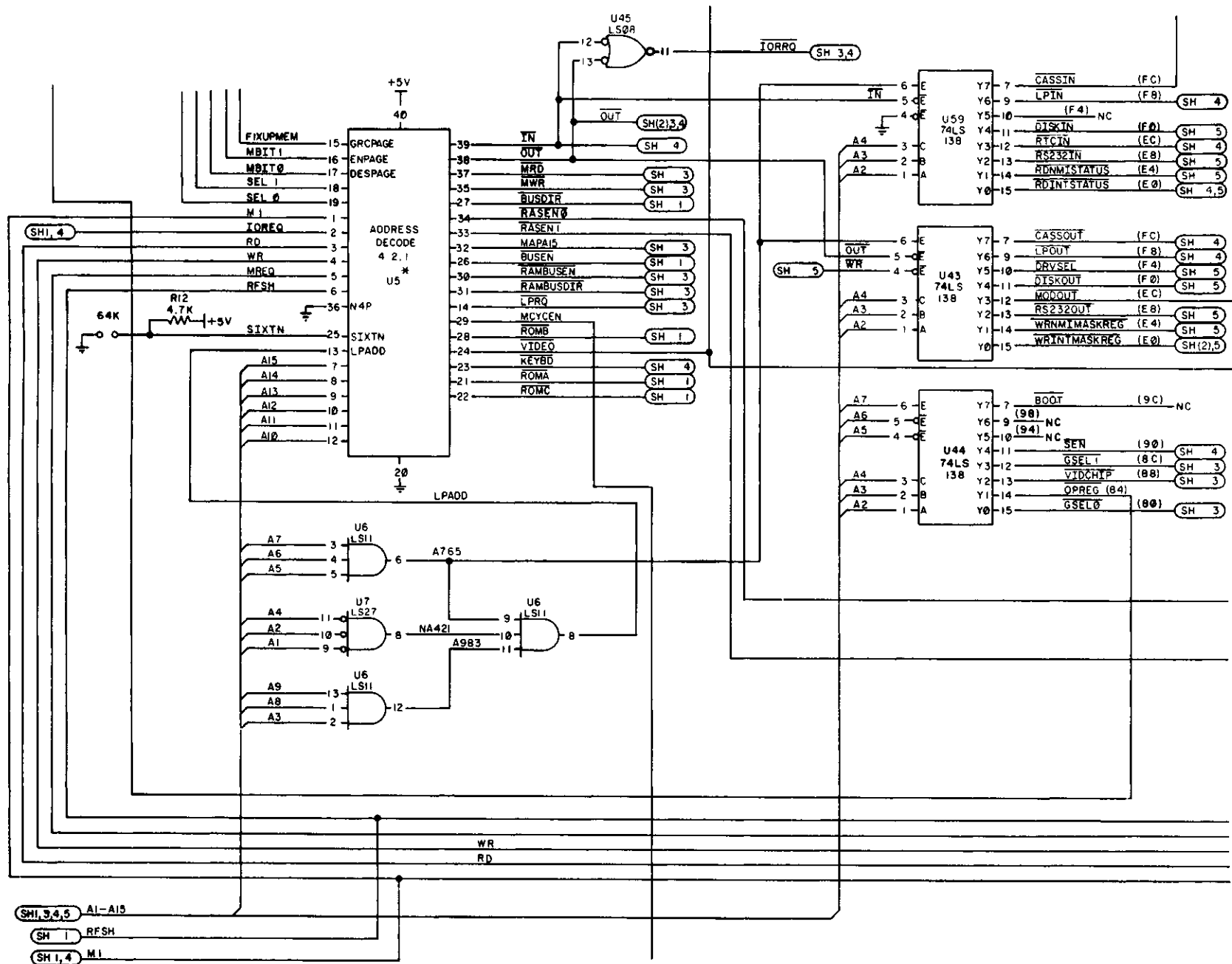


Figure 2-13. Address Decode
(Page 2 of Schematic)

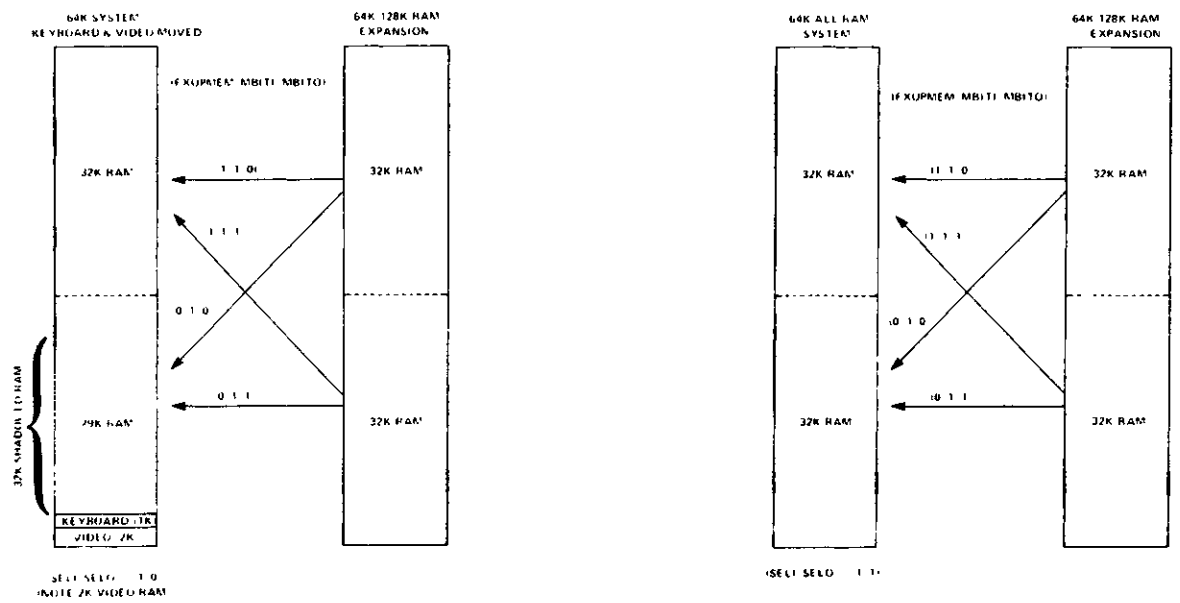
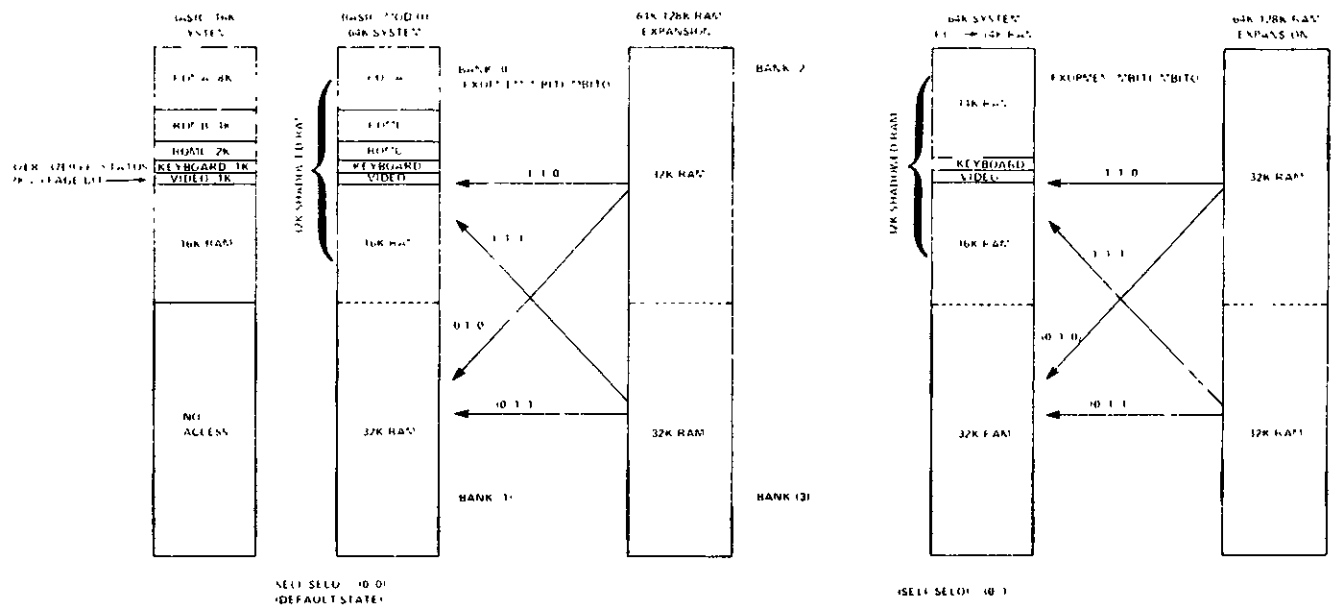


Table 2-2. RAM Memory

I/O port Address

The Port Map decoding is accomplished by three 74LS138s (U43,U44, and U59) These ICs decode the low order address lines (A0 – A7) from the CPU and decode the port being selected The IN* signal and OUT* signal are used in the decode for U59 and U43, but U44 is a pure address decode and, therefore, needs to be gated with IN*, OUT*, or IOREQ* later For a complete I/O map see Table 2-3

2.1.6. ROM

The A ROM is enabled by the decode as appropriate by the address logic described above, and is addressed in a simple straight forward fashion The enable for the B/C ROM is also similarly accomplished, however, the address has a jumper option available This option is designed to allow for testing of the board logic in the factory When jumper is moved from JP8 to JP7, the ROM is in the test mode, with the options appearing on the screen

2.1.7 DRAM

The DRAM timing was described earlier in the timing section, the actual DRAM is contained in two banks of eight each U65 to U74 and U85 to U92 They are arranged in order of data bits D0 through D7, U65 and U85 being D0, through U74 and U92 being D7 Note in Figure 2-15 that the two banks are different with jumper options in the lower bank, these options are for the possible use of 16k three voltage parts When jumpered as shown in Figure 2-14 the bank is identical to the second bank and is for using 64k DRAMs With both banks filled there is 128k available to the user

2.1.8 Video Circuit

Video Modes

The Model 4 has many video options available through hardware and software Software has control of inverse video on a character by character basis by turning on IN-VIDE Note that this implies the available number of characters is now 128 since the most significant bit of the character code in memory is now used to indicate inverse character Similarly, an alternate character set can be enabled by turning on ENALTSET This enables a new 64 characters in place of the last 64 characters, that is, the Kana set in place of the game set An option not available to software is an enhanced character, which moves characters down one row in their character block to make an inverse character appear within the inverse block and not on the edge of the block This is done by moving jumper JP11 to JP12 As an example of a combination of hardware and software options available in the video is the overlay, which not only requires the Graphics Board to be installed, but also software to enable the graphics data and the video data with text at the same time

The Model 4 also has an option for either 64 character or 80 character wide screen The 64 character screen is compatible with the Model III and displays 16 lines The 80 character screen displays 24 lines In addition each of these has a double width mode These options are controlled by two bits, MODSEL and 8064 which provide the screens as shown in the following table

8064	MODSEL	Video Screen Size
0	0	64 x 16
0	1	32 x 16
1	0	80 x 24
1	1	40 x 24

Table 2-4

With this information of the options available to the user we can now view the actual operation of the circuit with the final objectives in mind and see how they are achieved For the rest of this section all references will be made to Figure 2-16 The first task to be accomplished would be the screen refresh and this is done by the CRTC or 68045 (U11) which will generate the addresses continuously on its address lines Then to allow the CPU access to the same memory the address lines are multiplexed at U12, U14, and U15 on opposite phases of the CRT clock The CPUs access timing is then handed by the timing circuit described earlier

The data bus of the RAM (U16) is a two way bus with the RAM as a source or destination on all accesses, the video gate array (U17) is the destination on the screen refresh half of the cycle, the 74LS373 (U36) is the destination on a read of the RAM by the CPU, and the 74LS244 (U35) is the source on writes to the RAM

The video gate array then gates the RAM data INVIDE, and ENALTSET to determine the ROM addressing for these two options and CHRADD to the 74LS283 (U13) which takes the row address from the 68045 and adds a zero to the row address or a minus one to form the character enhanced mode

The data out of the ROM is then sent back to the gate array where it is then changed to a serial stream of data which is synchronized with the data that would come from the graphics board, GRAFVID The signal CL166 will inhibit the data out of the serial register and the signal ENGRAF enables the graphics data, hence, if both are enabled the effect is an overlay The output data is sent to U20 pin 9 where it is gated with one of two phases of the dot clock, then after being filtered to lower the RFI it is output to the sweep board

Model 4 Port Bit Map

Port	D7	D6	D5	D4	D3	D2	D1	D0
FC - FF	Cass							Cassette
(READ)	data 500 bd							data 1500 bd
FC - FF								
								(MIRROR of PORT EC)
								(Note, also resets cassette data latch)
								cass.
								cassette
(WRITE)	x	x	x	x	x	x	out	data out
F8 - FB	Prntr	Prntr	Prntr	Prntr	x	x	x	x
(READ)	BUSY	Paper	Select	Fault	x	x	x	x
F8 - FB	Prntr	Prntr	Prntr	Prntr	Prntr	Prntr	Prntr	Prntr
(WRITE)	D7	D6	D5	D4	D3	D2	D1	D0
EC - EF								
								(Any Read causes reset of Real Time Clock Interrupt)
EC - EF	x	CPU	x	Enable	Enable	Mode	Cass	x
(WRITE)	x	Fast	x	EX I/O	Altset	Select	Mot On	x
E0 - E3	x	Receive	Receive	Xmit	10 Bus	RTC	C Fall	C Rise
(READ)	x	Error	Data	Empty	Int	Int	Int	Int
E0 - E3	x	Enable	Enable	Enable	Enable	Enable	Enable	Enable
(WRITE)	x	Rec Err	Rec Data	Xmit Emp	10 Int	RT Int	CF Int	CR Int
90 - 93	x	x	x	x	x	x	x	Sound
(WRITE)	x	x	x	x	x	x	x	Bit
84 - 87	Page	Fix Up	Memory	Memory	Invert	80/64	Select	Select
(WRITE)		Memory	Bit 1	Bit 0	Video		Bit 1	Bit 0

Table 2-3. I/O Port Map

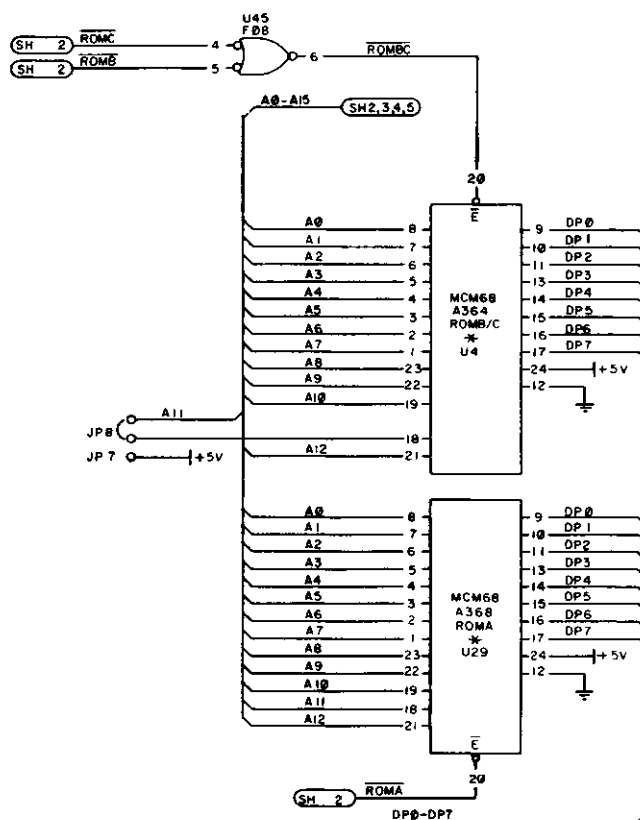
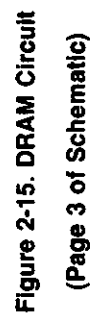


Figure 2-14. ROM Circuit
(Page 1 of Schematic)



2.1.9 Keyboard

The interface to the keyboard is a matrix composed of address lines in one direction and data lines in the other. The address lines have two open collector buffers (U26 and U40) on the output to the keyboard.

The input is pulled-up with an 820 ohm resistor and is then fed into two CMOS Inputs (U55 and U56) which act as a driver on data lines.

2.1.10 Real Time Clock

The Real Time Clock circuit in the Model 4 provides a 30 Hz (in the 2 MHz CPU Mode) or 60 Hz (in the 4 MHz CPU Mode) interrupt to the CPU. By counting the number of interrupts that have occurred, the CPU can keep track of the time. The 60 Hz vertical sync signal from the video circuitry is divided by two (2 MHz Mode) by U10 and the 30 Hz at pin 9 of U46 is used to generate the interrupts. In the 4 MHz mode, the signal FAST places a logic low at pin 4 of U10, causing the signal VSYNC to pass through U46 at its normal rate and trigger interrupts at the 60 Hz rate. Note that any time interrupts are disabled, the accuracy of the clock suffers.

2.1.11 Line Printer Port

The printer status lines are read by the CPU by enabling buffer U108. This buffer will be enabled for any input from port F8 or F9, or any memory read from location 37E8 or 37E9 when in the Model III mode. For a listing of bit status, refer to the bit map.

After the printer driver software determines that the printer is ready to receive a character (by reading the status), the character to be printed is output to port F8. This latches the character into U107, and simultaneously fires the one-shot U63 to provide the appropriate strobe to the printer.

2.1.12 Graphics Port

The graphics port on the Model 4 is provided to attach the optional high resolution graphics board and provides the necessary signals to interface not only to the CPU (such as data lines, address lines, address decodes, and control lines), but also the signals needed to synchronize the output of the Video Circuit and the Graphics board and control to provide features such as overlay.

Pin Number Signature

1	D0
2	D1
3	D2
4	D3
5	D4
6	D5
7	D6
8	D7
9	GEN*
10	DCLK
11	A0
12	A1
13	A2
14	J
15	GRAPVID
16	ENGRAF
17	DISBEN
18	VSYNC
19	HSYNC
20	RESET*
21	WAIT*
22	H
23	I
24	IN*
25	GND
26	+5
27	N/C
28	CL166
29	GND
30	+5
31	GND
32	+5
33	GND
34	+5

Table 2-5

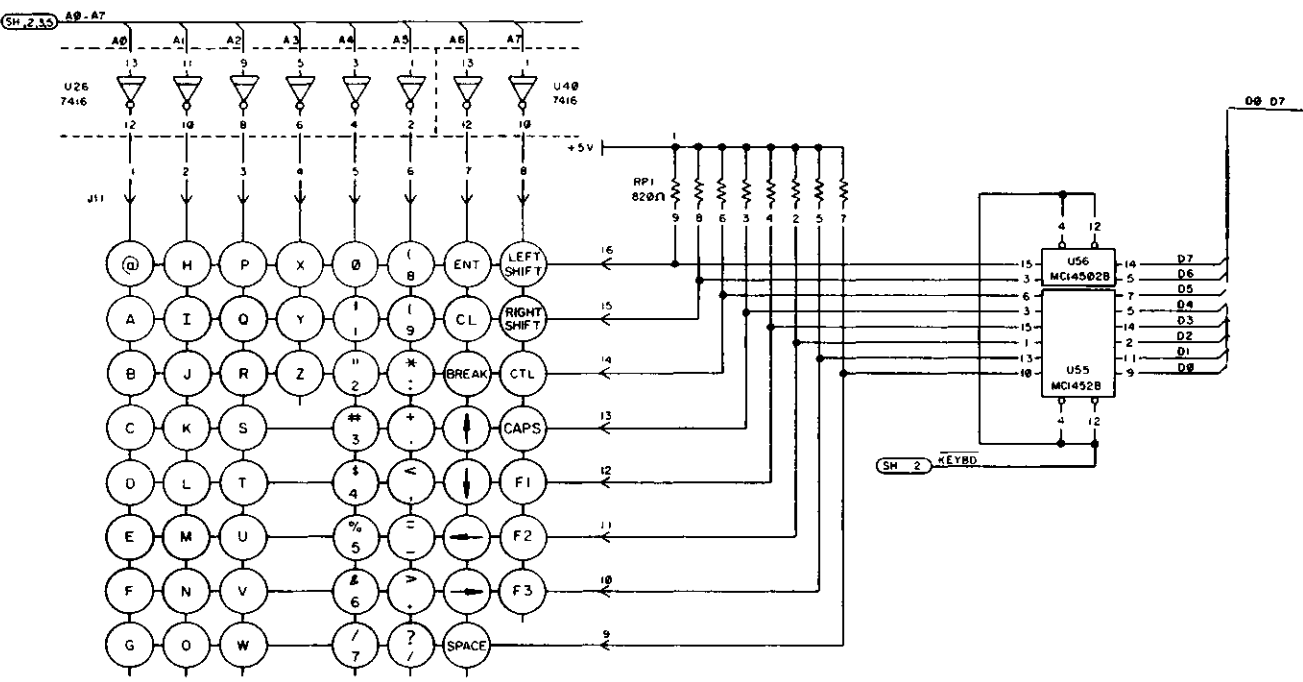


Figure 2-17. Keyboard
(Page 4 of Schematic)

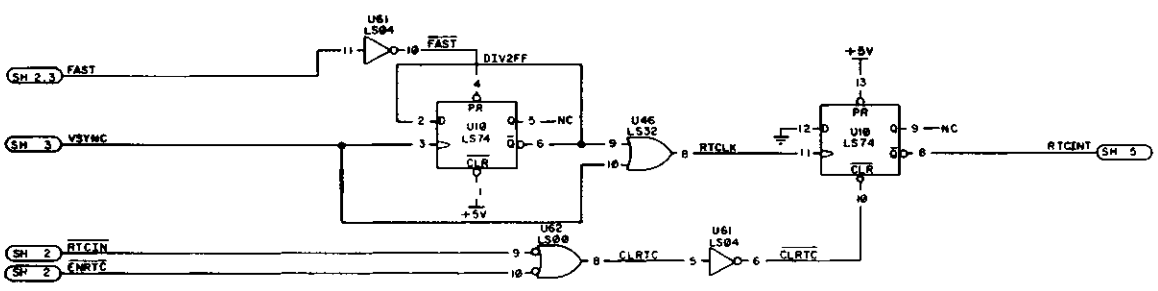


Figure 2-18. RTC
(Page 4 of Schematic)

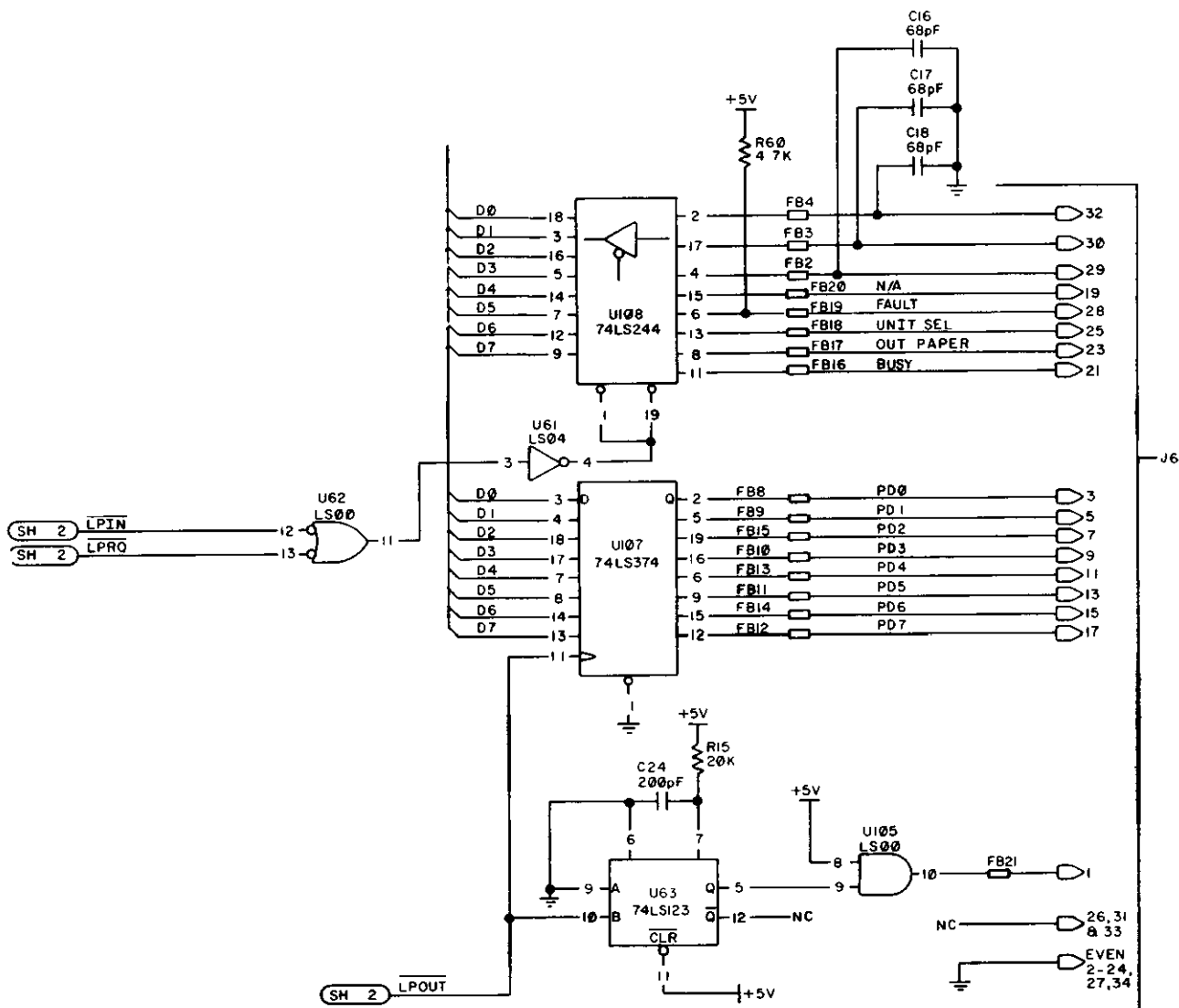


Figure 2-19. Printer Circuit

(Page 4 of Schematic)

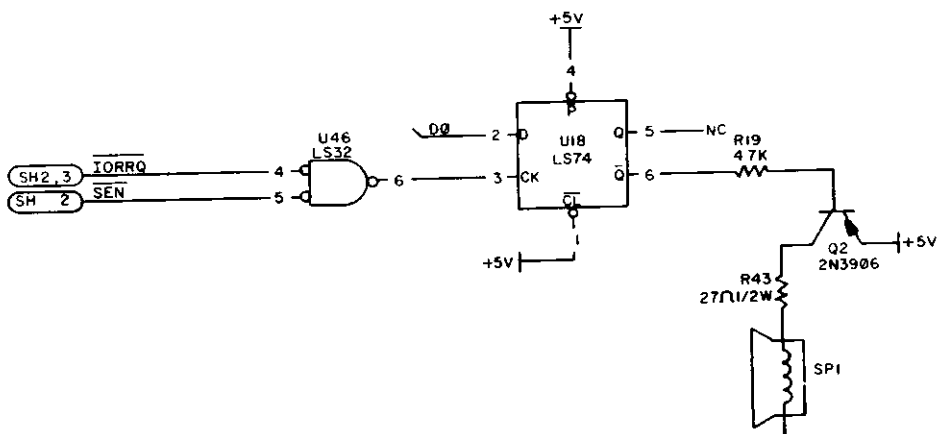


Figure 2-20. Sound

(Page 4 of Schematic)

2.1.13 Sound Port

The sound circuit is compatible with the optional sound board on the older version of the Model 4 and works in a similar fashion. Sound is generated by setting and clearing data bit zero on successive OUTs to port 90H. The state of D0 is latched in U18 which is amplified by Q2 to drive the speaker (SP1).

2.1.14 I/O Bus Port

The Model 4 Gate Array Bus is designed to allow easy and convenient interfacing of I/O devices to the Model 4. The I/O Bus supports all the signals necessary to implement a device compatible with the Z-80s I/O structure. That is:

Addresses

A0 to A7 allow selection of up to 256 input and 256 output devices if external I/O is enabled.

Ports 80H to 0FFH are reserved for System use.

Data

DB0 to DB7 allow transfer of 8-bit data onto the processor data bus if external I/O is enabled.

Control Lines

- a IN* — Z-80 signal specifying that an input is in progress. Gated with IORQ.
- b OUT* — Z-80 signal specifying that an output is in progress. Gated with IORQ.
- c RESET* — system reset signal.
- d IOBUSINT* — input to the CPU signaling an interrupt from an I/O Bus device if I/O Bus interrupts are enabled.
- e IOBUSWAIT* — input to the CPU wait line allowing I/O Bus device to force wait states on the Z-80 if external I/O is enabled.
- f EXTIOSEL* — input to CPU which switches the I/O Bus data bus transceiver and allows an INPUT instruction to read I/O Bus data.
- g M1* — and IORQ* — standard Z-80 signals.

The address line, data line, and control lines a to c and e to g are enabled only when the ENEXIO bit is set to a one.

To enable I/O interrupts, the ENIOBUSINT bit in the CPU IO-PORT E0 (output port) must be a one. However, even if it is disabled from generating interrupts, the status of the IOBUSINT* line can still read on the appropriate bit of CPU IO-PORT E0 (input port).

See Model 4 Port Bit assignment for 0FF, 0EC, and 0E0.

The Model 4 CPU board is fully protected from foreign I/O devices in that all the I/O bus signals are buffered and can be disabled under software control. To attach and use an I/O device on the I/O Bus, certain requirements (both hardware and software) must be met.

For input port device use, you must enable external I/O devices by writing to port 0ECH with bit 4 on in the user software. This will enable the data bus, address lines, and control signals to the I/O Bus edge connector. When the input device is selected, the hardware will acknowledge by asserting EXTIOSEL* low. This switches the data bus transceiver and allows the CPU to read the contents of the I/O Bus data lines. See Figure 2-21 for the timing. EXTIOSEL* can be generated by NANDing IN and the I/O port address.

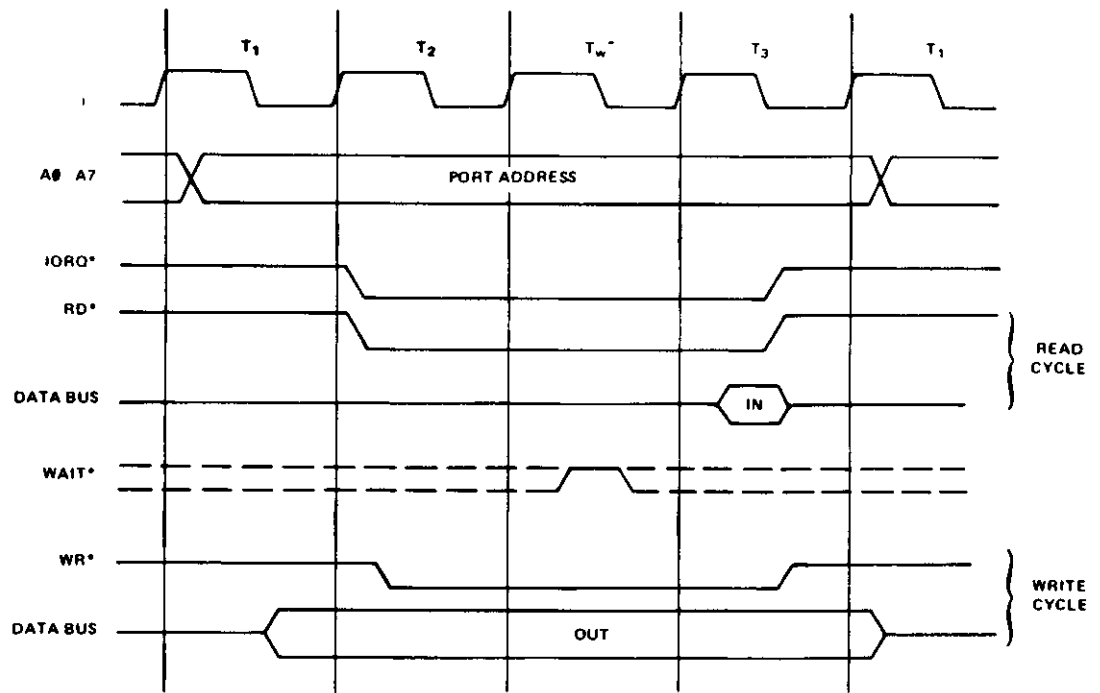
Output port device use is the same as the input port device in use, in that the external I/O devices must be enabled by writing to port 0ECH with bit 4 on in the user software — in the same fashion.

For either input or output devices, the IOBUSWAIT* control line can be used in the normal way for synchronizing slow devices to the CPU. Note that since dynamic memories are used in the Model 4, the wait line should be used with caution. Holding the CPU in a wait state for 2 msec or more may cause loss of memory contents since refresh is inhibited during this time. It is recommended that the IOBUSWAIT* line be held active no more than 500 msec with a 25% duty cycle.

The Model 4 will support Z-80 mode 1 interrupts. A RAM jump table is supported by the LEVEL II BASIC ROMs and the user must supply the address of his interrupt service routine by writing this address to locations 403E and 403F. When an interrupt occurs, the program will be vectored to the user supplied address if I/O Bus interrupts have been enabled. To enable I/O Bus interrupts, the user must set bit 3 of Port 0E0H.

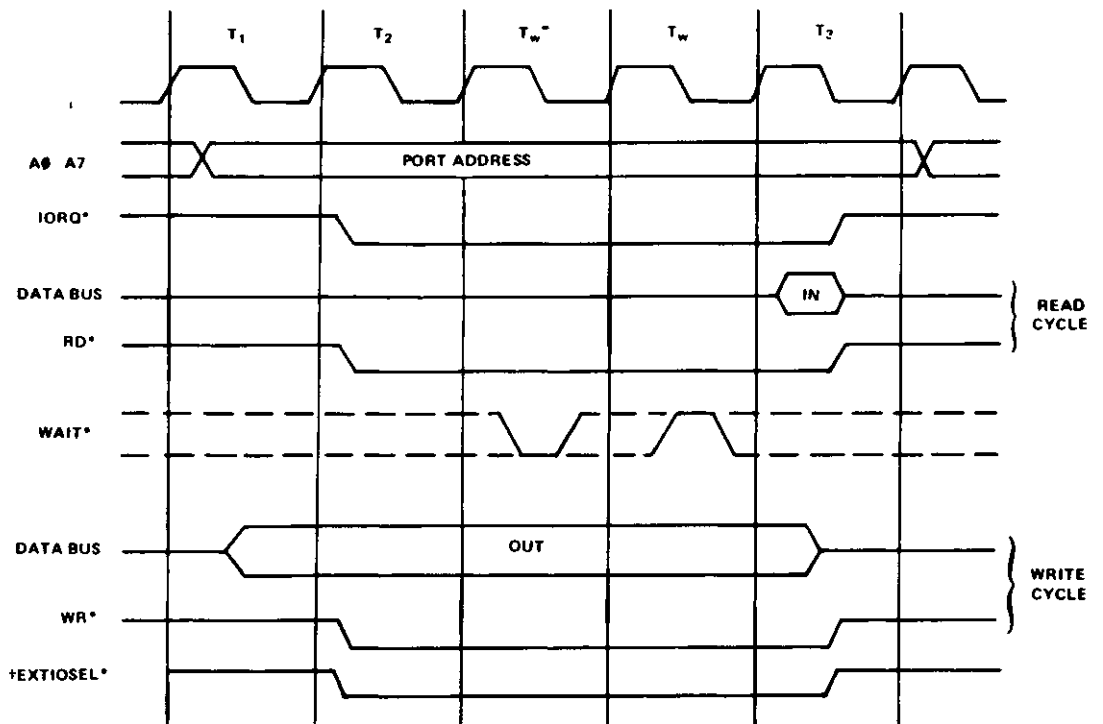
The actual implementation is shown in Figure 2-22. The data is buffered in both directions using a 74LS245 (U101). The addresses are buffered with a 74LS244 (U102) and the control lines out are buffered by a 74LS367. Note that RESET* is always enabled out, this is to power-up reset any device or clear any device before enabling the bus structure. This prevents any user from tying-up the bus when enabling the port in an unknown state.

Input or Output Cycles.



*Inserted by Z80 CPU

Input or Output Cycles with Wait States.



*Inserted by Z80 CPU

†Coincident with IORQ* only on INPUT cycle.

Figure 2-21. I/O BUS TIMING DIAGRAM

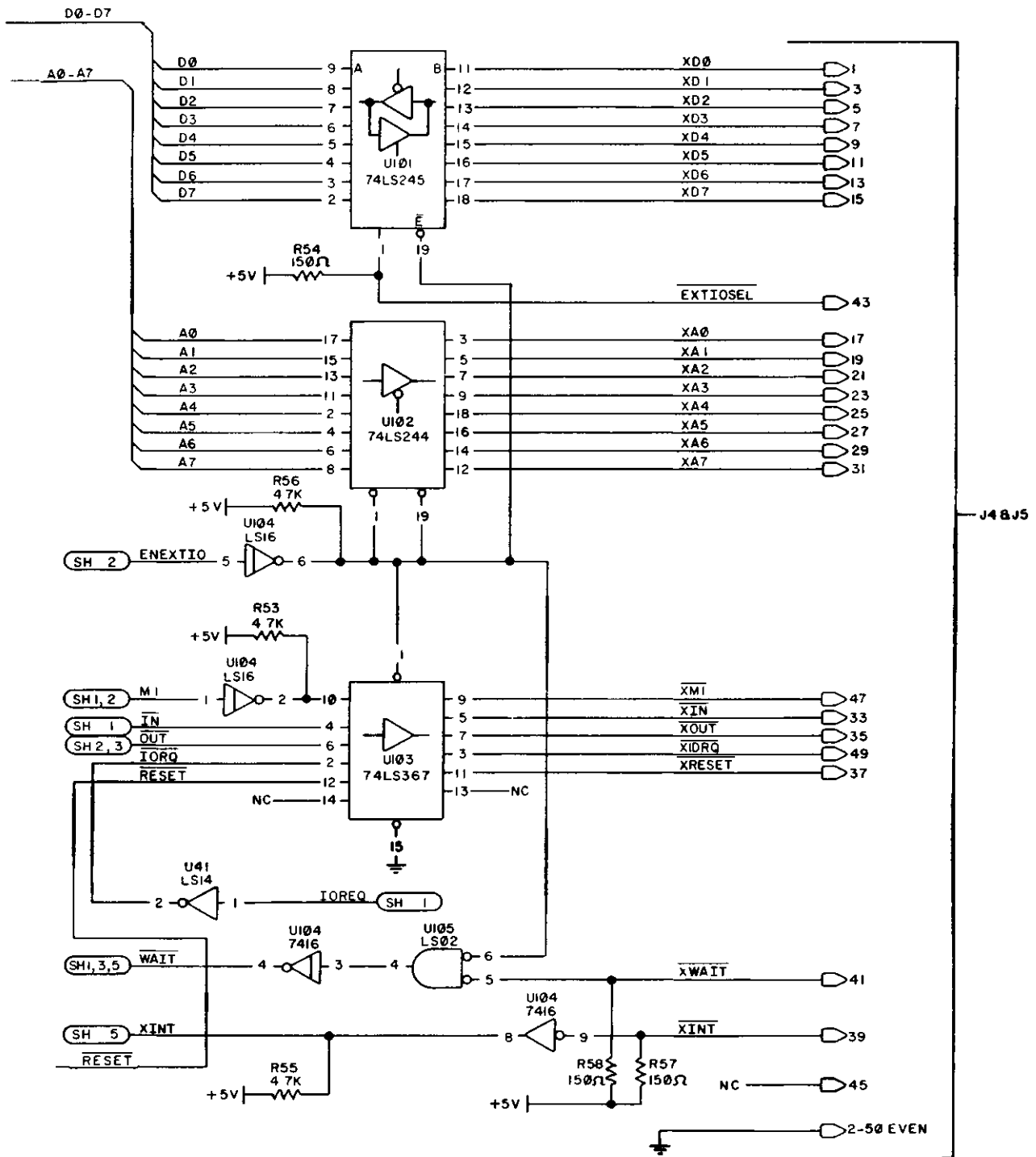


Figure 2-22. I/O Port
(Page 4 of Schematic)

Data Bit	Function
D0	Selects Drive 0 when set*
D1	Selects Drive 1 when set*
D2	Selects Drive 2 when set*
D3	Selects Drive 3 when set*
D4	Selects Side 0 when reset Selects Side 1 when set
D5	Write precompensation enabled when set dis- abled when reset
D6	Generates WAIT if set
D7	Selects MFM mode if set Selects FM mode if reset

*Only one of these bits should be set per output

Hex D flip-flop U79 (74L174) latches the drive select bits, side select and FM* MFM bits on the rising edge of the control signal DRVSEL*. Gate Array 4 4(U76) is used to latch the Wait Enable and Write precompensation enable bits on the rising edge of DRVSEL*. The rising edge of DRVSEL* also triggers a one-shot (Internal to U76) which produces a Motor On to the disk drives. The duration of the Motor On signal is approximately three seconds. The spindle motors are not designed for continuous operation. Therefore, the inactive state of the Motor On signal is used to clear the Drive Select Latch, which de-selects any drives which were previously selected. The Motor On one-shot is retriggerable by simply executing another OUT instruction to the Drive Select Latch.

Wait State Generation and WAITIMOUT Logic

As previously mentioned, a wait state to the CPU can be initiated by an OUT to the Drive Select Latch with D6 set. Pin 10 of U76 will go high after this operation. This signal is inverted by 1/4 of U96 and is routed to the CPU where it forces the Z80A into a wait state. The Z80A will remain in the wait state as long as WAIT* is low. Once initiated, the WAIT* will remain low until one of five conditions is satisfied. If INTRQ, DRQ, or RESET inputs become active (logic high), it causes WAIT* to go high which allows the Z80 to exit the wait state. An internal timer on U70 serves as a watchdog timer to insure that a wait condition will not persist long enough to destroy dynamic RAM contents. This internal watchdog timer logic will limit the duration of a wait to 1024 μ sec, even if the FDC chip should fail to generate a DRQ or an INTRQ.

If an OUT to Drive Select Latch is initiated with D6 reset (logic low), a WAIT is still generated. The internal timer on U70 will count to 2 which will clear the WAIT state. This allows the WAIT to occur only during the OUT instruction to prevent violating any Dynamic RAM parameters.

NOTE This automatic WAIT will cause a 5 to 1 μ sec wait each time an out to Drive Select Latch is performed.

Clock Generation Logic

A 16 MHz crystal oscillator and Gate Array 4 4 (U76) are used to generate the clock signals required by the FDC board. The 16 MHz oscillator is implemented internal to U76 and a quartz crystal (Y2). The output of the oscillator is divided by 2 to generate an 8 MHz clock. This is used by the FDC 1773 (U75) for all internal timing and data separation. U76 further divides the 16 MHz clock to drive the watchdog timer circuit.

Disk Bus Output Drivers

High current open collector drivers U96, 94 and 93 are used to buffer the output signals from the FDC circuit to the disk drives.

Write Precompensation and Write Data Pulse Shaping Logic

All write precompensation is generated internal to the FDC chip 1773 (U75). Write Precompensation occurs when WG goes high and write precompensation is enabled from the software. ENP is multiplexed with RDY and is controlled by WG at pin 20 of U75. Write data is output on pin 22 of U75 and is shaped by a one-shot (1/2 of U98) which stretches the data pulses to approximately 500 nsec.

Clock and Read Data Recovery Logic

The Clock and Read Data Recovery Logic is done internal to the 1773 (U75).

Floppy Disk Controller Chip

The 1773 is an MOS LSI device which performs the functions of a floppy disk formatter/controller in a single chip implementation. The following port addresses are assigned to the internal registers of the 1773 FDC chip:

Port No.	Function
F0H	Command/Status Register
F1H	Track Register
F2H	Sector Register
F3H	Data Register

2.1.15 Cassette Circuit

The cassette write circuitry latches the two LSBs (D0 and D1) for any output to port FF (hex). The outputs of these latches (U51) are then resistor summed to provide three discrete voltage levels (500 Baud only). The firmware toggles the bits to provide an output signal of the desired frequency at the summing node.

There are two types of cassette Read circuits — 500 baud and 1500 baud. The 500 baud circuit is compatible with both Model I and III. The input signal is amplified and filtered by Op amps (U23 and U54). Part of U22 then forms a Zero Crossing Detector, the output of which sets the latch U37. A read of Port FF enables buffer U52 which allows the CPU to determine whether the latch has been set, and simultaneously resets the latch. The firmware determines by the timing between settings of the latch whether a logic "one" or "zero" was read in from the tape.

The 1500 baud cassette read circuit is compatible with the Model III cassette system. The incoming signal is compared to a threshold by part of U22. U22's output will then be either high or low and clock about one-half of U37, depending on whether it is a rising edge or a falling edge. If interrupts are enabled, the setting of either latch will generate an interrupt. As in the 500 baud circuit, the firmware decodes the interrupts into the appropriate data.

For any cassette read or write operation, the cassette relay must be closed in order to start the motor of the cassette deck. A write to port EC hex with bit one set will latch U53, which turns on transistor Q3 and energizes the relay K1. A subsequent write to this port with bit one clear will clear the latch and de-energize the relay.

2.1.16 FDC Circuit

The TRS-80 Model 4 Floppy Disk Interface provides a standard 5-1/4" floppy disk controller. The Floppy Disk Interface supports single and double density encoding schemes. Write precompensation can be software enabled or disabled beginning at any track, although the system software enables write precompensation for all tracks greater than twenty-one. The amount of write precompensation is 125 nsec and is not adjustable. One to four drives may be controlled by the interface. All data transfers are accomplished by CPU data requests. In double density operation, data transfers are synchronized to the CPU by forcing a wait to the CPU and clearing the wait by a data request from the FDC chip. The end of the data transfer is indicated by generation of a non-maskable interrupt from the interrupt request output of the FDC chip. A hardware watchdog timer insures that any error condition will not hang the wait line to the CPU for a period long enough to destroy RAM contents.

Control and Data Buffering

The Floppy Controller is an I/O port-mapped device which utilizes ports E4H, F0H, F1H, F2H, F3H, and F4H. The decoding logic is implemented in the Address Decoding (for more information see Port Map). U78 is a bi-directional, 8-bit transceiver used to buffer data to and from the FDC and RS-232 circuits. The direction of data transfer is controlled by the combination of control signals DISKIN*, RDINTSTATUS*, RDNINSTATUS*, and RS232IN*. If any signal is active (logic low), U78 is enabled to drive data onto the CPU data bus. If all signals are inactive (logic high), U78 is enabled to receive data from the CPU board data bus. A second buffer U77 is used to buffer the FDC chip data to the FDC/RS232 Data Bus, (BD0-BD7). U77 is enabled by Chip Select and its direction controlled by DISKIN*. Again, if DISKIN* is active (logic low), data is enabled to drive from the FDC chip to the Main Data Busses. If DISKIN* is inactive (logic high), data is enabled to be transferred to the FDC chip.

Non-maskable Interrupt Logic

Gate Array 4 4 (U75) is used to latch data bits D6 and D7 on the rising edge of the control signal WRNMIMASKREG*. This enables the conditions which will generate a non-maskable interrupt to the CPU. The NMI interrupt conditions which are programmed by doing an OUT instruction to port E4H with the appropriate bits set. If data bit 7 is set, an FDC interrupt is enabled to generate an NMI interrupt. If data bit 7 is reset, interrupt requests from the FDC are disabled. If data bit 6 is set, a Motor Time Out is enabled to generate an NMI interrupt. If data bit 6 is reset, interrupts on Motor Time Out are disabled. An IN instruction from port E4H enables the CPU to determine the course of the non-maskable interrupt. Data bit 7 indicates the status of FDC interrupt request (INTRQ) (0=true, 1=false). Data bit 6 indicates the status of Motor Time Out (0=true, 1=false). Data bit 5 indicates the status of the Reset signal (0=true, 1=false). The control signal RDNMISTATUS* gates this status onto the CPU data bus when active (logic low).

Drive Select Latch and Motor ON Logic

Selecting a drive prior to disk I/O operation is accomplished by doing an OUT instruction to port F4H with the proper bit set. The following table describes the bit allocation of the Drive Select Latch.

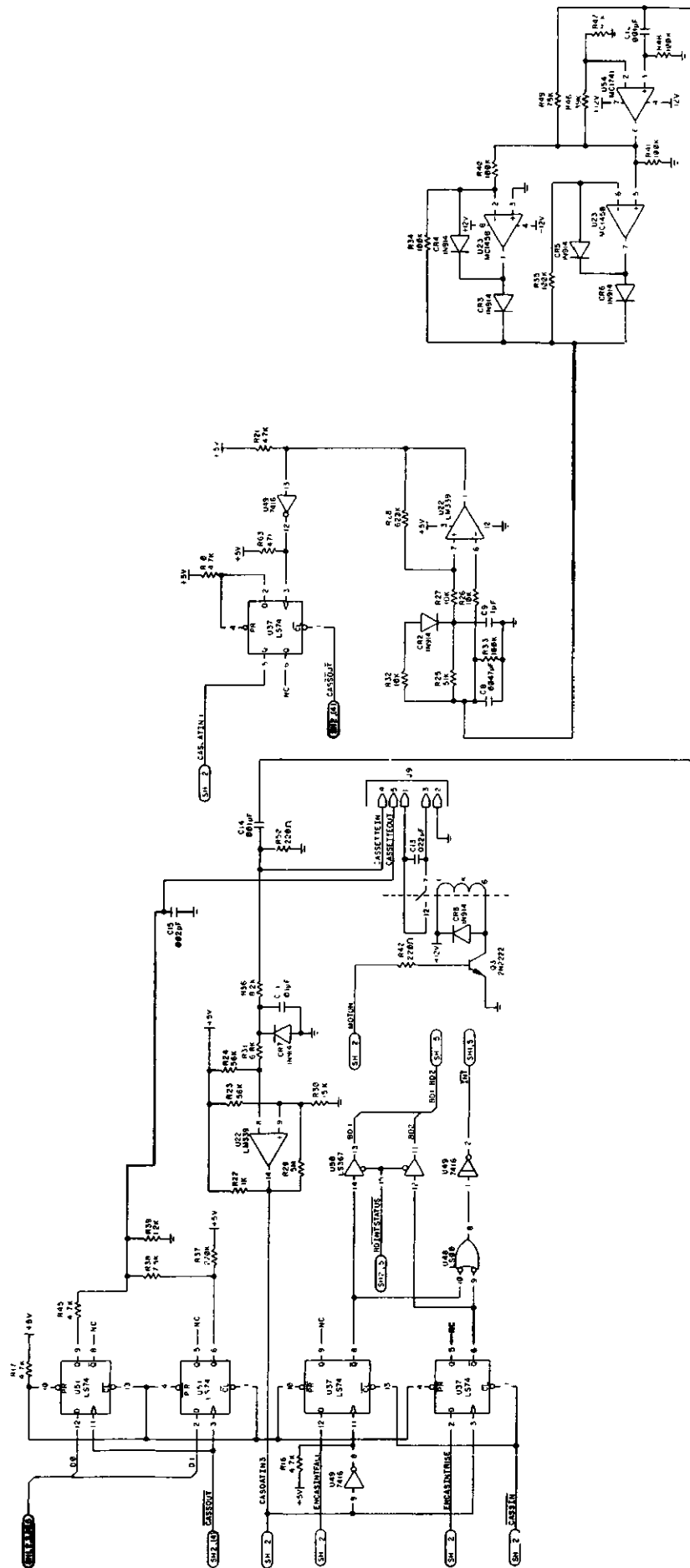


Figure 2-23. Circuit Cassette
(Page 4 of Schematic)

RS-232C Technical Description

The RS-232C circuit for the Model 4 computer supports asynchronous serial transmissions and conforms to the EIA RS-232C standards at the input-output interface connector (J3). The heart of the circuit is the TR1865 Asynchronous Receiver/Transmitter U84. It performs the job of converting the parallel byte data from the CPU to a serial data stream including start, stop, and parity bits. For a more detailed description of how this LSI circuit performs these functions, refer to the TR1865 data sheets and application notes. The transmit and receive clock rates that the TR1865 needs are supplied by the Baud Rate Generator U104. This circuit takes the 5.0688 MHz supplied by the system timing circuit and the programmed information received from the CPU over the data bus and divides the basic clock rate to provide two clocks. The rates available from the BRG go from 50 Baud to 19200 Baud. See the BRG table for the complete list.

Interrupts are supported in the RS-232C Circuit by the Interrupt mask register and the Status register internal to Gate Array 4.5 (U82). The CPU looks here to see which kind of interrupt has occurred. Interrupts can be generated on receiver data register full, transmitter register empty, and any one of the errors — parity, framing, or data overrun. This allows a minimum of CPU overhead in transferring data to or from the UART. The interrupt mask register is port E0 (write) and the interrupt status register is port E0 (read). Refer to the IO Port description for a full breakdown of all interrupts and their bit positions.

All Model I, III, and 4 software written for the RS-232C interface is compatible with the Model 4 Gate Array RS-232C circuit, provided the software does not use the sense switches to configure the interface. The programmer can get around this problem by directly programming the BRG and UART for the desired configuration or by using the SETCOM command of the disk operating system to configure the interface. The TRS-80 RS-232C Interface hardware manual has a good discussion of the RS-232C standard and specific programming examples (Catalog Number 26-1145).

BRG Programming Table

Nibble Loaded	Transmit/Receive Baud Rate	16X Clock	Supported by SETCOM
0H	50	0.8 kHz	Yes
1H	75	1.2 kHz	Yes
2H	110	1.76 kHz	Yes
3H	134.5	2.1523 kHz	Yes
4H	150	2.4 kHz	Yes
5H	300	4.8 kHz	Yes
6H	600	9.6 kHz	Yes
7H	1200	19.2 kHz	Yes
8H	1800	28.8 kHz	Yes
9H	2000	32.081 kHz	Yes
AH	2400	38.4 kHz	Yes
BH	3600	57.6 kHz	Yes
CH	4800	76.8 kHz	Yes
DH	7200	115.2 kHz	Yes
EH	9600	153.6 kHz	Yes
FH	19200	307.2 kHz	Yes

Pinout Listing

The RS-232C circuit is port mapped and the ports used are E8 to EB. Following is a description of each port on both input and output.

Port	Input	Output
E8	Modem status	Master Reset, enables UART control register load
EA	UART status	UART control register load and modem control
E9	Not Used	Baud rate register load enable bit
EB	Receiver Holding register	Transmitter Holding register

The following list is a pinout description of the DB-25 connector (P1).

Pin No.	Signal
1	PGND (Protective Ground)
2	TD (Transmit Data)
3	RD (Receive Data)
4	RTS (Request to Send)
5	CTS (Clear To Send)
6	DSR (Data Set Ready)
7	SGND (Signal Ground)
8	CD (Carrier Detect)
19	SRTS (Spare Request to Send)
20	DTR (Data Terminal Ready)
22	RI (Ring Indicate)

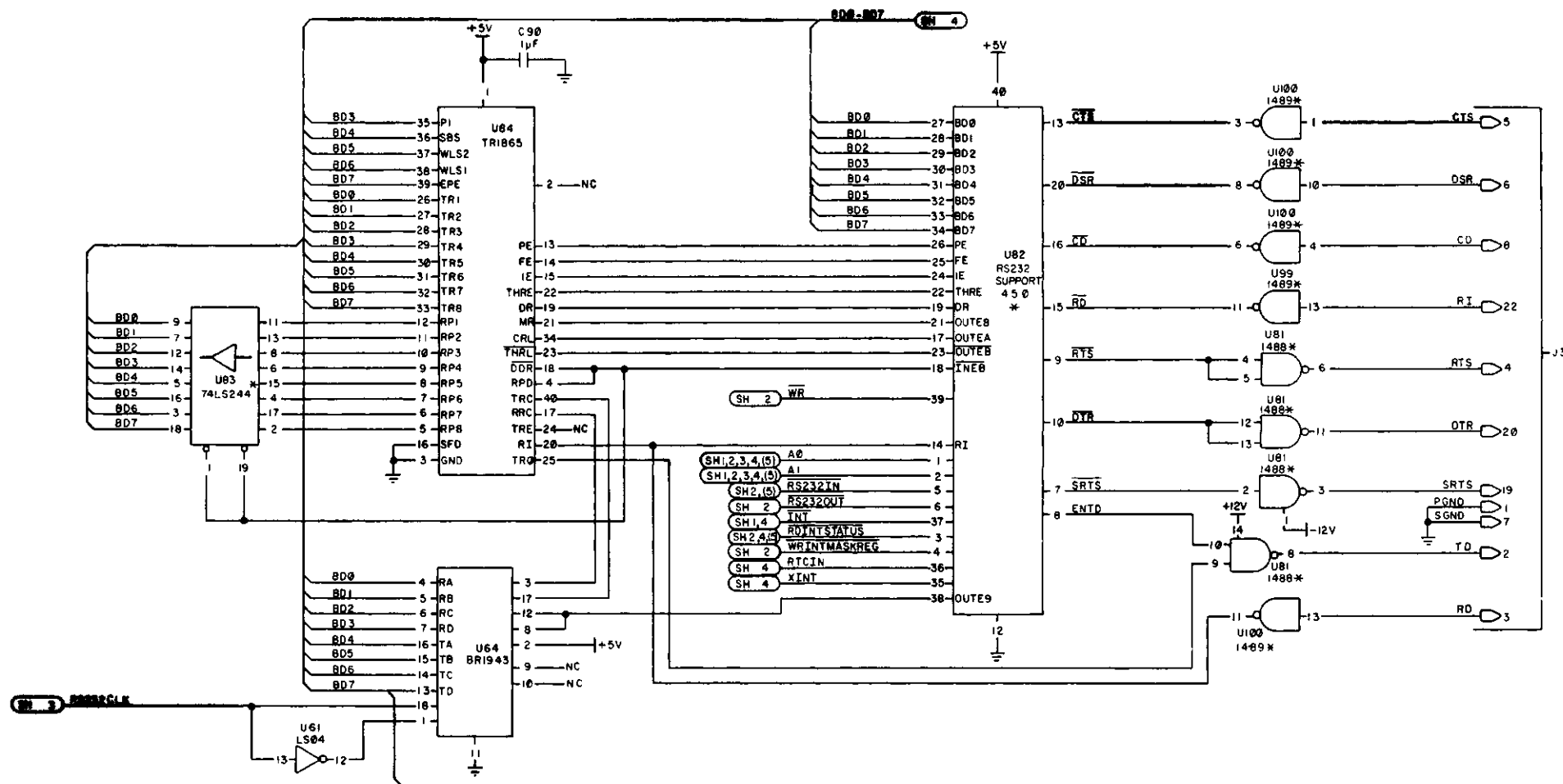


Figure 2-25. RS232C Circuit

(Page 5 of Schematic)

**Model 4 Gate Array
I/O Pin Assignments**

J1		J2		J3		J4		J5	
Pin No.	Signal	Pin No.	Signal	Pin No.	Signal	Pin No.	Signal	Pin No.	Signal
1.	GND	1.	GND	1.	PGND	1.	XDO	1.	XDO
2.		2.		2.	TD	2.	GND	2.	GND
3.	GND	3.	GND	3.	RD	3.	XD1	3.	XD1
4.		4.		4.	RTS	4.	GND	4.	GND
5.	GND	5.	GND	5.	CTS	5.	XD2	5.	XD2
6.		6.		6.	DSR	6.	GND	6.	GND
7.	GND	7.	GND	7.	SGND	7.	XD3	7.	XD3
8.	IPE*	8.	IPI*	8.	CD	8.	GND	8.	GND
9.	GND	9.	GND	9.		9.	XD4	9.	XD4
10.	DS2*	10.	DS0*	10.		10.	GND	10.	GND
11.	GND	11.	GND	11.		11.	XD5	11.	XD5
12.	DS3*	12.	DS1*	12.		12.	GND	12.	GND
13.	GND	13.	GND	13.		13.	XD6	13.	XD6
14.		14.		14.		14.	GND	14.	GND
15.	GND	15.	GND	15.		15.	XD7	15.	XD7
16.	MOTNE*	16.	MOTONI*	16.		16.	GND	16.	GND
17.	GND	17.	GND	17.		17.	XA0	17.	XA0
18.	DIRE*	18.	DIRI*	18.		18.	GND	18.	GND
19.	GND	19.	GND	19.	SRTS	19.	XA1	19.	XA1
20.	STEPE*	20.	STEPI*	20.	DTR	20.	GND	20.	GND
21.	GND	21.	GND	21.		21.	XA2	21.	XA2
22.	WDE*	22.	WDI*	22.	RI	22.	GND	22.	GND
23.	GND	23.	GND	23.		23.	XA3	23.	XA3
24.	WGE*	24.	WGI*	24.		24.	GND	24.	GND
25.	GND	25.	GND	25.		25.	XA4	25.	XA4
26.	TRKOE*	26.	TRKOI*	26.		26.	GND	26.	GND
27.	GND	27.	GND	27.		27.	XA5	27.	XA5
28.	WPRTE*	28.	WPRTI*	28.		28.	GND	28.	GND
29.	GND	29.	GND	29.		29.	XA6	29.	XA6
30.	RDE*	30.	RDI*	30.		30.	GND	30.	GND
31.	GND	31.	GND	31.		31.	XA7	31.	XA7
32.	SDSELE	32.	SDSELI	32.		32.	GND	32.	GND
33.	GND	33.	GND	33.		33.	XIN*	33.	XIN*
34.		34.		34.		34.	GND	34.	GND
						35.	XOUT*	35.	XOUT*
						36.	GND	36.	GND
						37.	XRESET*	37.	XRESET*
						38.	GND	38.	GND
						39.	XINT*	39.	XINT*
						40.	GND	40.	GND
						41.	XWAIT*	41.	XWAIT*
						42.	GND	42.	GND
						43.	EXTIO-	43.	EXTIO-
						44.	SEL*	44.	SEL*
						45.	GND	45.	GND
						46.	NC	46.	NC
						47.	GND	47.	GND
						48.	XMI*	48.	XMI0
						49.	GND	49.	GND
						50.	XIDRQ*	50.	XIDRQ*
							GND		GND

Model 4 Gate Array
I/O Pin Assignments

J6		J8		J9		J12	
Pin No.	Signal	Pin No.	Signal	Pin No.	Signal	Pin No.	Signal
1.		1.		1.		1.	D0
2.	GND	2.		2.	GND	2.	D1
3.	PD0	3.		3.		3.	D2
4.	GND	4.	VSYNCO*	4.	CASSETTE-	4.	D3
5.	PD1	5.		5.	IN	5.	D4
6.	GND	6.	HSYNCO	6.	CASSETTE-	6.	D5
7.	PD2	7.		7.	OUT	7.	D6
8.	GND	8.		8.		8.	D7
9.	PD3	9.		9.		9.	GEN*
10.	GND	10.		10.		10.	DCLK
11.	PD4	11.		11.		11.	A0
12.	GND	12.		12.		12.	A1
13.	PD5	13.		13.		13.	A2
14.	GND	14.		14.		14.	J
15.	PD6	15.		15.		15.	GRAPVID
16.	GND	16.		16.		16.	ENGRAF
17.	PD7	17.		17.		17.	DISBEN
18.	GND	18.		18.		18.	VSYNCO
19.	N/A	19.		19.		19.	HSYNCO
20.	GND	20.		20.		20.	RESET*
21.	BUSY	21.		21.		21.	WAIT*
22.	GND	22.		22.		22.	H
23.	OUT PAPER	23.		23.		23.	I
24.	GND	24.		24.		24.	IN*
25.	UNIT SEL					25.	GND
26.	NC					26.	+5V
27.	GND					27.	
28.	FAULT					28.	CL166
29.						29.	GND
30.						30.	+5V
31.	NC					31.	GND
32.						32.	+5V
33.	NC					33.	GND
34.	GND					34.	+5V

SECTION III

4P THEORY OF OPERATION

3.1 MODEL 4P THEORY OF OPERATION

3.1.1 Introduction

Contained in the following paragraphs is a description of the component parts of the Model 4P CPU. It is divided into the logical operational functions of the computer. All components are located on the Main CPU board inside the case housing. Refer to Section 3 for disassembly assembly procedures.

3.1.2 Reset Circuit

The Model 4P reset circuit provides the necessary reset pulses to all circuits during power up and reset operations. R25 and C218 provide a time constant which holds the input of U121 low during power-up. This allows power to be stable to all circuits before the RESET* and RESET signals are applied. When C218 charges to a logic high, the output of U121 triggers the input of a retriggerable one-shot multivibrator (U1). U1 outputs a pulse with an approximate width of 70 microseconds. When the reset switch is pressed on the front panel, this discharges C218 and holds the input of U121 low until the switch is released. On release of the switch, C218 again charges up, triggering U121 and U1 to reset the microcomputer.

3.1.3 CPU

The central processing unit (CPU) of the Model 4P microcomputer is a Z80A microprocessor. The Z80A is capable of running in either 2 MHz or 4 MHz mode. The CPU controls all functions of the microcomputer through use of its address lines (A0-A15), data lines (D0-D7) and control lines (M1, IOREQ, RD, WR, MREQ, and RFSH). The address lines (A0-A15) are buffered to other ICs through two 74LS244s (U68 and U26) which are enabled all the time with their enables pulled to GND. The control lines are buffered to other ICs through a 74F04 (U86). The data lines (D0-D7) are buffered through a bi-directional 74LS245 (U71) which is enabled by BUSEN* and the direction is controlled by BUSDIR*.

3.1.4 System Timing

The main timing reference of the microcomputer, with the exception of the FDC circuit, comes from a 20.2752 MHz Crystal Oscillator (Y1). This reference is divided and used for generating all necessary timing for the CPU, video circuit, and RS-232-C circuit. The output of the crystal oscillator is filtered by a ferrite bead (FB5), 470 ohm resistor (R46), and a 68 pF capacitor (C242). After being filtered, it is fed into U126, a 16R6A PAL (Programmable Array Logic), where it is divided by 2 to generate a 10.1376 MHz signal (10M) for the 64 X 16 video display. U126 divides the 20.2752 MHz by 4 to generate a 5.0688 MHz signal (RS232CLK) for the baud rate generator in the RS-232-C circuit. The CPU clock is also generated by U126 which can be either 2 or 4 MHz depending on the state of FAST input

(pin 9 of U126). If FAST is a logic low, the 20.2752 MHz is divided by 10 which generates a 2.02752 MHz signal. If FAST is a logic high, the 20.2752 MHz is divided by 5 which generates a 4.05504 MHz signal. The CPU clock (PCLK) is fed through an active pull-up circuit which generates a full 5-volt swing with fast rise and fall times required by the Z80A. U126, the 16R6A PAL, generates all symmetrical output signals and also does not allow the PCLK output to short cycle or generate a low or high pulse under 110 nanoseconds which the Z80A also requires. Refer to System Timing Fig. 3-2.

3.1.4.1 Video Timing

The video timing is controlled by a 10L8 PAL (U127) and a four-bit synchronous counter U128 (74LS161). These two ICs generate all the necessary timing signals for the four video modes: 64 x 16, 32 x 16, 80 x 24, and 40 x 24. Two reference clock signals are required for the four video modes. One reference clock, the 10.1376 MHz signal (10M), is generated by U126 and is used by the 64 x 16 and 32 x 16 modes. The second reference clock is a 12.672 MHz (12M) signal which is generated by a Phase Locked Loop (PLL) circuit and is used by the 80 x 24 and 40 x 24 modes. The PLL circuit consists of U147 (74LS93), U148 (NE564 PLL), and U149 (74LS90). The original 20.2752 MHz clock is divided by 16 through U147 which generates a 1.2672 MHz signal. The output of U147 is reduced in amplitude by the voltage divider network R27 and R28 and the output is coupled to the reference input of U148 by C227.

The PLL (NE564) is adjusted to oscillate at 12.672 MHz by the tuning capacitor C231. This 12.672 MHz clock is then divided by 10 through U149 to generate a second 1.2672 MHz signal which is fed to a second input of U148. The two 1.2672 MHz signals are compared internally to the PLL where it corrects the 12.672 MHz output so it is synchronized with the 20.2752 MHz clock.

MODESEL and 8064* signals are used to select the desired video mode. 8064* controls which reference clock is used by U127 and MODESEL controls the single or double character width mode. Refer to the following chart for selecting each video mode.

8064*	MODESEL	Video Mode
0	0	64 x 16
0	1	32 x 16
1	0	80 x 24
1	1	40 x 24

*This is the state to be written to latch U89. Signal is inverted before being input to U127.

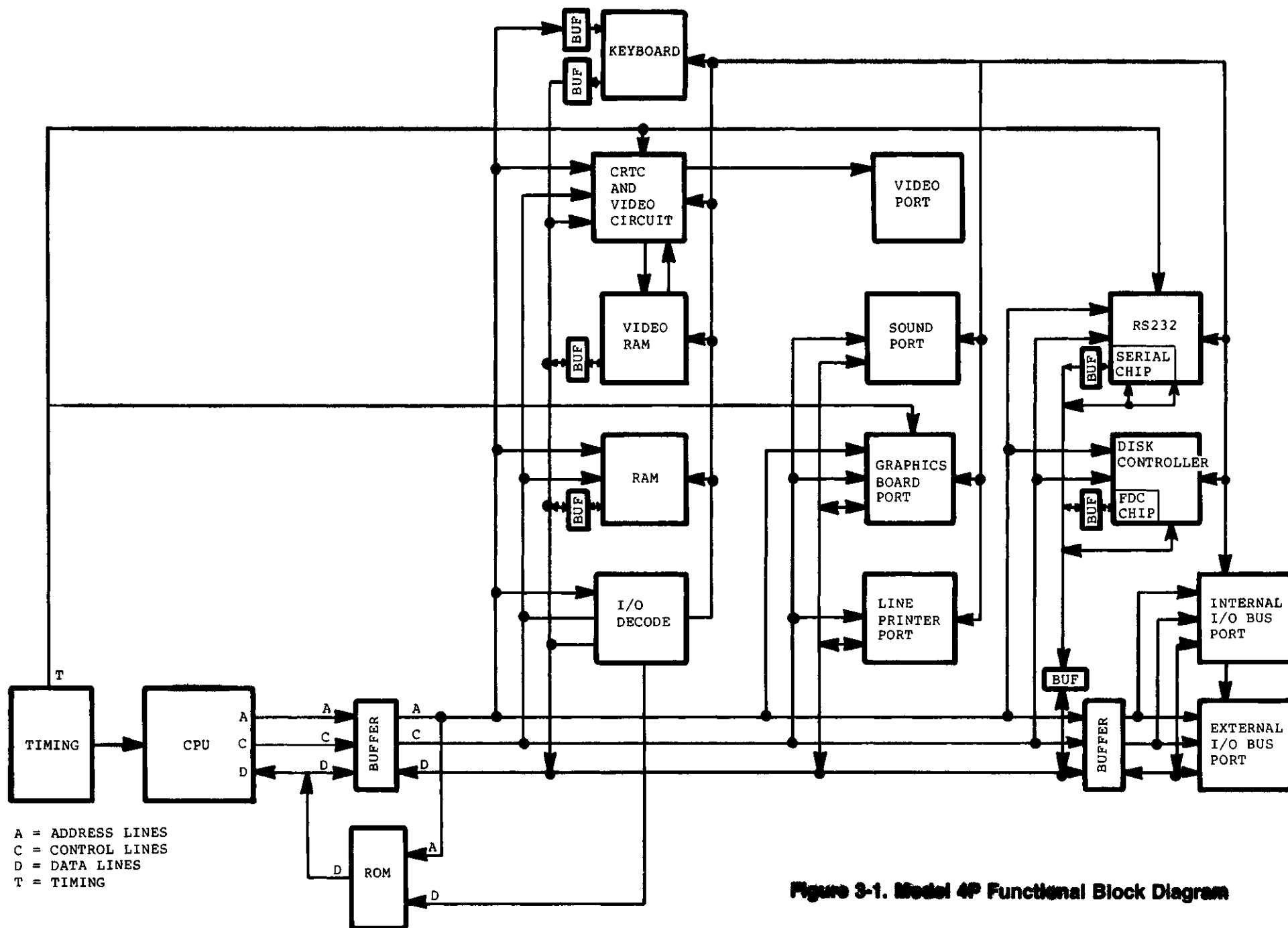


Figure 3-1. Model 4P Functional Block Diagram

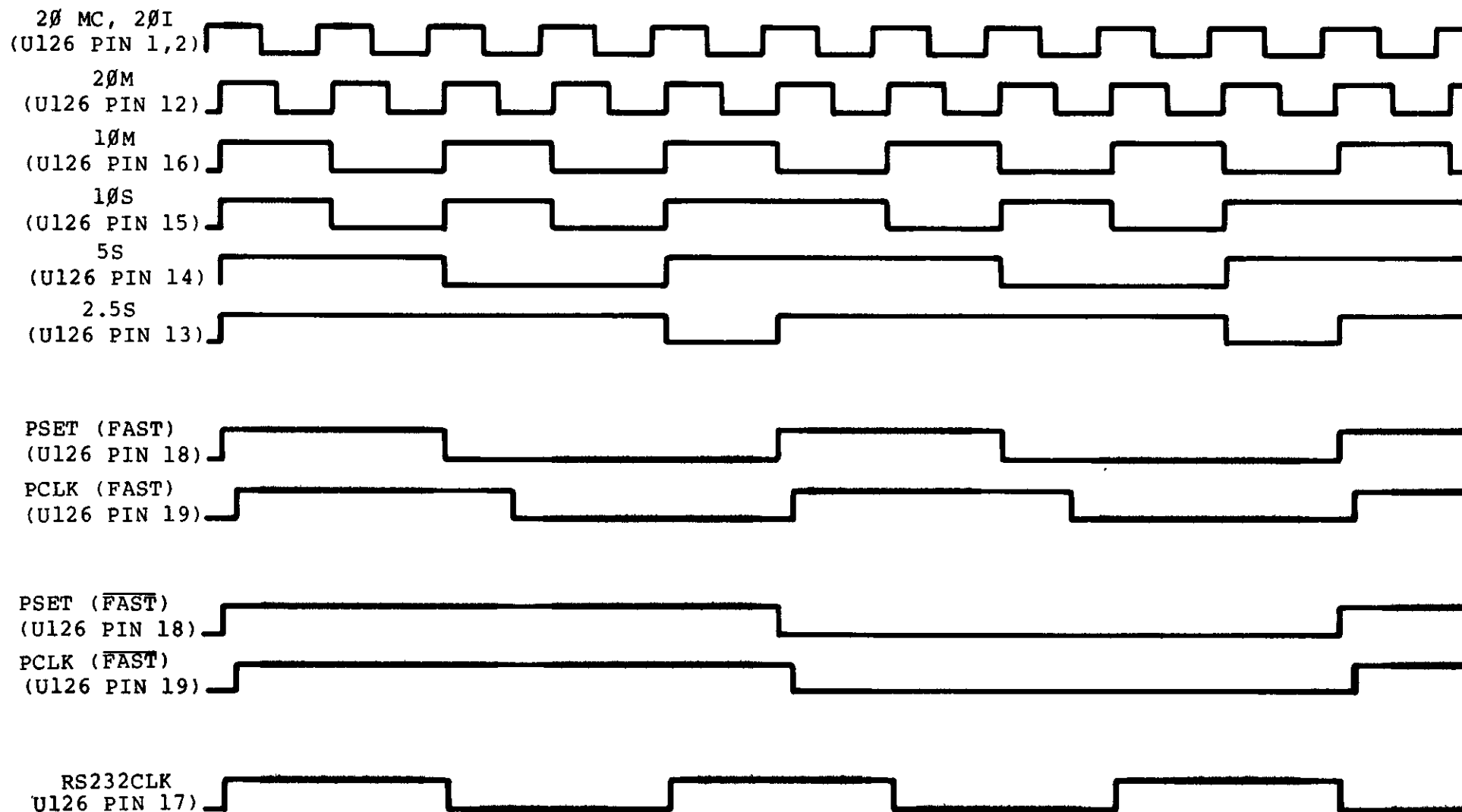


Figure 3-2. System Timing

DCLK, the reference clock selected, is output from U127. DCLK is fed back into U127 for internal timing reference and is also fed to the clock input of U128 (74LS161). U128 is configured to preload with a count of 9 each time it reaches a count of 0. This generates a signal output of TC (128 pin 15) that occurs at the start of every character time of video output. TC is used to generate LOADS* (Load Shift Register). QA and QC of U128 are used to generate SHIFT*, XADDR7*, CRTCLK and LOAD* for proper timing for the four video modes. QA, QB, and QC which are referred to as H, I, and J are fed to the Graphics Port J7 for reference timings of Hires graphics video. Refer to Video Timing, Figs. 3-3 and 3-4 for timing reference.

3.1.5 Address Decode

The Address Decode section will be divided into two sub-sections: Memory Map decoding and Port Map decoding.

3.1.5.1 Memory Map Decoding

Memory Map Decoding is accomplished by a 16L8 PAL (U109). Four memory map modes are available which are compatible with the Model III and Model 4 microcomputers. A second 16L8 PAL (U110) is used in conjunction with U109 for the memory map control which also controls page mapping of the 32K RAM pages. Refer to Memory Maps below.

3.1.5.2 Port Map Decoding

Port Map Decoding is accomplished by three 74LS138s (U87, U88, and U107). These ICs decode the low order address (A0-A7) from the CPU and decode the port being selected. The IN* signal from U108 enables U87 which allows the CPU to read from a selected port and the OUT* signal, also from U108, enables U88 which allows the CPU to write to the selected port. U107 only decodes the address and the IN* and OUT* signals are ANDed with the generated signals.

3.1.6 ROM

The Model 4P contains only a 4K x 8 Boot ROM (U70). This ROM is used only to boot up a Disk Operating System into the RAM memory. If Model III operation or DOS is required, then the RAM from location 0000-37FFH must be loaded with an image of the Model III or 4 ROM code and then executed. A file called MODEL A/III is supplied with the Model 4P which contains the ROM image for proper Model III operation. On power-up, the Boot ROM is selected and mapped into location 0000-0FFFH. After the Boot Sector or the ROM image is loaded, the Boot ROM must be mapped out by OUTing to port 9CH with D0 set or by selecting Memory Map modes 2 or 3. In Mode 1 the RAM is write enabled for the full 14K. This allows the RAM area mapped where Boot ROM is located to be written to while executing out of the Boot ROM. Refer to Memory Maps.

The Model 4P Boot ROM contains all the code necessary to initialize hardware, detect options selected from the keyboard, read a sector from a hard disk or floppy, and load a copy of the Model III ROM Image (as mentioned) into the lower 14K of RAM.

The firmware is divided into the following routines:

- Hardware Initialization
- Keyboard Scanner
- Control
- Floppy and Hard Disk Driver
- Disk Directory Searcher
- File Loader
- Error Handler and Displayer
- RS-232 Boot
- Diagnostic Package

Theory of Operation

This section describes the operation of various routines in the ROM. Normally, the ROM is not addressable by normal use. However, there are several routines that are available through fixed calling locations and these may be used by operating systems that are booting.

On a power-up or RESET condition, the Z80's program counter is set to address 0 and the boot ROM is switched-in. The memory map of the system is set to Mode 0. (See Memory Map for details.) This will cause the Z80 to fetch instructions from the boot ROM.

The Initialization section of the Boot ROM now performs these functions:

1. Disables maskable and non-maskable interrupts
2. Interrupt mode 1 is selected
3. Programs the CRT Controller
4. Initializes the boot ROM control areas in RAM
5. Sets up a stack pointer
6. Issues a Force Interrupt to the Floppy Disk Controller to abort any current activity
7. Sets the system clock to 4mhz
8. Sets the screen to 64 x 16
9. Disables reverse video and the alternate character sets
10. Tests for "key being pressed"
11. Clears all 2K of video memory

* This is a special test. If the "key being pressed" is being pressed, then control is transferred to the diagnostic package in the ROM. All other keys are scanned via the Keyboard Scanner.

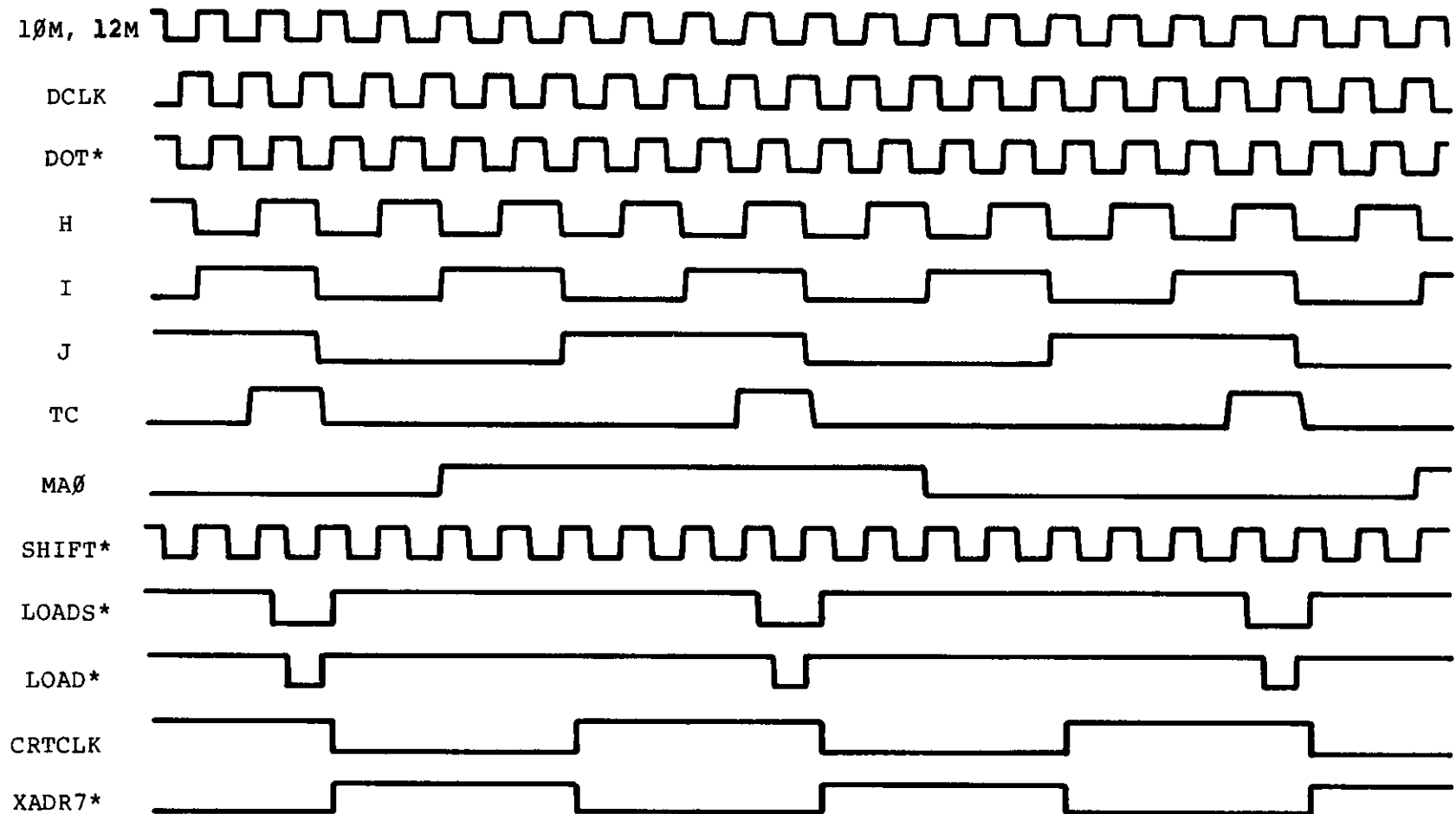


Figure 3-3. Video Timing 64 x 16 Mode 80 x 24 Mode

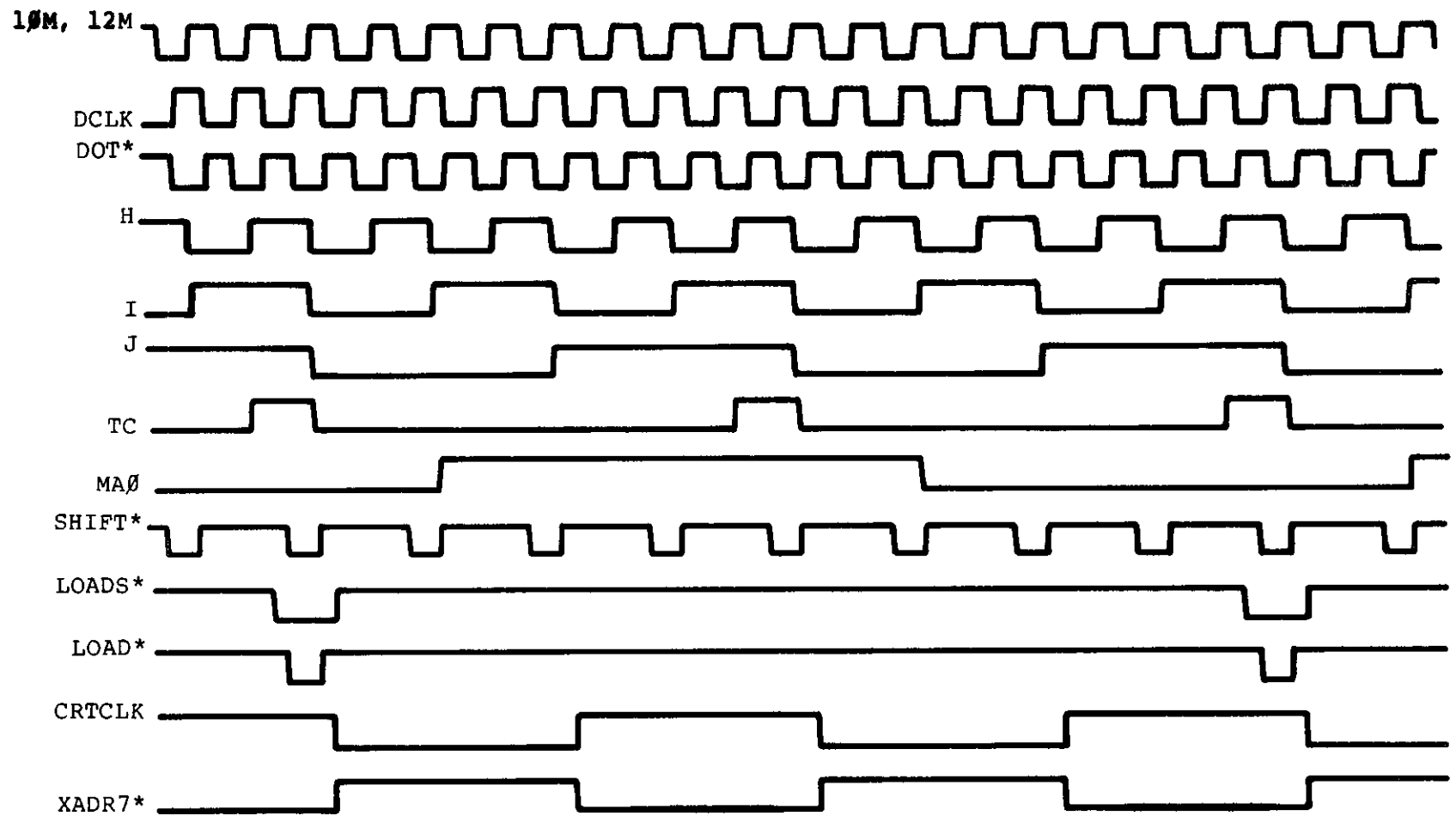


Figure 3-4. Video Timing 32 x 16 Mode 40 x 24 Mode

The Keyboard scanner is now called . It scans the keyboard for a set period of time and returns several parameters based on which (if any) keys were pressed

The keyboard scanner checks for several different groups of keys. These are shown below

Function Group	Selection Group
F1	A
<F2>	B
<F3>	C
<1>	D
<2>	E
<3>	F
<Left-Shift>	G
<Right-Shift>	
<Ctrl>	
<Caps>	
Special Keys	Misc Keys
<P>	Enter
<L>	Break
<N>	

When any key in the Function Group is pressed, it is recorded in RAM and will be used by the Control routine in directing the action of the boot. If more than one of these keys are pressed during the keyboard scan, the last one detected will be the one that is used. The Function group keys are currently defined as

<F1> or <1>	Will cause hard disk boot
<F2> or <2>	Will cause floppy disk boot
<F3> or <3>	Will force Model III mode
<Left-Shift>	Reserved for future use
<Right-Shift>	Boot from RS-232 port
<Ctrl>	Reserved for future use
<Caps>	Reserved for future use

The Special keys are commands to the Control routine which direct handling of the Model III ROM-image. Each key is detected individually

<P>	When loading the Model III ROM image, the user will be prompted when the disks can be switched or when ROM BASIC can be entered by pressing Break
<N>	Instructs the Control routine to not load the Model III ROM image even if it appears that the operating system being booted requires it

L

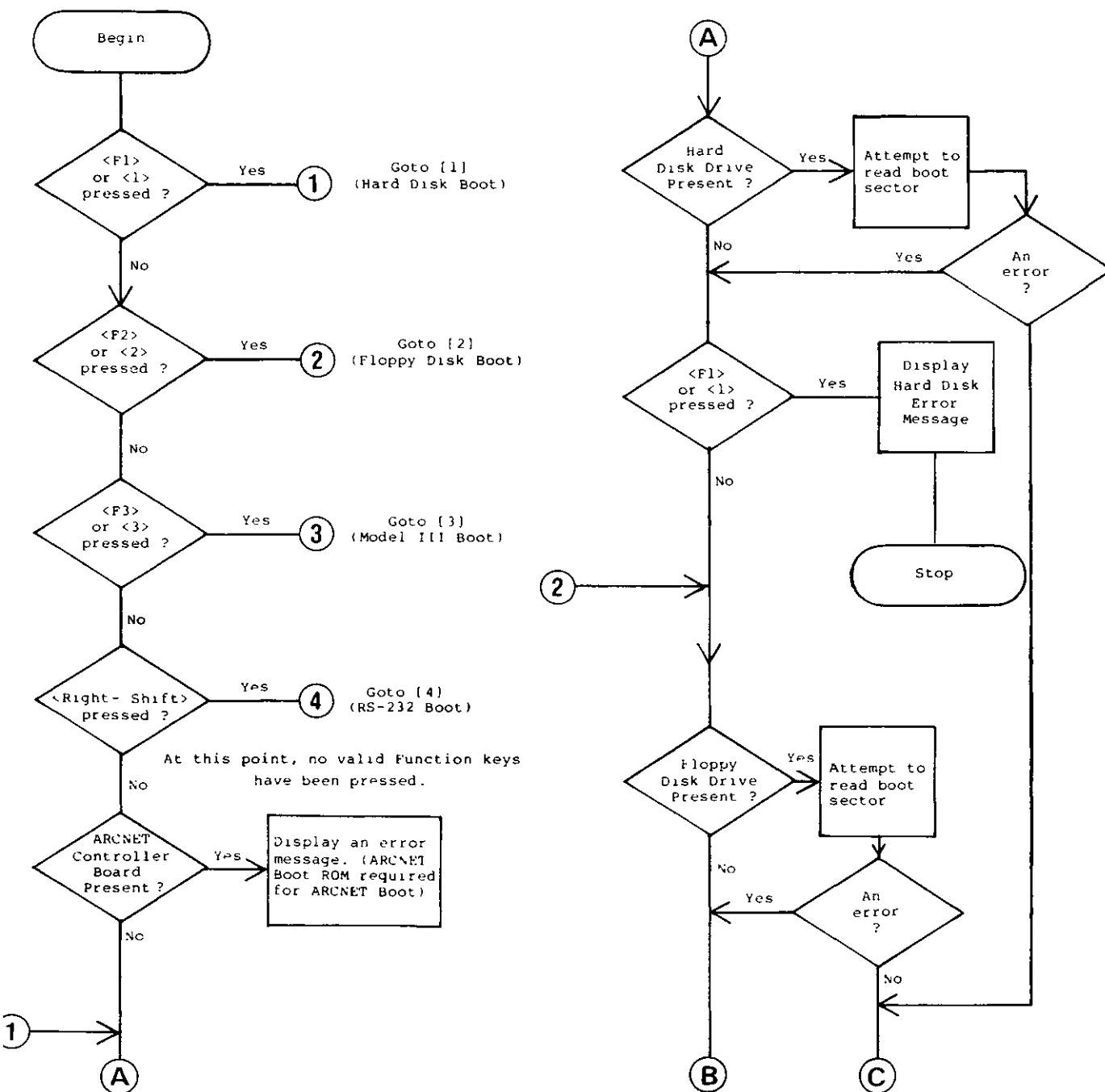
Instructs the Control routine to load the Model III ROM image even if it is already loaded. This is useful if the ROM image has been corrupted or when switching ROM images. (Note that this will not cause the ROM image to be loaded if the boot sector check indicates that the Model III ROM image is not needed. Press F3 or F3 and L to accomplish that

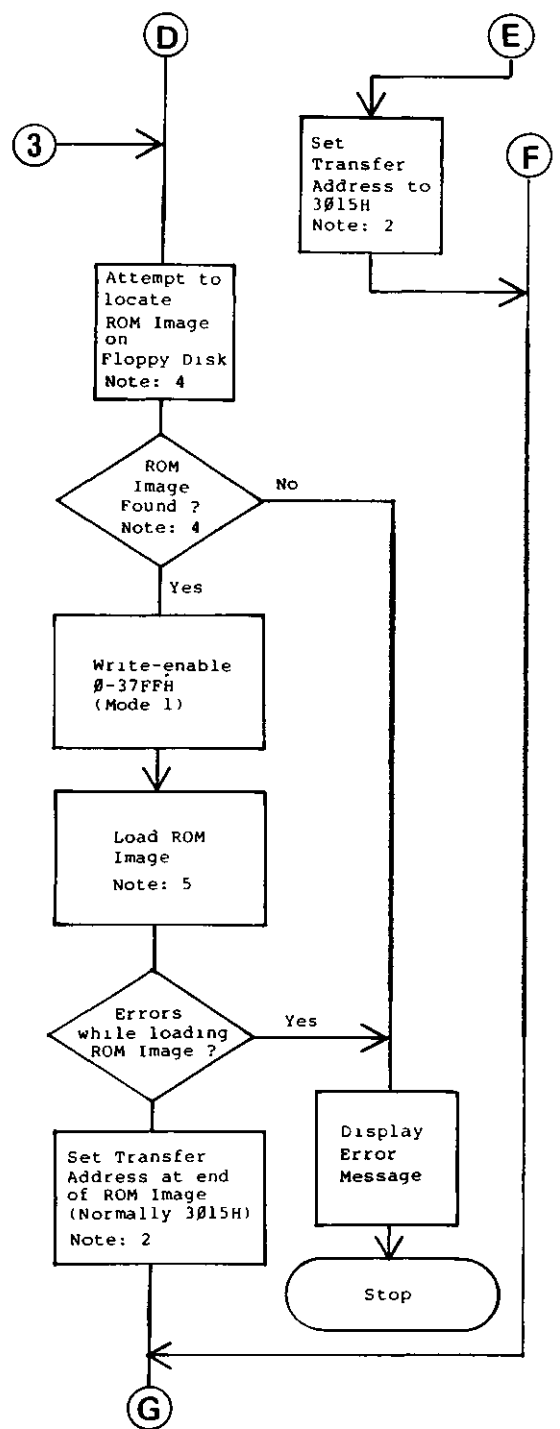
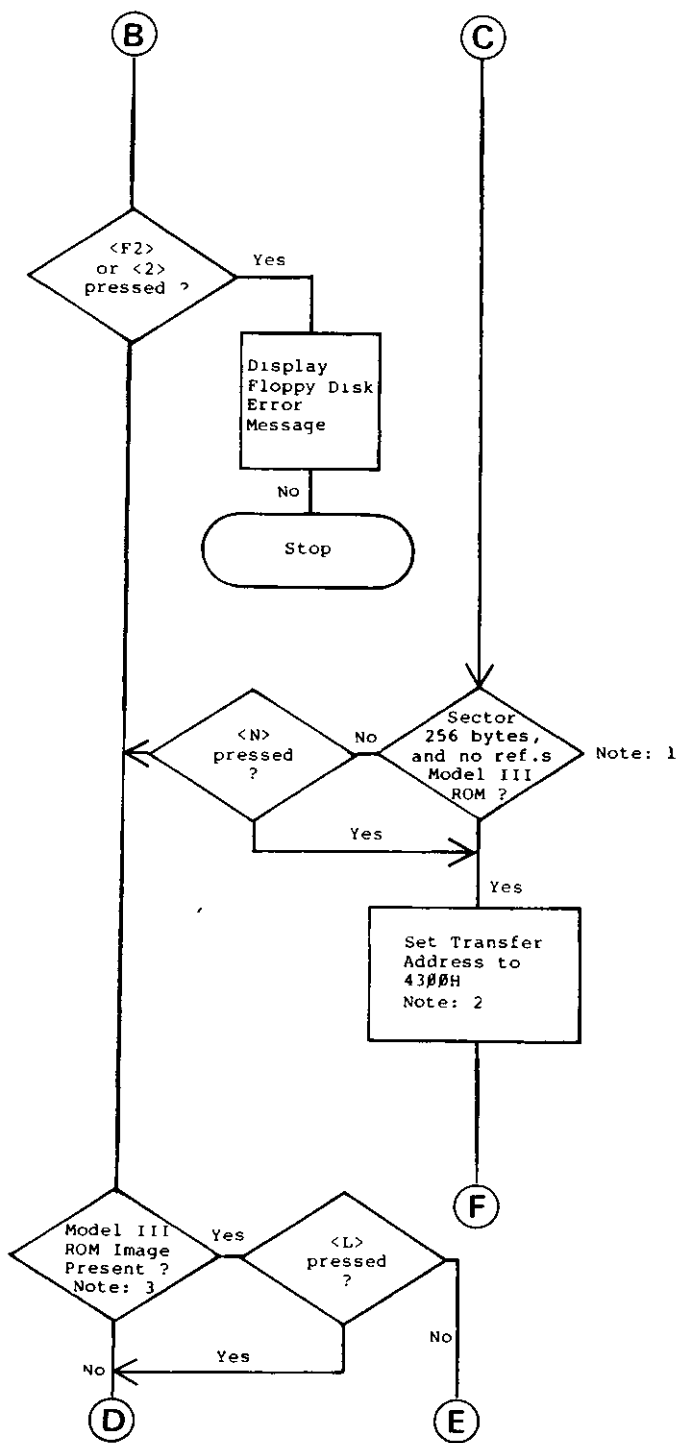
The Selection group keys are used in determining which file will be read from disk when the ROM image is loaded. For details of this operation, see the Disk Directory Searcher. If more than one of the Selection group keys are pressed, the last one detected will be the one that is used.

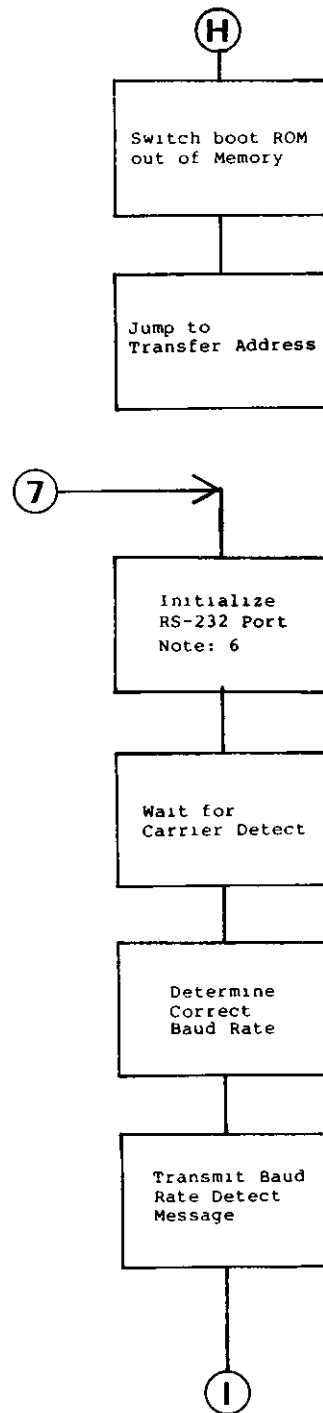
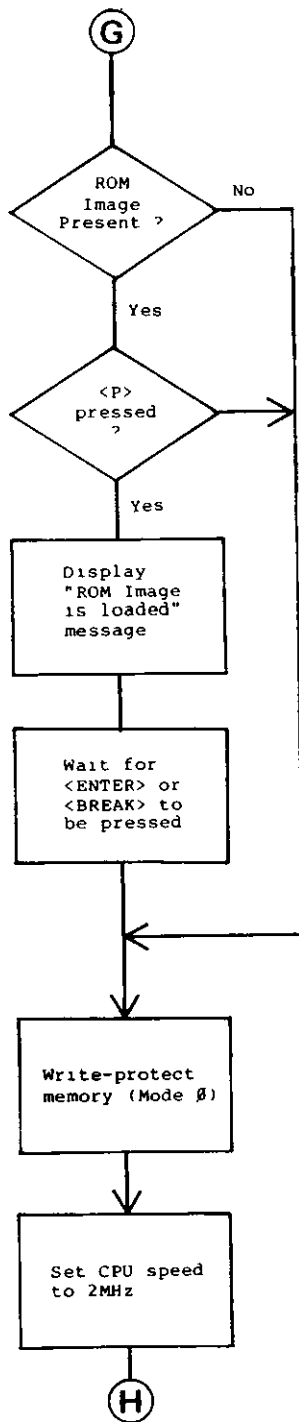
The Miscellaneous keys are

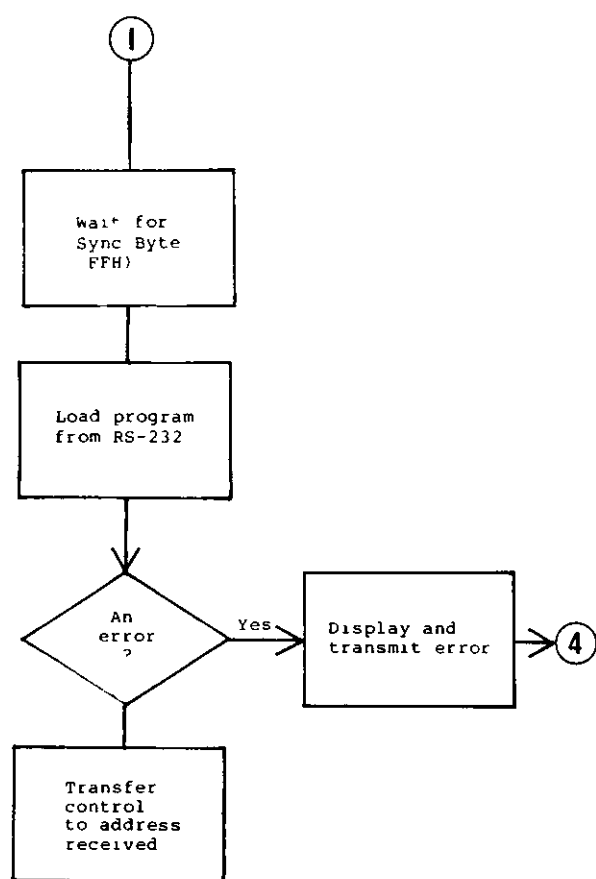
<Break>	Pressing this key is simply recorded by setting location 405BH non-zero. It is up to an operating system to use this flag if desired.
Enter	Terminates the Keyboard routine. Any other keys pressed up to that time will be acted upon. Enter is useful for experienced users who do not want to wait until the keyboard timer expires.

The Control section now takes over and follows the following flowchart









Notes

- (1) If the boot sector was not 256 bytes in length then it is assumed to be a Model III package and the ROM image will be needed. If the sector is 256 bytes in length then the sector is scanned for the sequence CDxx00H. The CD is the first byte of a Z80 unconditional subroutine call. The next byte can have any value. The third byte is tested against a zero. What this check does is test for any references to the first 256 bytes of memory. All Radio Shack Model III operating systems and many other packages all reference the ROM at some point during the boot sector. Most boot sectors will display a message if the system can not be loaded. To save space these routines use the Model III ROM calls to display the message. Several ROM calls have their entry points in the first 256 bytes of memory and these references are detected by the boot ROM.

Packages that do not reference the Model III ROM in the boot sector can still cause the Model III ROM image to be loaded by coding a CDxx00 somewhere in the boot sector. It does not have to be executable. At the same time Model 4 packages must take care that there is no sequence of bytes in the boot sector that could be misinterpreted to be a reference to the Boot ROM. An example of this would be sequence 06CD0E00 which is a LD B 0CDH and a LD C 0. If the boot sector cannot be changed then the user must press the F3 key each time the system is started to inform the ROM that the disk contains a Model III package which needs the Model III ROM image.

- (2) If you are loading a Model 4 operating system then the boot ROM will always transfer control to the first byte of the boot sector which is at 4300H. If you are loading a Model III operating system or about to use Model III ROM BASIC then the transfer address is 3015H. This is the address of a jump vector in the C ROM of the Model III ROM image and this will cause the system to behave exactly like a Model III. If the ROM image file that is loaded has a different transfer address then that address will be used when loading is complete. If the image is already present the Boot ROM will use 3015H.
- (3) Two different tests are done to insure that the Model III ROM image is present. The first test is to check every third location starting at 3000H for a C3H. This is done for 10 locations. If any of these locations does not contain a C3H then the ROM image is considered to be not present. The next test is to check two bytes at location 000BH. If these addresses contain E9E1H then the ROM image is considered to be present.
- (4) See Disk Director Searcher for more information.
- (5) See File Loader for more information.
- (6) The RS 232 loader is described under RS 232 Boot.

Disk Directory Searcher

When the Model III ROM image is to be loaded it is always read from the floppy in drive 0.

Before the operation begins some checks are made. First the boot sector is read in from the floppy and the first byte is checked to make sure it is either a 00H or a FEH. If the byte contains some other value no attempt will be made to read the ROM image from that disk. The location of the directory cylinder is then taken from the boot sector and the type of disk is determined. This is done by examining the Data Address Mark that

was picked up by the Floppy Disk Controller (FDC) during the read of the sector. If the DAM equals 1, the disk is a TRSDOS 1.x style disk. If the DAM equals 0, then the disk is a LDOS 5.1 TRSDOS 6 style disk. This is important since TRSDOS 1.x disks number sectors starting with 1 and LDOS style disks number sectors starting with 0.

Once the disk type has been determined, an extra test is made if the disk is a LDOS style disk. This test reads the Granule Address Location Table (GAT) to determine if the disk is single sided or double sided.

The directory is then read one record at a time and a compare is made against the pattern 'MODEL%' for the filename and 'III' for the extension. The '%' means that any character will match this position. If the user pressed one of the selection keys (A-G) during the keyboard scan, then that character is substituted in place of the '%' character. For example, if you pressed 'D', then the search would be for the file MODEL.D, with the extension 'III'. The searching algorithm searches until it finds the entry or it reaches the end of the directory.

Once the entry has been found, the extent information for that file is copied into a control block for later use.

File Loader

The file loader is actually two modules — the actual loader and a set of routines to fetch bytes from the file on disk. The loader is invoked via a RST 28H. The byte fetcher is called by the loader using RST 20H. Since restart vectors can be re-directed, the same loader is used by the RS-232 boot. The difference is that the RST 20H is redirected to point to the RS-232 data receiving routine. The loader reads standard loader records and acts upon two types:

- 01 Data Load
 - 1 byte with length of block, including address
 - 1 word with address to load the data
 - n bytes of data, where $n + 2$ equals the length specified
- 02 Transfer Address
 - 1 byte with the value of 02
 - 1 word with the address to start execution at

Any other loader code is treated as a comment block and is ignored. Once an 02 record has been found, the loader stops reading, even if there is additional data, so be sure to place the 02 record at the end of the file.

Floppy and Hard Disk Driver

The disk drivers are entered via RST 8H and will read a sector anywhere on a floppy disk and anywhere on head 1 (top-head) in a hard disk drive. Either 256 or 512 byte sectors are readable by these routines and they make the determination of the sector size. The hard disk driver is compatible with both the WD1000 and the WD1010 controllers. The floppy disk driver is written for the WD1793 controller.

Serial Loader

Invoking the serial loader is similar to forcing a boot from hard disk or floppy. In this case the right shift key must be pressed at some time during the first three seconds after reset. The program does not care if the key is pressed forever, making it convenient to connect pins 8 and 10 of the keyboard connector with a shorting plug for bench testing of boards. This assumes that the object program being loaded does not care about the key closure.

Upon entry, the program first asserts DTR (J4 pin 20) and RTS (J4 pin 4) true. Next, Not Ready is printed on the topmost line of the video display. Modem status line CD (J4 pin 8) is then sampled. The program loops until it finds CD asserted true. At that time the message "Ready" is displayed. Then the program sets about determining the baud rate from the host computer.

To determine the baud rate, the program compares data received by the UART to a test byte equal to 55 hex. The receiver is first set to 19200 baud. If ten bytes are received which are not equal to the test byte, the baud rate is reduced. This sequence is repeated until a valid test byte is received. If ten failures occur at 50 baud, the entire process begins again at 19200 baud. If a valid test byte is received, the program waits for ten more to arrive before concluding that it has determined the correct baud rate. If at this time an improper byte is received or a receiver error (overrun, framing, or parity) is intercepted, the task begins again at 19200 baud.

In order to get to this point, the host or the modem must assert CD true. The host must transmit a sequence of test bytes equal to 55 hex with 8 data bits, odd parity, and 1 or 2 stop bits. The test bytes should be separated by approximately 0.1 second to avoid overrun errors.

When the program has determined the baud rate, the message

"Found Baud Rate x"

is displayed on the screen, where 'x' is a letter from A to P meaning

A = 50 baud	E = 150	I = 1800	M = 4800
B = 75	F = 300	J = 2000	N = 7200
C = 110	G = 600	K = 2400	O = 9600
D = 134.5	H = 1200	L = 3600	P = 19200

The same message less the character signifying the baud rate is transmitted to the host, with the same baud rate and protocol. This message is the signal to the host to stop transmitting test bytes.

After the program has transmitted the baud rate message, it reads from the UART data register in order to clear any overrun error that may have occurred due to the test bytes coming in during the transmission of the message. This is because the receiver must be made ready to receive a sync byte signalling the beginning of the command file. For this reason, it is important that the host wait until the entire baud rate message (16 characters) is received before transmitting the sync byte, which is equal to FF hex.

When the loader receives the sync byte, the message

"Loading"

is displayed on the screen. Again, the same message is transmitted to the host, and, again, the host must wait for the entire transmission before starting into the command file.

If the receiver should intercept a receive error while waiting for the sync byte, the entire operation up to this point is aborted. The video display is cleared and the message

"Error, x"

is displayed near the bottom of the screen, where x is a letter from B to H, meaning

- B = parity error
- C = framing error
- D = parity & framing errors
- E = overrun error
- F = parity & overrun errors
- G = framing & overrun errors
- H = parity & framing & overrun errors

The message

"Error"

is then transmitted to the host. The entire process is then repeated from the 'Not Ready' message. A six second delay is inserted before reinitialization. This is longer than the time required to transmit five bytes at 50 baud, so there is no need to be extra careful here.

If the sync byte is received without error, then the "Loading" message is transmitted and the program is ready to receive the command file. After receiving the 'Loading' message, the host can transmit the file without nulls or delays between bytes.

(Since the file represents Z80 machine code and all 256 combinations are meaningful, it would be disastrous to transmit nulls or other ASCII control codes as fillers, acknowledgements, or start-stop bytes. The only control codes needed are the standard command file control bytes.)

Data can be transmitted to the loader at 19200 baud with no delays inserted. Two stop bits are recommended at high baud rates.

See the File Loader description for more information on file loading.

If a receive error should occur during file loading, the abort procedure described above will take place, so when attempting remote control, it is wise to monitor the host receiver during transmission of the file. When the host is near the object board as is the case in the factory application, or when more than one board is being loaded, it may be advantageous or even necessary to ignore the transmitted responses of the object board(s) and to manually pace the test byte, sync byte, and command file phases of the transmission process, using the video display for handshaking.

System Programmers Information

The Model 4P Boot ROM uses two areas of RAM while it is running. These are 4000H to 40FFH and 4300H to 43FFH. (For 512 byte boot sectors, the second area is 4300H to 44FFH.) If the Model III ROM Image is loaded, additional areas are used. See the technical reference manual for the system you are using for a list of these areas.

Operating systems that want to support a software restart by re-executing the contents of the boot ROM can accomplish this in one of two ways. If the operating system relies on the Model III ROM Image, then jump to location 0 as you have in the past. If the operating system is a Model 4 mode package, a simple way is to code the following instructions in your assembly and load them before you want to reset.

Absolute Location	Instruction	
0000	DI	
0001	LD	A 1
0003	OUT	(9CH) A

These instructions cause the boot ROM to become addressable. After executing the OUT instruction, the next instruction executed will be one in the boot ROM. (These instructions also exist in the Model III ROM image at location 0.) The boot ROM has been written so that the first instruction is at address 0005. The hardware must be in memory mode 0 or 1, or else the boot ROM will not be switched in. This operation can be done with an OUT instruction and then a RST 0 can be executed to have the ROM switched in.

Restarts can be redirected at any time while the ROM is switched in. All restarts jump to fixed locations in RAM and these areas may be changed to point to the routine that is to be executed.

Restart	RAM Location	Default Use
0	none	Cold Start Boot
8	4000H	Disk I/O Request
10	4003H	Display string
18	4006H	Display block
20	4009H	Byte Fetch (Called by Loader)
28	400CH	File Loader
30	400FH	Keyboard scanner
38	4012H	Reserved for future use
66	4015H	NMI (Floppy I/O Command Complete)

The above routines have fixed entry parameters. These are described here.

Disk I/O Request (RST 8H)

Accepts

A	1 for floppy, 2 for hard disk
B	Command
	Initialize 1
	Restore 4
	Seek 6
	Read 12 (All reads have an implied seek)
C	Sector number to read
	The contents of the location disktype (405CH) are added to this value before an actual read. If the disk is a two-sided floppy, just add 18 to the sector number.
DE	Cylinder number (Only E is used in floppy operations)
HL	Address where data from a read operation is to be stored

Returns

Z	Success: Operation Completed
NZ	Error: Error code in A

Error Codes

3	Hard Disk drive is not ready
4	Floppy disk drive is not ready
5	Hard Disk drive is not available
6	Floppy disk drive is not available
7	Drive Not Ready and no Index (Disk in drive door open)
8	CRC Error
9	Seek Error
11	Lost Data
12	ID Not Found

Display String (RST 10H)

Accepts

HL	Pointer to text to be displayed
	Text must be terminated with a null (0)
DE	Offset position on screen where text is to be displayed
	(A 0000H will be the upper left-hand corner of the display)

Returns

Success Always

A	Altered
DE	Points to next position on video
HL	Points to the null (0)

Display Block (RST 18H)

Accepts

HL	Points to control vector in the format
+ 0	Screen Offset
+ 2	Pointer to text, terminated with null
+ 4	Pointer to text terminated with null
+ n	word FFFFH End of control vector
or	+ n word FFFEH Next word is new Screen Offset

If Z flag is set on entry, then the first screen offset is read from DE instead of from the control vector.

Each string is positioned after the previous string unless a FFFEH entry is found. This is used heavily in the ROM to reduce duplication of words in error messages.

Returns

Success Always

DE	Points to next position on video
-----------	----------------------------------

Byte Fetch (RST 20H)

Accepts None

Returns

Z	Success: byte in A
NZ	Failure: error code in A

Errors

2	Any errors from the disk I/O call and ROM image can't be loaded — Too many extents
10	ROM image can't be loaded — Disk drive is not ready

File Loader (RST 28H)

Accepts None

Returns

Z	Success
NZ	Failure, error code in A

Errors

	Any errors from the disk I/O call or the byte fetch call and:
0	The ROM image was not found on drive 0

There are several pieces of information left in memory by the boot ROM which are useful to system programmers. These are shown below:

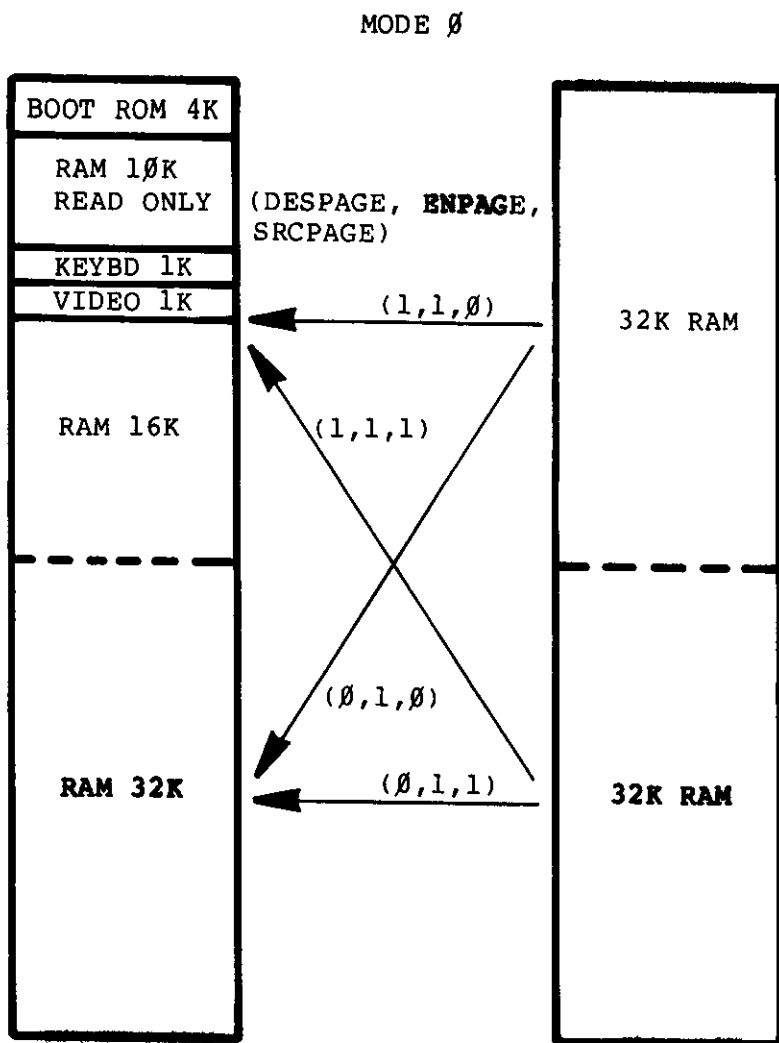
RAM Location	Description
401DH	ROM Image Selected (% for none selected or A-G)
4055H	Boot type 1 = Floppy 2 = Hard disk 3 = ARCNET 4 = RS-232C 5 - 7 = Reserved
4056H	Boot Sector Size (1 for 256, 2 for 512)
4057H	RS-232 Baud Rate (only valid on RS-232 boot)
4059H	Function Key Selected 0 = No function key selected <F1> or <1> 86 <F2> or <2> 87 <F3> or <3> 88 <Caps> 85 <Ctrl> 84 <Left-Shift> 82 <Right-Shift> 83 Reserved 80-81 and 89-90
405BH	Break Key Indication (non-zero if <Break> pressed)
405CH	Disk type (0 for LDOS TRSDOS 6.1 for TRSDOS 1.x)

Keep in mind that Model III ROM image will initialize these areas, so this information is useful only to the Model 4 mode programmer.

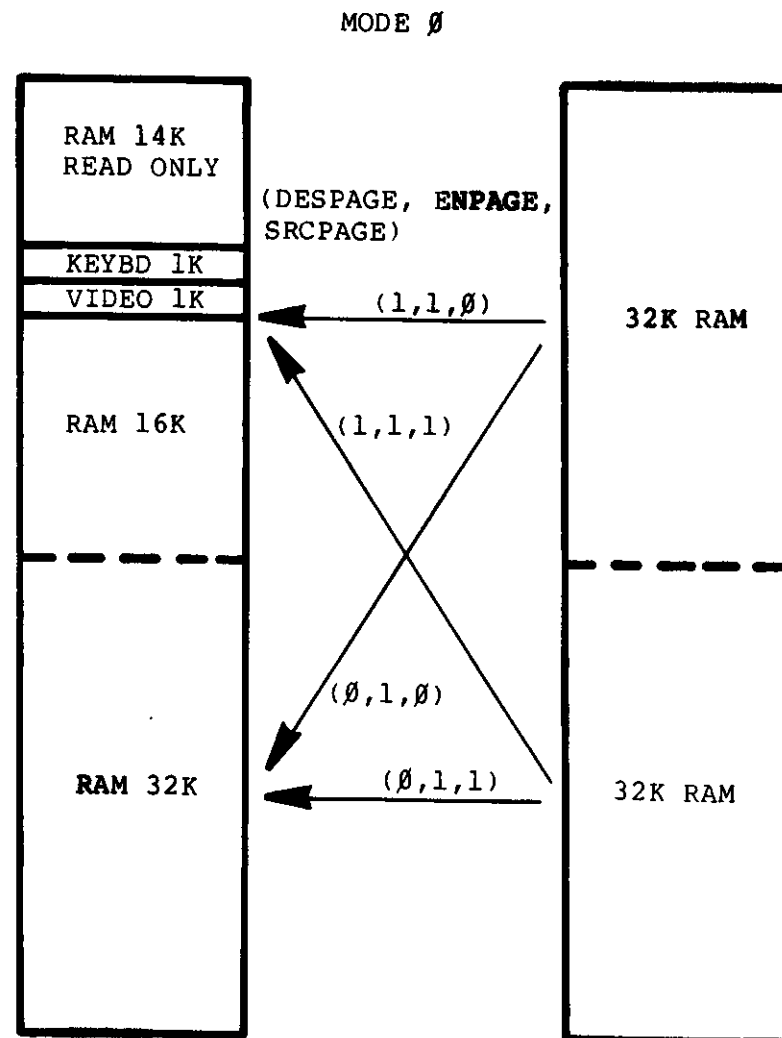
3.1.7 RAM

Two configurations of Random Access Memory (RAM) are available on the Model 4P: 64K and 128K. The 64K and 128K option use the 6665-type 64K x 1 200NS Dynamic RAM, which requires only a single - 5v supply voltage.

The DRAMs require multiplexed incoming address lines. This is accomplished by ICs U111 and U112 which are 74LS157 multiplexers. Data to and from the DRAMs are buffered by a 74LS245 (U117) which is controlled by Page Map PAL, U110. The proper timing signals RAS0*, RAS1*, MUX*, and CAS* are generated by a delay line circuit U97. U115 (1 2 of a 74S112) and U116 (1 4 of a 74F08) are used to generate a precharge circuit. During M1 cycles of the Z80A in 4 MHz mode, the high time in MREQ has a minimum time of 110 nanosecs. The specification of 6665 DRAM requires a minimum of 120 nanosecs so this circuit will shorten the MREQ signal during the M1 cycle. The resulting signal PMREQ is used to start a RAM memory cycle through U113 (a 74S64). Each different cycle is controlled at U113 to maintain a fast M1 cycle so no wait states are required. The output of U113 (PRAS*) is ANDed with RFSH to not allow MUX* and CAS* to be generated during a REFRESH cycle. PRAS* also generates either RAS0* or RAS1*, depending on which bank of RAM the CPU is selecting. GCAS* generated by the delay line U97 is latched by U115 (1 2 of a 74S112) and held to the end of the memory cycle. The output of U115 is ANDed with VIDEO signal to disable the CAS* signal from occurring if the cycle is a video memory access. Refer to M1 Cycle Timing (Figure 3-8. and 3-9.), Memory Read and Memory Write Cycle Timing (Figure 3-10.) and (Figure 3-11.).



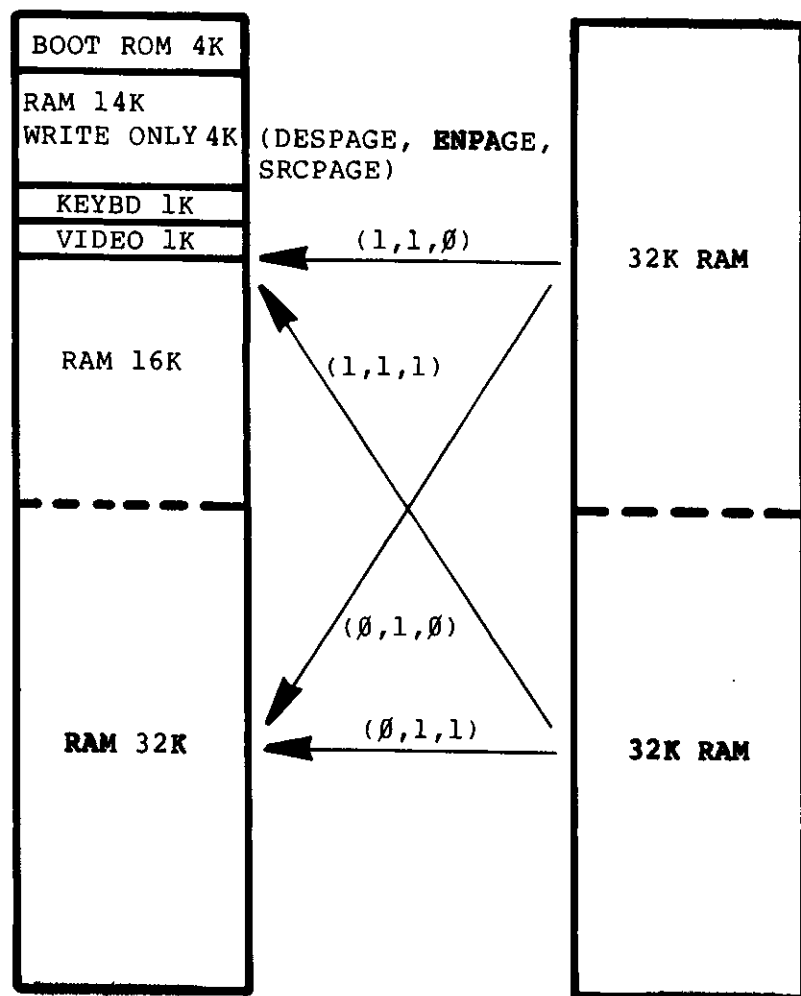
	STATE	LEVEL
SEL0 =	0	0V
SEL1 =	0	0V
ROM =	1	0V



	STATE	LEVEL
SEL0 =	0	0V
SEL1 =	0	0V
ROM =	0	5V

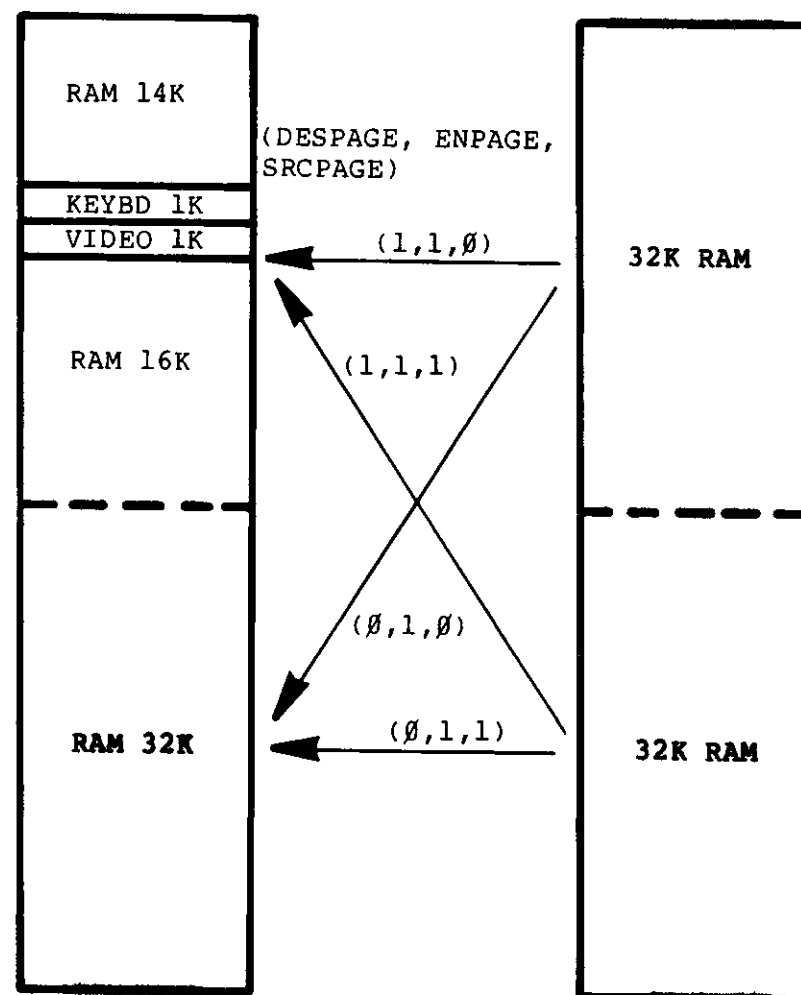
Figure 3-5. Memory

MODE 1



	STATE	LEVEL
SEL0 =	1	5V
SEL1 =	0	0V
ROM =	0	5V

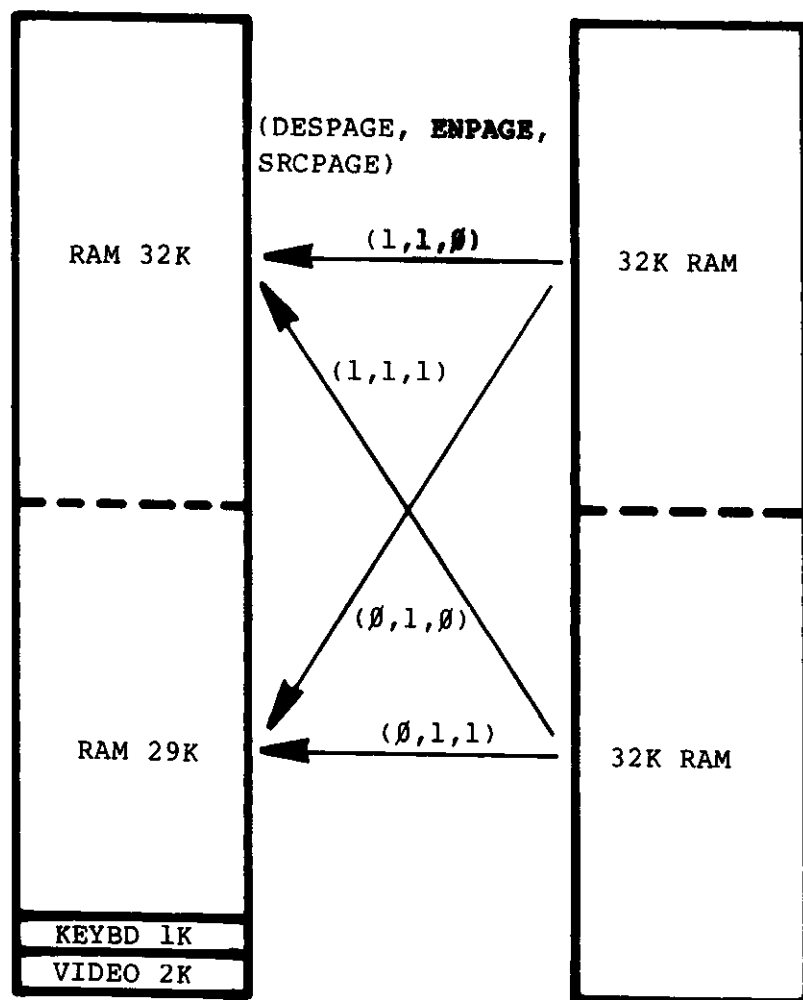
MODE 1



	STATE	LEVEL
SEL0 =	1	5V
SEL1 =	0	0V
ROM =	1	0V

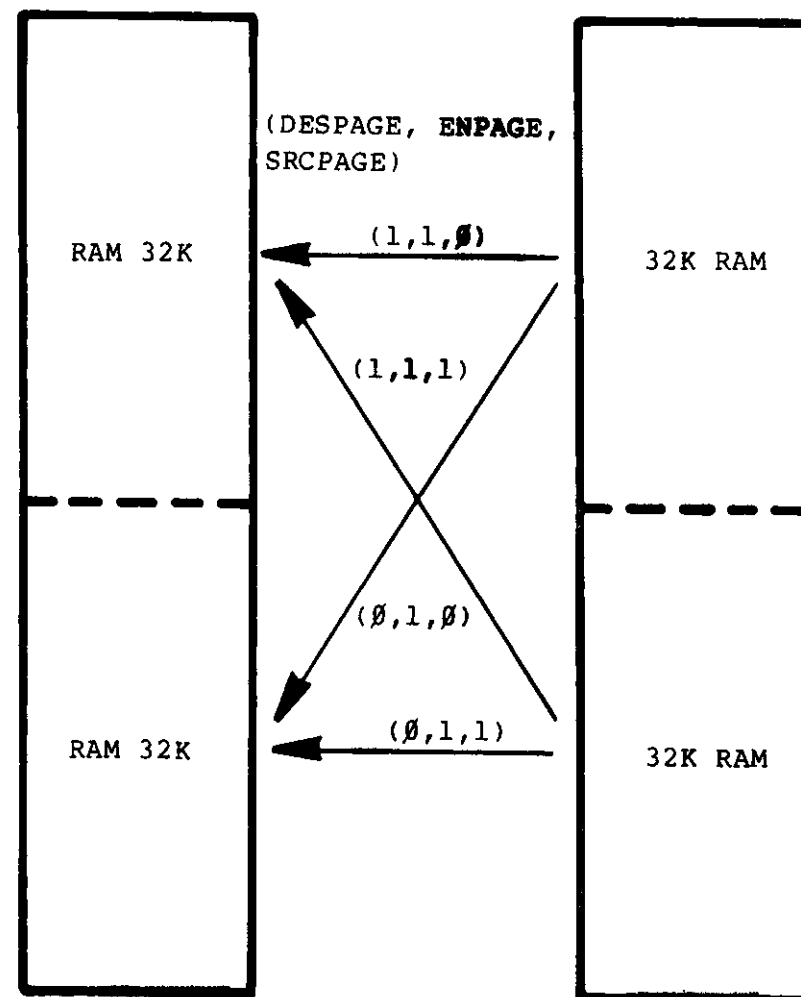
Figure 3-6. Memory

MODE 2



	STATE	LEVEL
SEL0 =	0	0V
SEL1 =	1	5V
ROM =	X	

MODE 3



	STATE	LEVEL
SEL0 =	1	5V
SEL1 =	1	5V
ROM =	X	

Figure 3-7. Memory

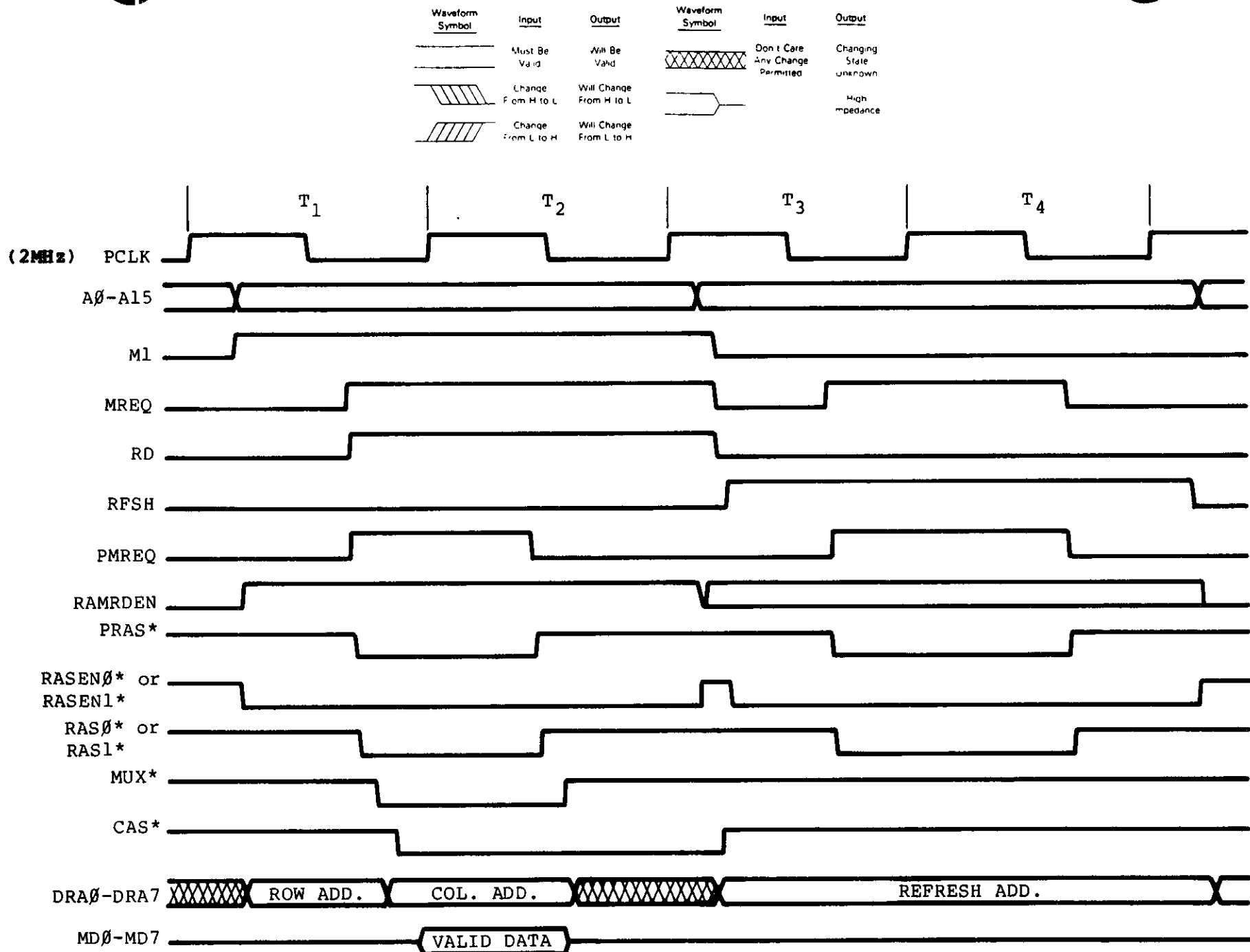


Figure 3-8. M1 Cycle Timing (2MHZ) 100ns/div.

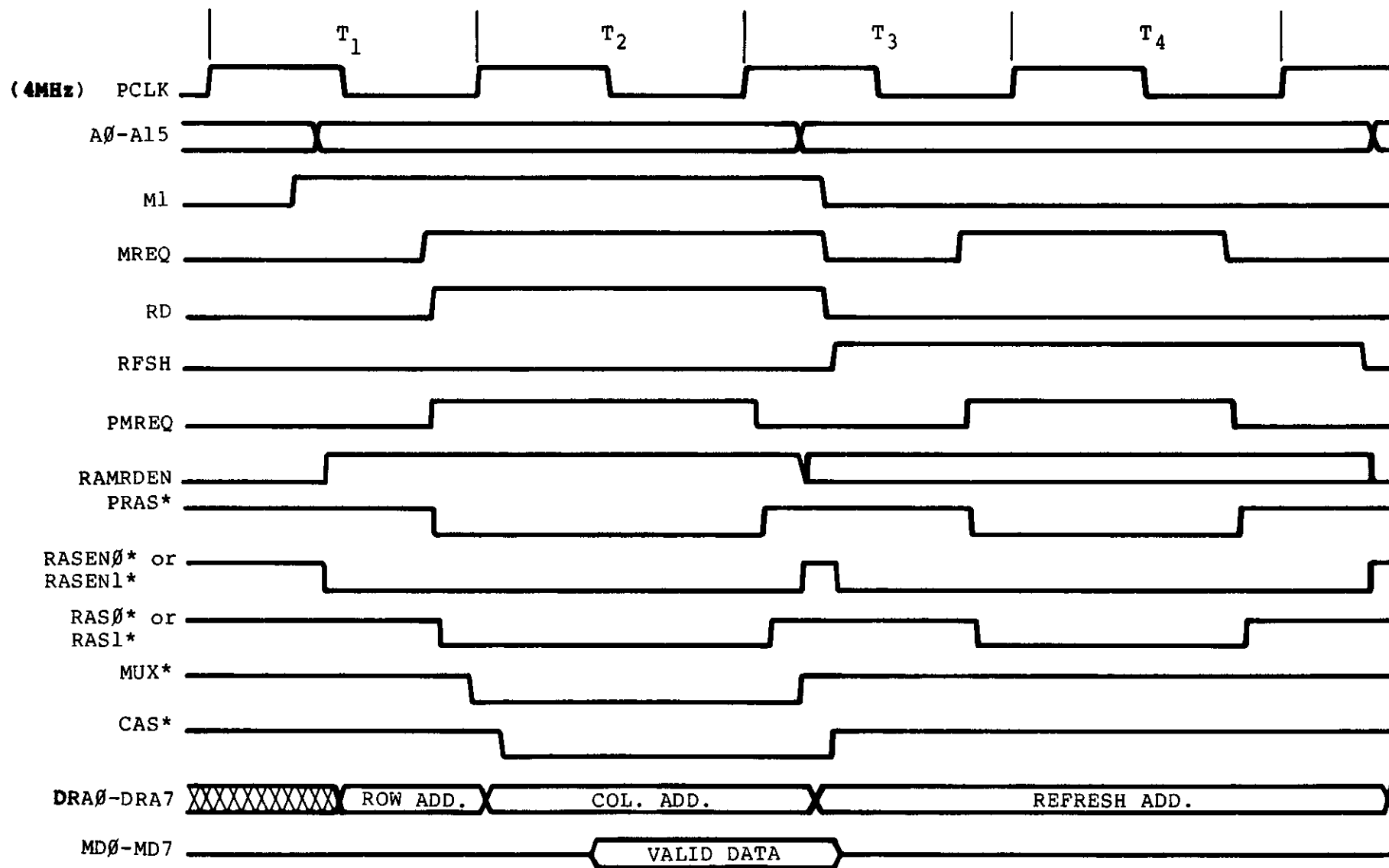


Figure 3-9. M1 Cycle Timing (4MHZ) 50ns/dir.

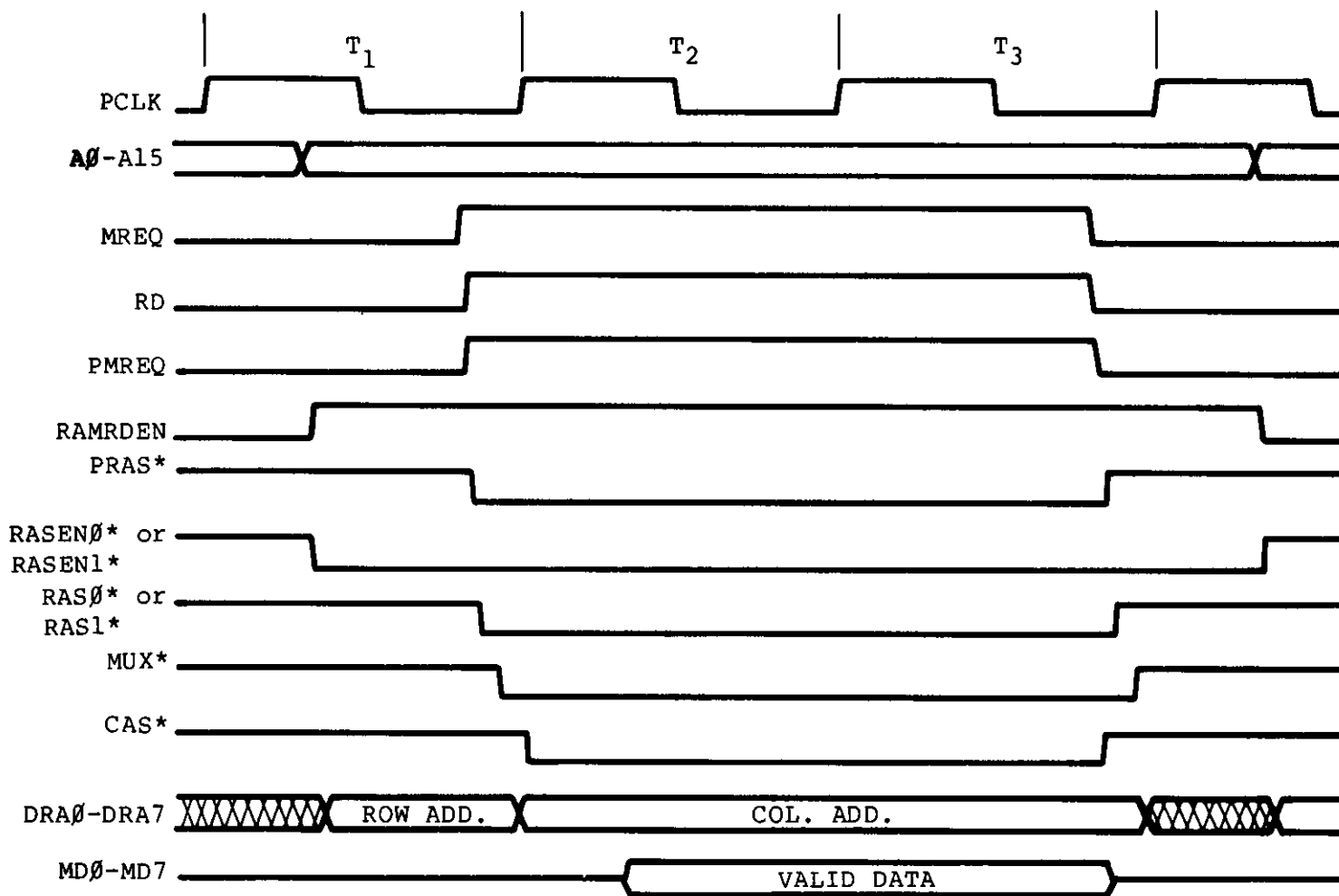


Figure 3-10. Memory Read Cycle Timing

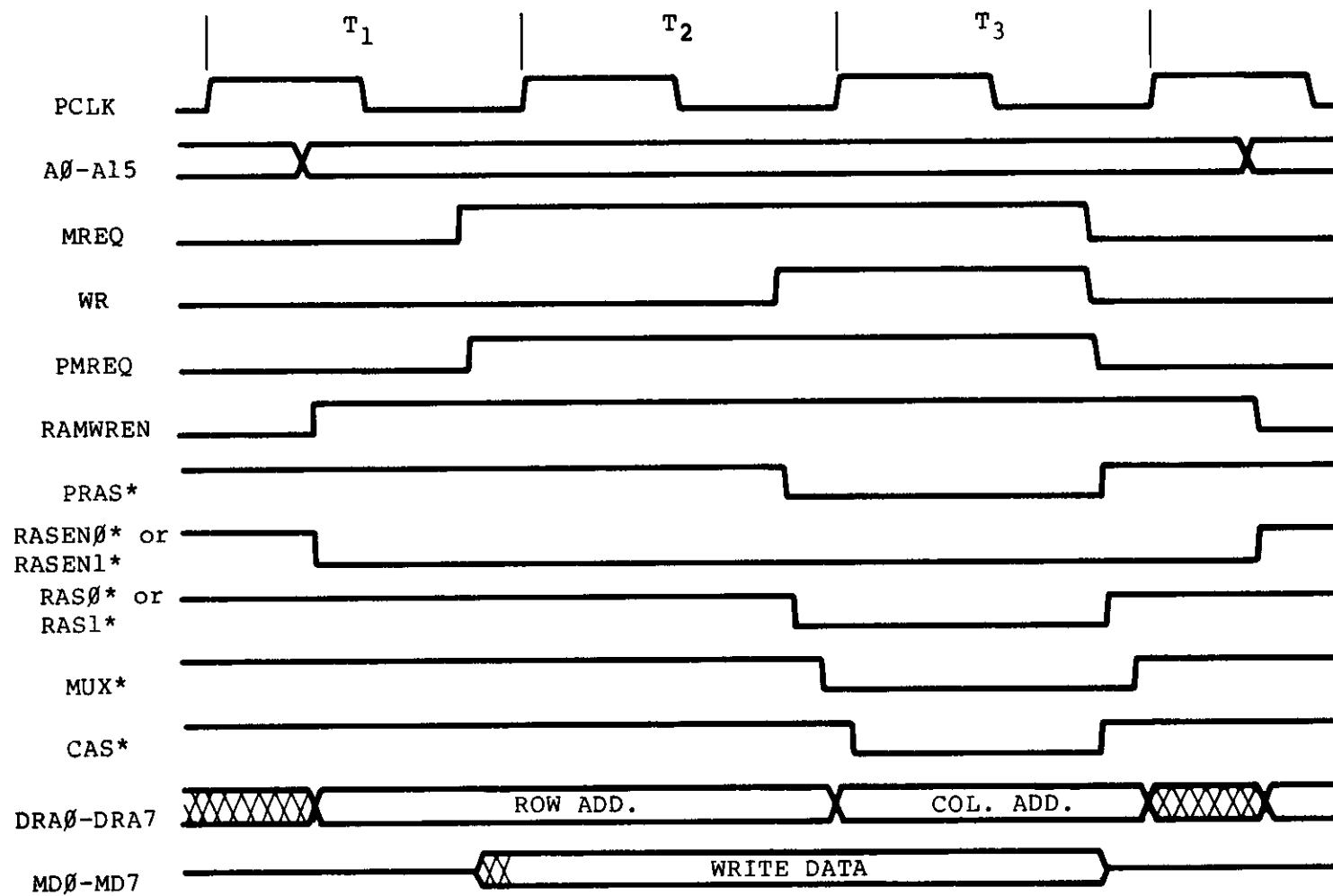


Figure 3-11. Memory Write Cycle Timing

Memory Map — Model 4P

Mode 0
SEL0 0 - 0V
SEL1 0 0V
ROM 1 0V

0000 — 0FFF	Boot ROM	4K
1000 — 37FF	RAM (Read Only)	10K
37E8 — 37E9	Printer Status (Read Only)	2
3800 — 3BFF	Keyboard	1K
3C00 — 3FFF	Video	1K
4000 — FFFF	RAM	48K

Mode 0
SEL0 - 0 - 0V
SEL1 - 0 - 0V
ROM - 0 - +5V

0000 — 37FF	RAM (Read Only)	14K
37E8 — 37E9	Printer Status (Read Only)	2
3800 — 3BFF	Keyboard	1K
3C00 — 3FFF	Video	1K
4000 — FFFF	RAM	48K

Mode 1
SEL0 - 1 - +5V
SEL1 - 0 - 0V
ROM - 1 - 0V

0000 — 0FFF	Boot ROM	4K
0000 — 0FFF	RAM (Write Only)	4K
1000 — 37FF	RAM	10K
3800 — 3BFF	Keyboard	1K
3C00 — 3FFF	Video	1K
4000 — FFFF	RAM	48K

Mode 1
SEL0 - 1 - +5V
SEL1 - 0 - 0V
ROM - 0 - +5V

0000 — 37FF	RAM	14K
3800 — 3BFF	Keyboard	1K
3C00 — 3FFF	Video	1K
4000 — FFFF	RAM	48K

Mode 2
SEL0 - 0 - 0V
SEL1 - 1 - +5V
ROM = X - Don't Care

0000 — F3FF	RAM	61K
F400 — F7FF	Keyboard	1K
F800 — FFFF	Video	2K

Mode 3
SEL0 = 1 - +5V
SEL1 = 1 - +5V
ROM - X - Don't Care

0000 — FFFF	RAM	64K
-------------	-----	-----

I/O Port Assignment

Normally

Port #	Used	Out	In
FC — FF	FF	CASSOUT *	MODIN *
F8 — FB	F8	LPOUT *	LPIN *
F4 — F7	F4	DRVSEL *	(RESERVED)
F0 — F3	-	DISKOUT *	DISKIN *
F0	F0	FDC COMMAND REG.	FDC STATUS REG.
F1	F1	FDC TRACK REG.	FDC TRACK REG.
F2	F2	FDC SECTOR REG.	FDC SECTOR REG.
F3	F3	FDC DATA REG.	FDC DATA REG.
EC — EF	EC	MODOUT *	RTCIN *
E8 — EB	-	RS232OUT *	RS232IN *
E8	E8	UART MASTER RESET	MODEM STATUS
E9	E9	BAUD RATE GEN. REG.	(RESERVED)
EA	EA	UART CONTROL AND MODEM CONTROL REG.	UART STATUS REG.
EB	EB	UART TRANSMIT HOLDING REG.	UART HOLDING REG. (RESET D.R.)
E4 — E7	E4	WR NMI MASK REG. *	RD NMI STATUS *
E0 — E3	E0	WR INT MASK REG. *	RD INT MASK REG. *
A0 — DF	-	(RESERVED)	(RESERVED)
9C — 9F	9C	BOOT *	(RESERVED)
94 — 9B	-	(RESERVED)	(RESERVED)
90 — 93	90	SEN *	(RESERVED)
8C — 8F	-	GSEL0 *	GSEL0 *
88 — 8B	-	CRTCCS *	(RESERVED)
88, 8A	88	CRCT ADD. REG.	(RESERVED)
89, 8B	89	CRCT DATA REG.	(RESERVED)
84 — 87	84	OPREG *	(RESERVED)
80 — 83	-	GSEL1 *	GSEL1 *

I/O Port Description

Name: CASSOUT *
Port Address: FC — FF
Access: WRITE ONLY
Description: Output data to cassette or for sound generation

Note: The Model 4P **does not** support cassette storage. this port is only used to generate sound that was to be output via cassette port. The Model 4P sends data to onboard sound circuit.

D0 = Cassette output level (sound data output)

D1 = Reserved

D2 — D7 = Undefined

Name: MODIN * (CASSIN *)
Port Address: FC — FF
Access: READ ONLY
Description: Configuration Status

D0 = 0

D1 = CASSMOTORON STATUS

D2 = MODSEL STATUS

D3 = ENALTSET STATUS

D4 = ENEXTIO STATUS

D5 = (NOT USED)

D6 = FAST STATUS

D7 = 0

Name: LPOUT *
Port Address: F8 — FB
Access: WRITE ONLY
Description: Output data to line printer

D0 — D7 = ASCII BYTE TO BE PRINTED

Name: LPIN *
Port Address: F8 — FB
Access: READ ONLY
Description: Input line printer status

D0 — D3 = (RESERVED)

D4 = FAULT
1 = TRUE
0 = FALSE

D5 = UNIT SELECT
1 = TRUE
0 = FALSE

D6 = OUTPAPER
1 = TRUE
0 = FALSE

D7 = BUSY
1 = TRUE
0 = FALSE

Name: DRVSEL *
Port Address: F4 — F7
Access: WRITE ONLY
Description: Output FDC Configuration

Note: Output to this port will **ALWAYS** cause a 1-2 msc. (Microsecond) wait to the Z80.

D0 = DRIVE SELECT 0

D1 = DRIVE SELECT 1

D2 = (RESERVED)

D3 = (RESERVED)

D4 = SDSEL
0 = SIDE 0
1 = SIDE 1

D5 = PRECOMPEN
0 = No write precompensation
1 = Write Precompensation enabled

D6 = WSGEN
0 = No wait state generated
1 = wait state generated

Note: This wait state is to sync Z80 with FDC chip during FDC operation.

D7 = DDEN *
0 = Single Density enabled (FM)
1 = Double Density enabled (MFM)

Name: DISKOUT *
Port Address: F0 — F3
Access: WRITE ONLY
Description: Output to FDC Control Registers

Port F0 = FDC Command Register

Port F1 = FDC Track Register

Port F2 = FDC Sector Register

Port F3 = FDC Data Register

(Refer to FDC Manual for Bit Assignments)

Name: DISKIN *
Port Address: F0 — F3
Access: READ ONLY
Description: Input FDC Control Registers

Port F0 = FDC Status Register

Port F1 = FDC Track Register

Port F2 = FDC Sector Register

Port F3 = FDC Data Register

(Refer to FDC Manual for Bit Assignment)

Name: MODOUT *
Port Address: EC — EF
Access: WRITE ONLY
Description: Output to Configuration Latch

D0 = (RESERVED)

D1 = CASSMOTORON (Sound enable)
 0 = Cassette Motor Off (Sound enabled)
 1 = Cassette Motor On (Sound disabled)

D2 = MODSEL
 0 = 64 or 80 character mode
 1 = 32 or 40 character mode

D3 = ENALTSET
 0 = Alternate character set disabled
 1 = Alternate character set enabled

D4 = ENEXTIO
 0 = External IO Bus disabled
 1 = External IO Bus enabled

D5 = (RESERVED)

D6 = FAST
 0 = 2 MHZ Mode
 1 = 4 MHZ Mode

D7 = (RESERVED)

Name: RTCIN *
Port Address: EC — EF
Access: READ ONLY
Description: Clear Real Time Clock Interrupt

D0 — D7 = DON'T CARE

Name: RS232OUT *
Port Address: E8 — EB
Access: WRITE ONLY
Description: UART Control Data Control Modem Control, BRG Control

Port E8 = UART Master Reset

Port E9 = BAUD Rate Gen Register

Port EA = UART Control Register (Modem Control Reg)

Port EB = UART Transmit Holding Reg

(Refer to Model III or 4 Manual for Bit Assignments)

Name: RS232IN *
Port Address: E8 — EB
Access: READ ONLY
Description: Input UART and Modem Status

Port E8 = MODEM STATUS

Port E9 = (RESERVED)

Port EA = UART Status Register

Port EB = UART Receive Holding Register (Resets DR)

(Refer to Model III or 4 Manual for Bit Assignments)

Name: WRNMIMASKREG *
Port Address: E4 — E7
Access: WRITE ONLY
Description: Output NMI Latch

D0 — D5 = (RESERVED)

D6 = ENMOTOROFFINT
 0 = Disables Motoroff NMI
 1 = Enables Motoroff NMI

D7 = ENINTRQ
 0 = Disables INTRQ NMI
 1 = Enables INTRQ NMI

Name: RDNMISTATUS *
Port Address: E4 — E7
Access: READ ONLY
Description: Input NMI Status

D0 = 0

D2 — D4 = (RESERVED)

D5 = RESET (not needed)
 0 = Reset Asserted (Problem)
 1 = Reset Negated

D6 = MOTOROFF
 0 = Motoroff Asserted
 1 = Motoroff Negated

D7 = INTRQ
 0 = INTRQ Asserted
 1 = INTRQ Negated

Name: WRINTMASKREG *
Port Address: E0 — E3
Access: WRITE ONLY
Description: Output INT Latch

D0 — D1 = (RESERVED)

D2 = ENRTC
 0 = Real time clock interrupt disabled
 1 = Real time clock interrupt enabled

D3 = ENIOBUSINT
 0 = External IO Bus interrupt disabled
 1 = External IO Bus interrupt enabled

D4 = ENXMITINT
 0 = RS232 Xmit Holding Reg. empty int. disabled
 1 = RS232 Xmit Holding Reg. empty int. enabled

D5 = ENRECINT
 0 = RS232 Rec. Data Reg. full int. disabled
 1 = RS232 Rec. Data Reg. full int. enabled

D6 = ENERRORINT
 0 = RS232 UART Error interrupts disabled
 1 = RS232 UART Error interrupts enabled

D7 = (RESERVED)

Name: RDINTSTATUS *
Port Address: E0 — E3
Access: READ ONLY
Description: Input INT Status

D0 — D1 = (RESERVED)

D2 = RTC INT

D3 = IOBUS INT

D4 = RS232 XMIT INT

D5 = RS232 REC INT

D6 = RS232 UART ERROR INT

D7 = (RESERVED)

Name: BOOT *
Port Address: 9C — 9F
Access: WRITE ONLY
Description: Enable or Disable Boot ROM

D0 = ROM *
 0 = Boot ROM Disabled
 1 = Boot ROM Enabled

D1 — D7 = (RESERVED)

Name: SEN *
Port Address: 90 — 93
Access: WRITE ONLY
Description: Sound output

D0 = SOUND DATA

D1 — D7 = (RESERVED)

Name: OPREG *
Port Address: 84
Access: WRITE ONLY
Description: Output to operation reg.

D0 = SEL0

D1 = SEL1

SEL1	SEL0	MODE
0	0	0
0	1	1
1	0	2
1	1	3

D2 = 8064
0 = 64 character mode
1 = 80 character mode

D3 = INVERSE
0 = Inverse video disabled
1 = Inverse video enabled

D4 = SRCPAGE — Points to the page to be mapped
as new page
0 = U64K, L32K Page
1 = U64K, U32K Page

D5 = ENPAGE — Enables mapping of new page
0 = Page mapping disabled
1 = Page mapping enabled

D6 = DESPAGE — Points to the page where new
page is to be mapped
0 = L64K, U32K Page
1 = L64K, L32K Page

D7 = PAGE
0 = Page 0 of Video Memory
1 = Page 1 of Video Memory

3.1.8 Video Circuit

The heart of the video display circuit in the Model 4P is the 68045 Cathode Ray Tube Controller (CRTC), U85. The CRTC is a preprogrammed video controller that provides two screen formats: 64 by 16 and 80 by 24. The format is controlled by pin 3 of the CRTC (8064*). The CRTC generates all of the necessary signals required for the video display. These signals are VSYNC (Vertical Sync), HSYNC (Horizontal Sync) for proper sync of the monitor, DISPEN (Display Enable) which indicates when video data should be output to the monitor, the refresh memory addresses (MA0-MA13) which addresses the video RAM, and the row addresses (RA0-RA4) which indicates which scan line row is being displayed. The CRTC also provides hardware scrolling by writing to the internal Memory Start Address Register by OUTing to Port 88H. The internal cursor control of the 68045 is not used in the Model 4P video circuit.

Since the 80 by 24 screen requires 1,920 screen memory locations, a 2K by 8 static RAM (U82) is used for the video RAM. Addressing to the video RAM (U82) is provided by the 68045 when refreshing the screen and by the CPU when updating of the data is performed. These two sets of address lines are multiplexed by three 74LS157s (U83, U84, and U104). The multiplexers are switched by CRTCLK which allows the CRTC to address the video RAM during the high state of CRTCLK and the CPU access during the low state. A10 from the CPU is controlled by PAGE* which allows two display pages in the 64 by 16 format. When updates to the video RAM are performed by the CPU, the CPU is held in a WAIT state until the CRTC is not addressing the video RAM. This operation allows reads and writes to video RAM without causing hashing on the screen. The circuit that performs this function is a 74LS244 buffer (U103), an 8 bit transparent latch, 74LS373 (U102) and a Delay line circuit shared with Dynamic RAM timing circuit consisting of a 74LS74 (U95), 74LS32 (U94), 74LS04 (U74), 74LS00 (U96), 74LS02 (U75), and Delay Line (U97). During a CPU Read Access to the Video RAM, the address is decoded by the PAL U109 and asserts VIDEO* low. This is inverted by U74 (1/6 of 74LS04) which pulls one input of U96 (1/4 of 74LS00) and in turn asserts VWAIT* low to the CPU. RD is high at this time and is latched into U95 (1/2 of 74LS74) on the rising edge of XADR7*. XADR7* is inverse of CRTCLK which drives the CRTC (68045), and the address multiplexers U83, U84, and U104.

When RD is latched by U95 the Q output goes low releasing WAIT* from the CPU. The same signal also is sent to the Delay Line (U97) through U116 (1/4 of 74F08). The Delay line delays the falling edge 240 ns for VLATCH* which latches the read data from the video RAM at U102. The data is latched so the CRTC can refresh the next address location and prevent any hashing. MRD* decoded by U108 and a memory read is ORed with VIDEO* which enables the data from U102 to the data bus. The CPU then reads the data and completes the cycle. A CPU write is slightly more complex in operation. As in the RD cycle, VIDEO* is asserted low which asserts VWAIT* low to the CPU. WR is high at this time which is Nanded with VIDEO and synced with CRTCLK to create VRAMDIS that disables the video RAM output. On the rising edge of XADR7*, WR is latched into U95 (1/2 of 74LS74) which releases VWAIT* and starts cycle through the Delay Line. After 30ns DLYVWR* (Delayed video write) is asserted low which also asserts VBUFEN* (Video Buffer Enable) low. VBUFEN* enabled data from the Data bus to the video RAM. Approximately 120ns later DLYVWR* is negated high which writes the data to the video RAM and negates VBUFEN* turning off buffer. The CPU then completes WR cycle to the video RAM. Refer to Video RAM CPU Access Timing Figure 5-12 for timing of above RD or WR cycles.

During screen refresh, CRTCLK is high allowing the CRTC to address Video RAM. The data out of the video RAM is latched by LOAD* into a 74LS273 (U101). D7 is generated by INVERSE* through U125 (1/6 of 74S04), and U123 (1/4 of 74LS08). This decoding determines if character should be alpha-numeric only (if inverse high) or unchanged (INVERSE* low). The outputs of U101 are used as address inputs the character generator ROM (U42). A9 is decoded with ENALTSET (Enable Alternate Set) and Q7 of U101, which resets A9 to a low if Q7 and ENALTSET are high. See ENALTSET Control Table below.

ENALTSET	Q7	Q6	A9
0	0	0	0
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	0
1	1	1	0

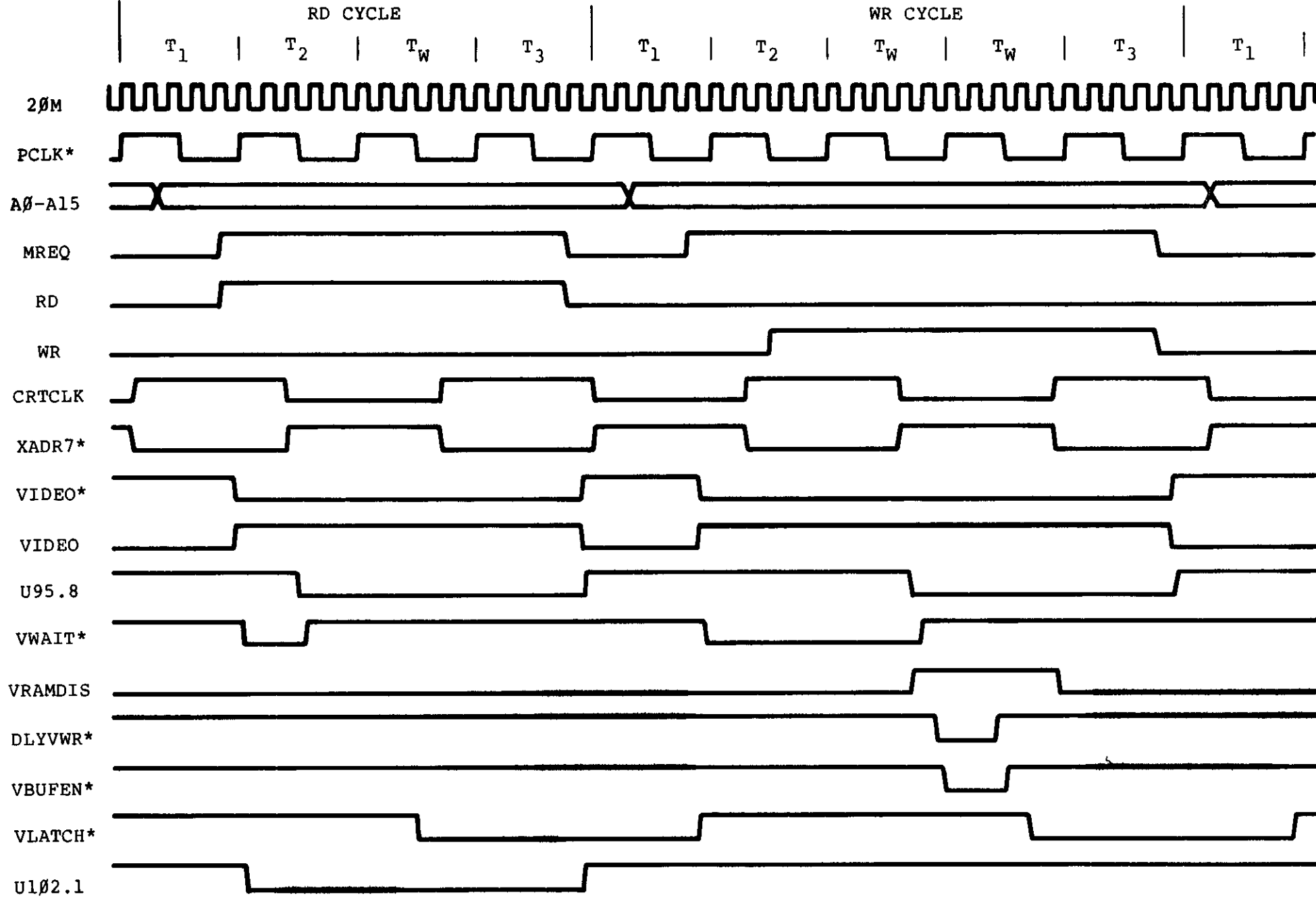


Figure 3-12. Video RAM CPU Access Timing

RA0-RA3 row addresses from the CRTC are used to control which scan line is being displayed. The Model 4P has a 4-bit full adder 74LS283 (U61) to modify the Row address. During a character display DLYGRAPHIC* is high which applies a high to all 4 bits to be added to row address. This will result in subtracting one from Row address count and allow all characters to be displayed one scan line lower. The purpose is so inverse characters will appear within the inverse block. When a graphic block is displayed DLYGRAPHIC* is low which causes the row address to be unmodified. Moving jumper from E14-E15 to E15-E16 will disable this circuit.

DLYCHAR* and DLYGRAPHICS are inverse signals and control which data is to be loaded into the shift register U63. When DLYCHAR* is low and DLYGRAPHIC* is high, the Character Generator ROM (U42) is enabled to output data when DLYCHAR* is high and DLYGRAPHIC* is low the graphics characters from U41 (74LS15) is buffered by U43 (74LS244) to the shift register. The data is loaded into the shift register on the rising edge of SHIFT* when LOADS* is low. Blanking is accomplished by masking off LOADS* so no data will be loaded and zero data will be shifted out with the serial input of U63, pin 1, grounded. Serial video data is output U63 pin 13 and is mixed with inverse and/or hires graphics information by (1/4 or 74LS86) U143. The video data is then mixed with a DO7 Rate clock, either DOT* and DCLK, to create distinct dots on the monitor. DOT* and DCLK are inverse signals and are provided to allow a choice to obtain the best video results. The video information is filtered by F34, R45 (47 ohm resistor), and C241 (100 pf Cap) and output to video monitor. VSYNC and HSYNC are buffered by (1/2 of 74LS86) U143 and are also output to video monitor. Refer to Video Circuit Timing Figure 3-13, Video Blanking Timing Figure 3-14, and Inverse Video Timing Figure 3-15 for timing relationships of Video Circuit.

3.1.9 Keyboard

The keyboard interface of the Model 4P consists of open collector drivers which drive an 8 by 8 key matrix keyboard and an inverting buffer which buffers the key or keys pressed on the data bus. The open collector drivers (U56 and U57 (7416) are driven by address lines A0-A7 which drive the column lines of the keyboard matrix. The ROW lines of the keyboard are pulled up by a 1.5 kohm resistor pack RP2. The ROW lines are buffered and inverted onto the data bus by U58 (74LS240) which is enabled when KEYBD* is a logic low. KEYBD* is a memory mapped decode of addresses 3800-3BFF in Model III Mode and F400-F7FF in Model 4/4P mode. Refer to the Memory Map under Address Decode for more information. During real time operation, the CPU will scan the keyboard periodically to check if any keys are pressed. If no key is pressed, the resistor pack RP2 keeps the inputs of U58 at a logic high. U58 inverts the data to a logic low and buffers it to the data bus which is read by the CPU. If a key is pressed when the CPU scans the correct column line, the key pressed will pull the corresponding row to a logic low. U58 inverts the signal to a logic high which is read by the CPU.

3.1.10 Real Time Clock

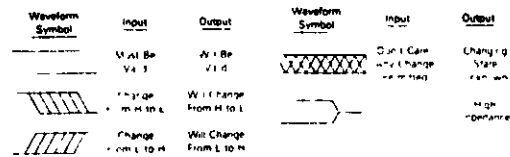
The Real Time Clock circuit in the Model 4P provides a 30 Hz (in the 2 MHz CPU mode) or 60 Hz (in the 4 MHz CPU mode) interrupt to the CPU. By counting the number of interrupts that have occurred, the CPU can keep track of the time. The 60 Hz vertical sync signal (VSYNC) from the video circuitry is used for the Real Time Clock's reference. In the 2 MHz mode, FAST is a logic low which sets the Preset input, pin 4 of U22 (74LS74), to a logic high. This allows the 60 Hz (VSYNC) to be divided by 2 to 30 Hz. The output of 1/2 of U22 is ORed with the original 60 Hz and then clocks another 74LS74 (1/2 of U22). If the real time clock is enabled (ENRTC at a logic high), the interrupt is latched and pulls the INT* line low to the CPU. When the CPU recognizes the interrupt, the pulse is counted and the latch reset by pulling RTCIN* low. In the 4 MHz mode, FAST is a logic high which keeps the first half of U22 in a preset state (the Q* output at a logic low). The 60 Hz is used to clock the interrupts.

NOTE: If interrupts are disabled, the accuracy of the real time clock will suffer.

3.1.11 Line Printer Port

The Line Printer Port Interface consists of a pulse generator, an eight-bit latch, and a status line buffer. The status of the line printer is read by the CPU by enabling buffer U3 (74LS244). This buffer is enabled by LPRD* which is a memory map and port map decode. In Model III mode, only the status can be read from memory location 37E8 or 37E9. The status can be read in all modes by an input from ports F8-FB. For a listing of the bit status, refer to Port Map section.

After the printer driver software determines that the printer is ready for printing (by reading the correct status), the characters to be printed are output to Port F8-FB. U2, a 74LS374 eight-bit latch, latches the character byte and outputs to the line printer. One-half of U1 (74LS123), a one-shot, is then triggered which generates an appropriate strobe signal to the printer which signifies a valid character is ready. The output of the one-shot is buffered by 1/6th of the U21 (74LS04) to prevent noise from the printer cable from false-triggering the one-shot.



Hardware 88

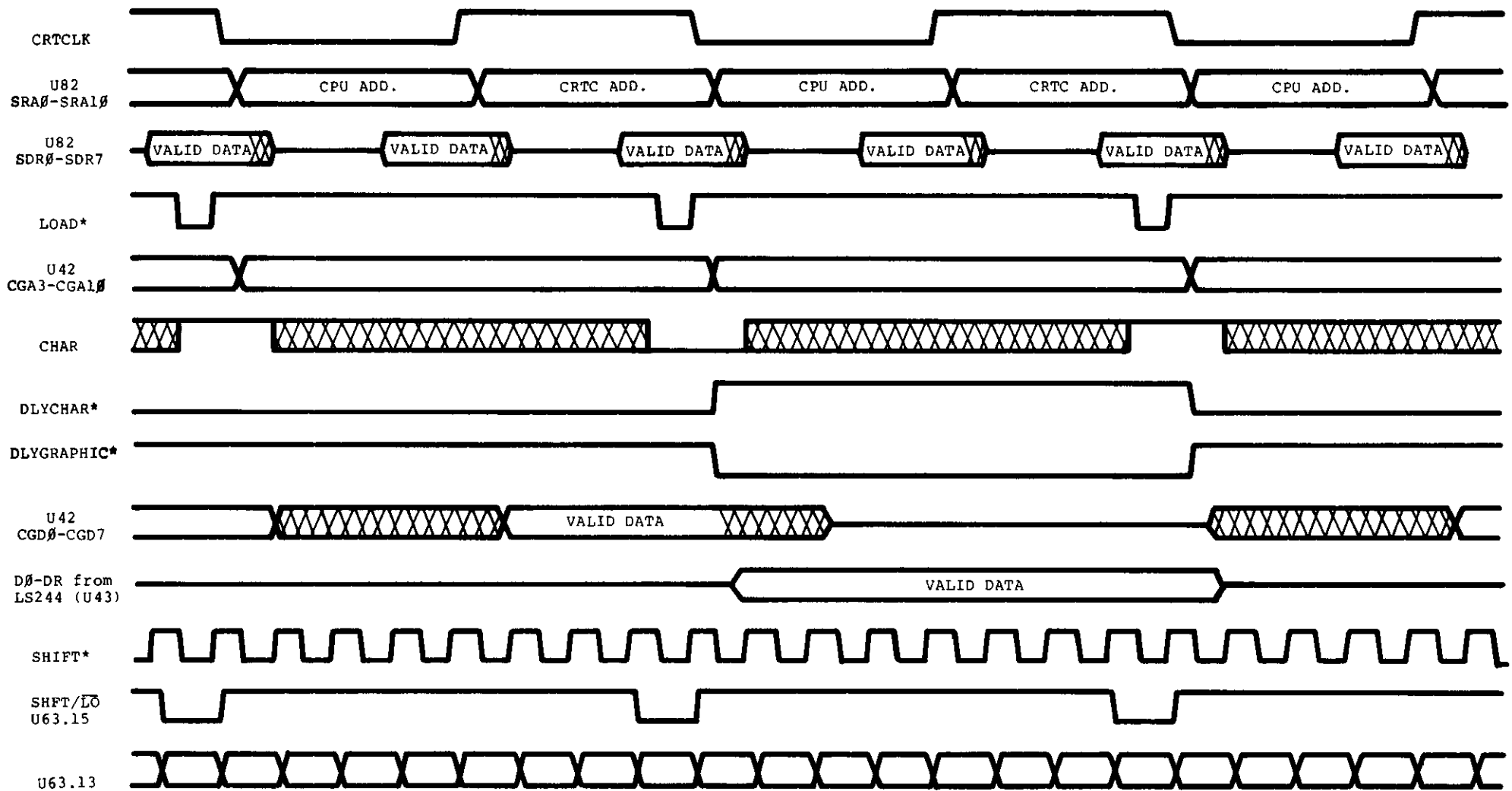


Figure 3-13. Video Circuit Timing

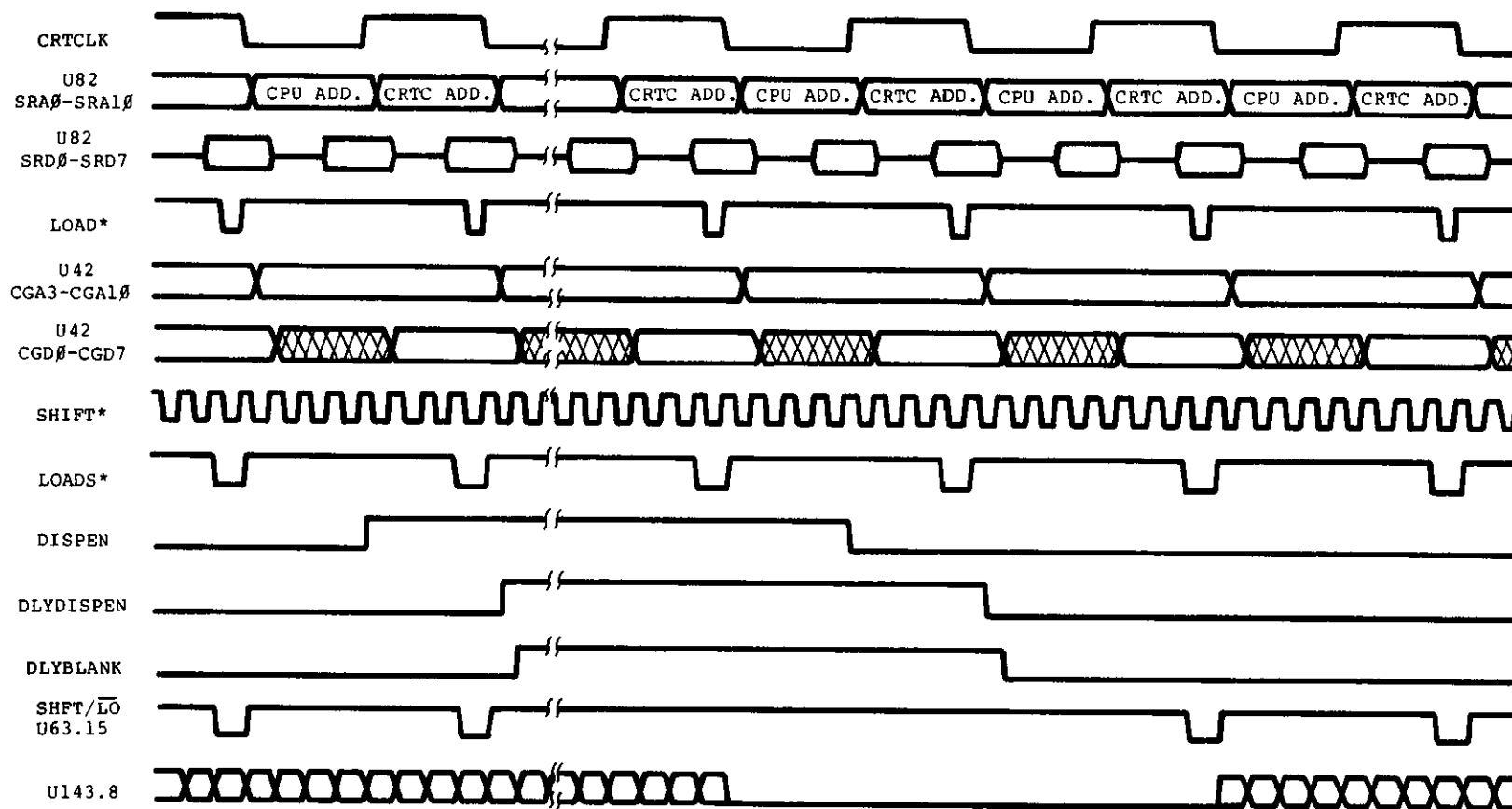


Figure 3-14. Video Blanking Timing

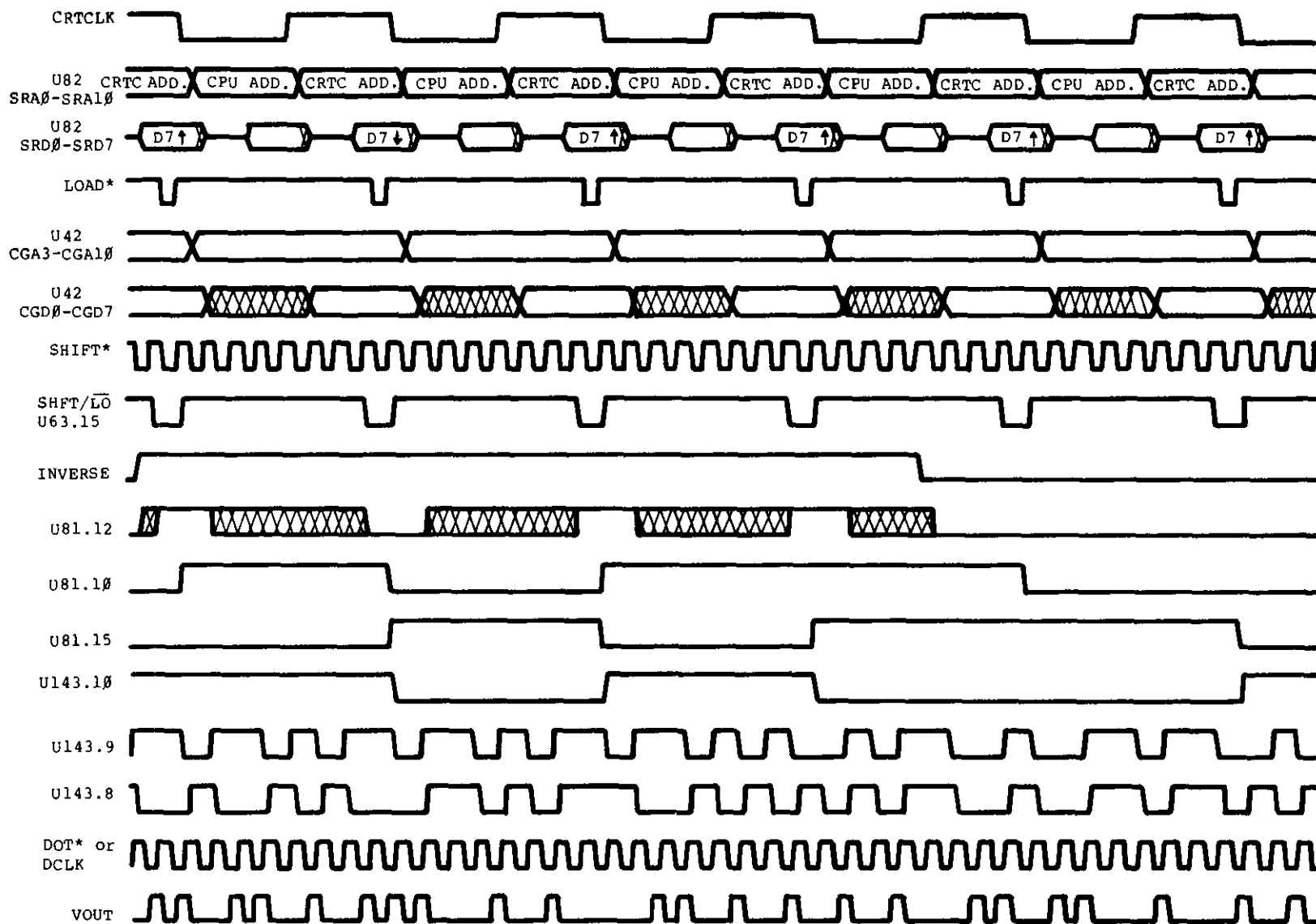


Figure 3-15. Inverse Video Timing

3.1.12 Graphics Port

The Graphics Port (J7) on the Model 4P is provided to attach the optional Graphics Board. The port provides D0-D7 (Data Lines), A0-A3 (Address Lines), IN*, GEN* and RESET* for the necessary interface signals for the Graphics Board. GEN* is generated by negative ORing Port selects GSEL0* (8C-8FH) and GSEL1* (80-83H) together by (1 4 of 74LS08) U23. The resulting signal is negative ANDed with IORQ* by (1 4 of 74S32) U62. Seven timing signals are provided to allow synchronization of Main Logic Board Video and Graphics Board Video. These timing signals are VSYNC, HSYNC, DISPEN, DCLK, H, I, and J. Three control signals from the Graphics Board are used to sync to CPU access and select different video modes. WAIT* controls the CPU access by causing the CPU to WAIT till video is in retrace area before allowing any writes or reads to Graphics Board RAM. ENGRAF is asserted when Graphics video is displayed. ENGRAF also disables inverse video mode on Main Logic Board Video. CL166* (Clear 74L166) is used to enable or disable mixing of Main Logic Board Video and Graphics Board Video. If CL166* is negated high, then mixing is allowed in all for video modes 80 x 24, 40 x 24, 64 x 16, and 32 x 16. If CL166* is asserted low, this will clear the video shift register U63, which allows no video from the Main Logic Board. In this state 8064* is automatically asserted low to put screen in 80 x 24 video mode. Refer to Figure 3-16 Graphic Board Video Timing for timing relationships. Refer to the Model 4/4P Graphics Board Service information for service or technical information on the Graphics Board.

3.1.13 Sound

The sound circuit in the Model 4P is compatible with the Sound Board which was optional in the Model 4. Sound is generated by alternately setting and clearing data bit D0 during an OUT to port 90H. The state of D0 is latched by U130 (1/2 of a 74LS74) and the output is amplified by Q2 which drives a piezoelectric sound transducer. The speed of the software loop determines the frequency, and thus, the pitch of the resulting tone. Since the Model 4P does not have a cassette circuit, some existing software that used the cassette output for sound would have been lost. The Model 4P routes the cassette latch to the sound board through U142. When the CASSMOTORON signal is a logic low, the cassette motor is off, then the cassette output is sent to the sound circuit.

3.1.14 I/O Bus Port

The Model 4P Bus is designed to allow easy and convenient interfacing of I/O devices to the Model 4P. The I/O Bus supports all the signals necessary to implement a device compatible with the Z80s I/O structure.

Addresses

A0 to A7 allow selection of up to 256* input and 256 output devices if external I/O is enabled.

*Ports 80H to 0FFH are reserved for System use.

Data

DB0 to DB7 allow transfer of 8-bit data onto the processor data bus if external I/O is enabled.

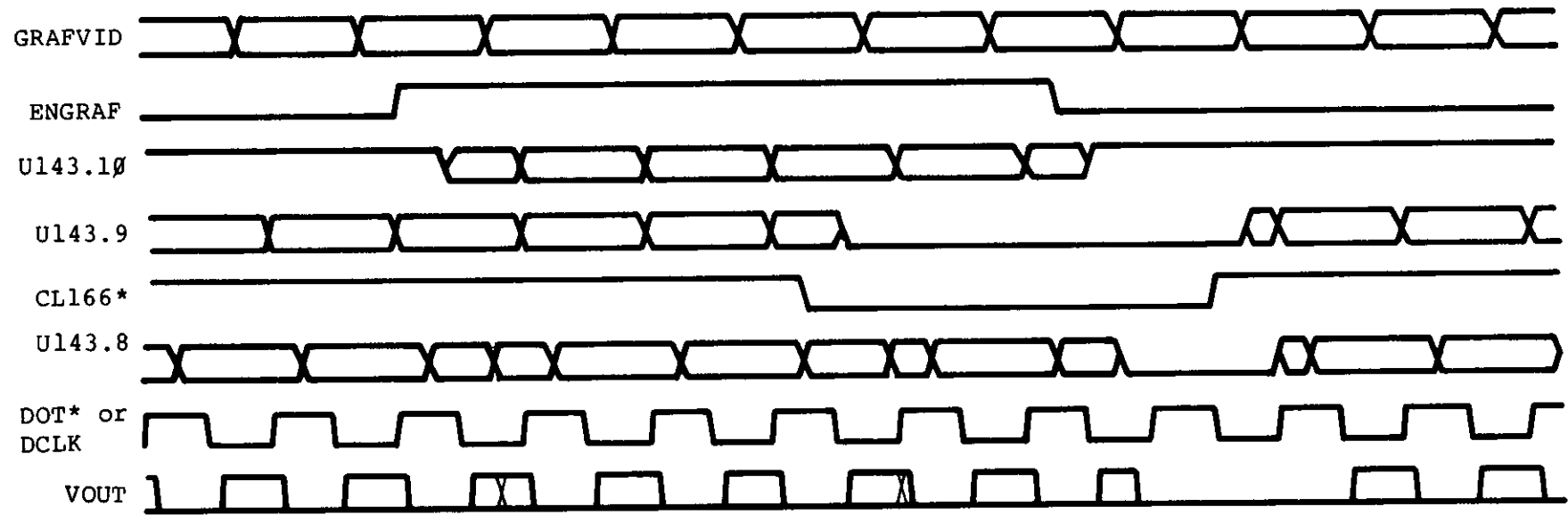
Control Lines

- 1 M1* — Z80A signal specifying an M1 or Operation Code Fetch Cycle or with IOREQ*, it specifies an Interrupt acknowledge.
- 2 IN* — Z80A signal specifying that an input is in progress. Logic AND of IOREQ* and WR*.
- 3 OUT* — Z80A signal specifying that an output is in progress. Logic AND of IOREQ* and WR*.
- 4 IOREQ* — Z80A signal specifying that an input or output is in progress or with M1*, it specifies an interrupt acknowledge.
- 5 RESET* — system reset signal.
- 6 IOBUSINT* — input to the CPU signaling an interrupt from an I/O Bus device if I/O Bus interrupts are enabled.
- 7 IOBUSWAIT* — input to the CPU wait line allowing I/O Bus device to force wait states on the Z80 if external I/O is enabled.
- 8 EXTIOSEL* — input to I/O Bus Port circuit which switches the I/O Bus data bus transceiver and allows an INPUT instruction to read I/O Bus data.

The address line, data line, and all control lines except RESET* are enabled only when the ENEXIO bit in port EC is set to one.

To enable I/O interrupts, the ENIOBUSINT bit in the PORT E0 (output port) must be a one. However, even if it is disabled from generating interrupts, the status of the IOBUSINT* line can still read on the appropriate bit of CPU IOPORT E0 (input port).

See Model 4P Port Bit assignments for port 0FF, 0EC, and 0E0.

**Figure 3-16. Graphic Board Video Timing**

The Model 4P CPU board is fully protected from foreign I/O devices in that all the I/O Bus signals are buffered and can be disabled under software control. To attach and use an I/O device on the I/O Bus, certain requirements (both hardware and software) must be met.

For input port device use, you must enable external I/O devices by writing to port 0ECH with bit 4 on in the user software. This will enable the data bus address lines and control signals to the I/O Bus edge connector. When the input device is selected, the hardware should acknowledge by asserting EXTIOSEL* low. This switches the data bus transceiver and allows the CPU to read the contents of the I/O Bus data lines. See Figure 3-17 for the timing. EXTIOSEL* can be generated by NANDing IN and the I/O port address.

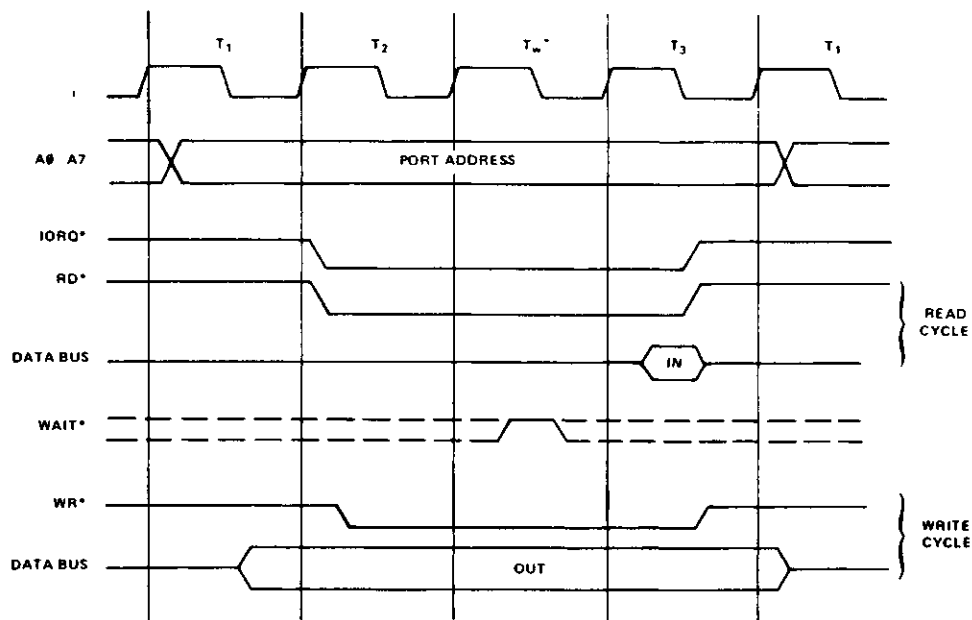
Output port device use is the same as the input port device in use, in that the external I/O devices must be enabled by writing to port 0ECH with bit 4 on in the user software — in the same fashion.

For either input or output devices, the IOBUSWAIT* control line can be used in the normal way for synchronizing slow devices to the CPU. Note that since dynamic memories are used in the Model 4P, the wait line should be used with caution. Holding the CPU in a wait state for 2 msec or more may cause loss of memory contents since refresh is inhibited during this time. It is recommended that the IOBUSWAIT* line be held active no more than 500 μ sec with a 25% duty cycle.

The Model 4P will support Z80 Mode 1 interrupts. A RAM jump table is supported by the LEVEL II BASIC ROMs image and the user must supply the address of his interrupt service routine by writing this address to locations 403E and 403F. When an interrupt occurs, the program will be vectored to the user-supplied address if I/O Bus interrupts have been enabled. To enable I/O Bus interrupts, the user must set bit 3 of Port 0E0H.

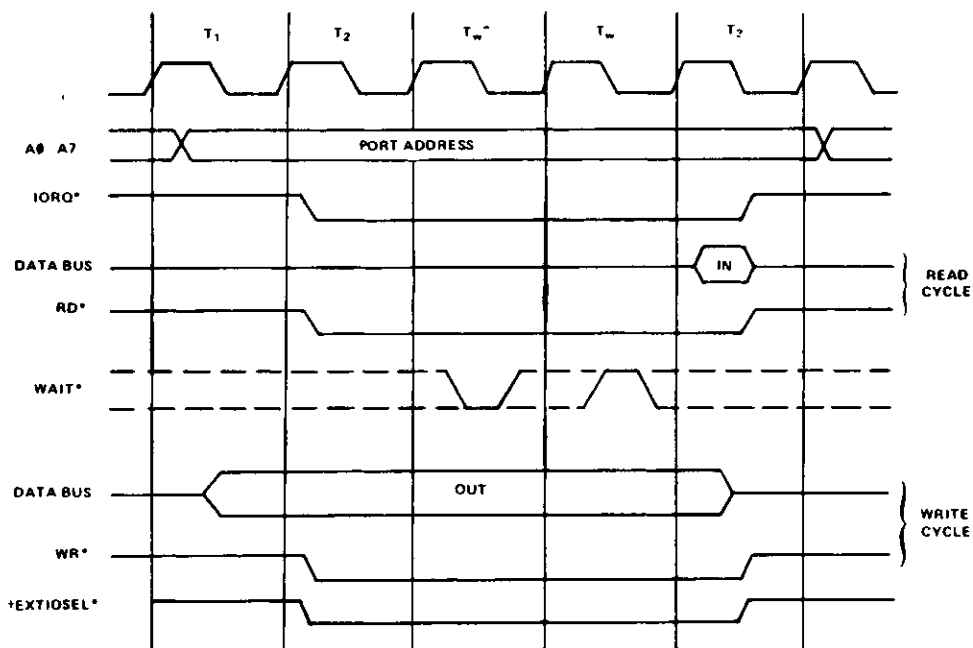
3.1.15 FDC Circuit

The TRS-80 Model 4P Floppy Disk Interface provides a standard 5-1/4" floppy disk controller. The Floppy Disk Interface supports both single and double density encoding schemes. Write precompensation can be software enabled or disabled beginning at any track, although the system software enables write precompensation for all tracks greater than twenty-one. The amount of write precompensation is 250 nsec and is not adjustable. The data clock recovery logic incorporates a digital data separator which achieves state-of-the-art reliability. One or two drives may be controlled by the interface. All data transfers are accomplished by CPU data requests. In double density operation, data transfers are synchronized to the CPU by forcing a wait to the CPU and clearing the wait by a data request from the FDC chip. The end of the data transfer is indicated by generation of a non-maskable interrupt from the interrupt request output of the FDC chip. A hardware watchdog timer insures that any error condition will not hang the wait line to the CPU for a period long enough to destroy RAM contents.



*Inserted by Z80 CPU

Input or Output Cycles with Wait States.



*Inserted by Z80 CPU

†Coincident with IORQ* only on INPUT cycle

Figure 3-17. I/O Bus Timing Diagram

The Floppy Disk Controller Board is an I/O port-mapped device which utilizes ports E4H, F0H, F1H, F2H, F3H, and F4H. The decoding logic is implemented on the CPU board. (Refer to Paragraph 5.1.5 Address Decoding for more information on Port Map). U31 is a bi-directional 8-bit transceiver used to buffer data to and from the FDC and RS-232 circuits. The direction of data transfer is controlled by the combination of control signals DISKIN* and RS232IN*. If either signal is active (logic low), U31 is enabled to drive data onto the CPU data bus. If both signals are inactive (logic high), U31 is enabled to receive data from the CPU board data bus. A second buffer (U12) is used to buffer the FDC chip data to the FDC RS232 Data Bus (BD0-BD7). U12 is enabled all the time and its direction is controlled by DISKIN*. Again, if DISKIN* is active (logic low), data is enabled to drive from the FDC chip to the Main Data Busses. If DISKIN* is inactive (logic high), data is enabled to be transferred to the FDC chip.

Nonmaskable Interrupt Logic

Dual D flip-flop U100 (74LS74) is used to latch data bits D6 and D7 on the rising edge of the control signal WRNMIMASKREG*. The outputs of U100 enable the conditions which will generate a non-maskable interrupt to the CPU. The NMI interrupt conditions which are programmed by doing an OUT instruction to port E4H with the appropriate bits set. If data bit 7 is set, an FDC interrupt is enabled to generate an NMI interrupt. If data bit 7 is reset, interrupt requests from the FDC are disabled. If data bit 6 is set, a Motor Time Out is enabled to generate an NMI interrupt. If data bit 6 is reset, interrupts on Motor Time Out are disabled. An IN instruction from port E4H enables the CPU to determine the source of the non-maskable interrupt. Data bit 7 indicates the status of FDC interrupt request (INTRQ) (0 = true, 1 = false). Data bit 6 indicates the status of Motor Time Out (0 = true, 1 = false). Data bit 5 indicates the status of the Reset signal (0 = true, 1 = false). The control signal RDNMISTATUS* gates this status onto the CPU data bus when active (logic low).

Drive Select Latch and Motor ON Logic

Selecting a drive prior to disk I/O operation is accomplished by doing an OUT instruction to port F4H with the proper bit set. The following table describes the bit allocation of the Drive Select Latch.

Data Bit	Function
D0	Selects Drive 0 when set*
D1	Selects Drive 1 when set*
D2	Selects Drive 2 when set*
D3	Selects Drive 3 when set*
D4	Selects Side 0 when reset Selects Side 1 when set
D5	Write precompensation enabled when set disabled when reset
D6	Generates WAIT if set
D7	Selects MFM mode if set Selects FM mode if reset

Hex D flip-flop U32 (74LS174) latches the drive select bits side select and FM* MFM bits on the rising edge of the control signal DRVSEL*. A dual D flip-flop (U98) is used to latch the Wait Enable and Write precompensation enable bits on the rising edge of DRVSEL*. The rising edge of DRVSEL* also triggers a one-shot (1/2 of U54, 74LS123) which produces a Motor On to the disk drives. The duration of the Motor On signal is approximately three seconds. The spindle motors are not designed for continuous operation. Therefore, the inactive state of the Motor On signal is used to clear the Drive Select Latch, which de-selects any drives which were previously selected. The Motor On one-shot is retriggerable by simply executing another OUT instruction to the Drive Select Latch.

Wait State Generation and WAITIMOUT Logic

As previously mentioned, a wait state to the CPU can be initiated by an OUT to the Drive Select Latch with D6 set. Pin 5 of U98 will go high after this operation. This signal is inverted by 1/4th of U79 and is routed to the CPU where it forces the Z80A into a wait state. The Z80A will remain in the wait state as long as WAIT* is low. Once initiated, the WAIT* will remain low until one of five conditions is satisfied. One half of U77 (a five input NOR gate) is used to perform this function. INTQ, DRQ, RESET, CLRWAIT, and WAITIMOUT are the inputs to the NOR gate. If any one of these inputs is active (logic high), the output of the NOR gate (U77 pin 5) will go low. This output is tied to the clear input of the wait latch. When this signal goes low, it will clear the Q output (U98 pin 5) and set the Q* output (U98 pin 6). This condition causes WAIT* to go high which allows the Z80 to exit the wait state. U99 is a 12-bit binary counter which serves as a watchdog timer to insure that a wait condition will not persist long enough to destroy dynamic RAM contents. The counter is clocked by a 1 MHz clock and is enabled to count when its reset pin is low (U99 pin 11). A logic high on U99 pin 11 resets the counter outputs. U99 pin 15 is a divide-by-1024 output and is used to generate the signal WAITIMOUT. This watchdog timer logic will limit the duration of a wait to 1024μsec, even if the FDC chip should fail to generate a DRQ or an INTRQ.

If an OUT to Drive Select Latch is initiated with D6 reset (logic low), a WAIT is still generated. The 12-bit binary counter will count to 2 which will output CLRWAIT and clear the WAIT state. This allows the WAIT to occur only during the OUT instruction to prevent violating any Dynamic RAM parameters.

NOTE: This automatic WAIT will cause a 1-2 μsec wait each time an out to Drive Select Latch is performed.

Clock Generation Logic

A 4 MHz crystal oscillator and a 4-bit binary counter are used to generate the clock signals required by the FDC board. The 4 MHz oscillator is implemented with two inverters (1 3 of U39) and a quartz crystal (Y2). The output of the oscillator is inverted and buffered by 1 6 of U39 to generate a TTL level square wave signal. U37 is a 4-bit binary counter which is divided into a divide-by-2 and a divide-by-8 section. The divide-by-2 section is used to generate the 2 MHz output at pin 12. The 2 MHz is NANDed with 4MHz by 1 4 of U19 and the output is used to clock the divide-by-8 section of U37. A 1 MHz clock is generated at pin 9 of U37 which is 90° phase-shifted from the 2 MHz clock. This phase relationship is used to gate the guaranteed Write Data Pulse (WD) to the Write precompensation circuit. The 4 MHz is used to clock the digital data separator U18 and the Write precompensation shift register U55. The 1 MHz clock is used to drive the clock input of the FDC chip (U13) and the clock input of the watchdog timer (U99).

Disk Bus Output Drivers

High current open collector drivers U20 and U56 are used to buffer the output signals from the FDC circuit to the disk drives.

Write Precompensation and Write Data Pulse Shaping Logic

The Write Precompensation logic is comprised of U55 (74LS195), 1 4 of U19 (74LS00), 1 4 of U74 (74LS04) and 1 2 of U77 (74LS260). U55 is a parallel in-serial out shift register and is clocked by 4 MHz which generates a precompensation value of 250 nsec. The output signals EARLY and LATE of the FDC chip (U13) are input to P0 and P2 of the shift register. A third signal is generated by 1 4 of U75 when neither EARLY nor LATE is active low and is input to P1 of U55. WD of the FDC chip is NANDed with 2 MHz to gate the guaranteed Write Data Pulse to U55 for the parallel load signal SHFT LD. When U55 pin 9 is active low, the signals preset at P1-P3 are clocked in on the rising edge of the 4 MHz clock. After U55 pin 9 goes high, the data is shifted out at a 250 nsec rate. EARLY will generate a 250 nsec delay, NOT EARLY AND NOT LATE will generate a 500 nsec delay, and LATE will generate a 750 nsec delay. This provides the necessary precompensation for the write data. As mentioned previously, Write Precompensation is enabled through software by an OUT to the Drive Select Latch with bit 5 set. This sets the Q output of the 74LS74 (U98 pin 9) which is ANDed with DDEN which disables the shift register U55. DDEN disables Write Precompensation in the single density mode. The resulting signal also enables U75 to allow the write data (WD) to bypass the Write Precompensation circuit. The Write Data (WD) pulse is shaped by a one-shot (1 2 of U54) which stretches the data pulses to approximately 500 nsec.

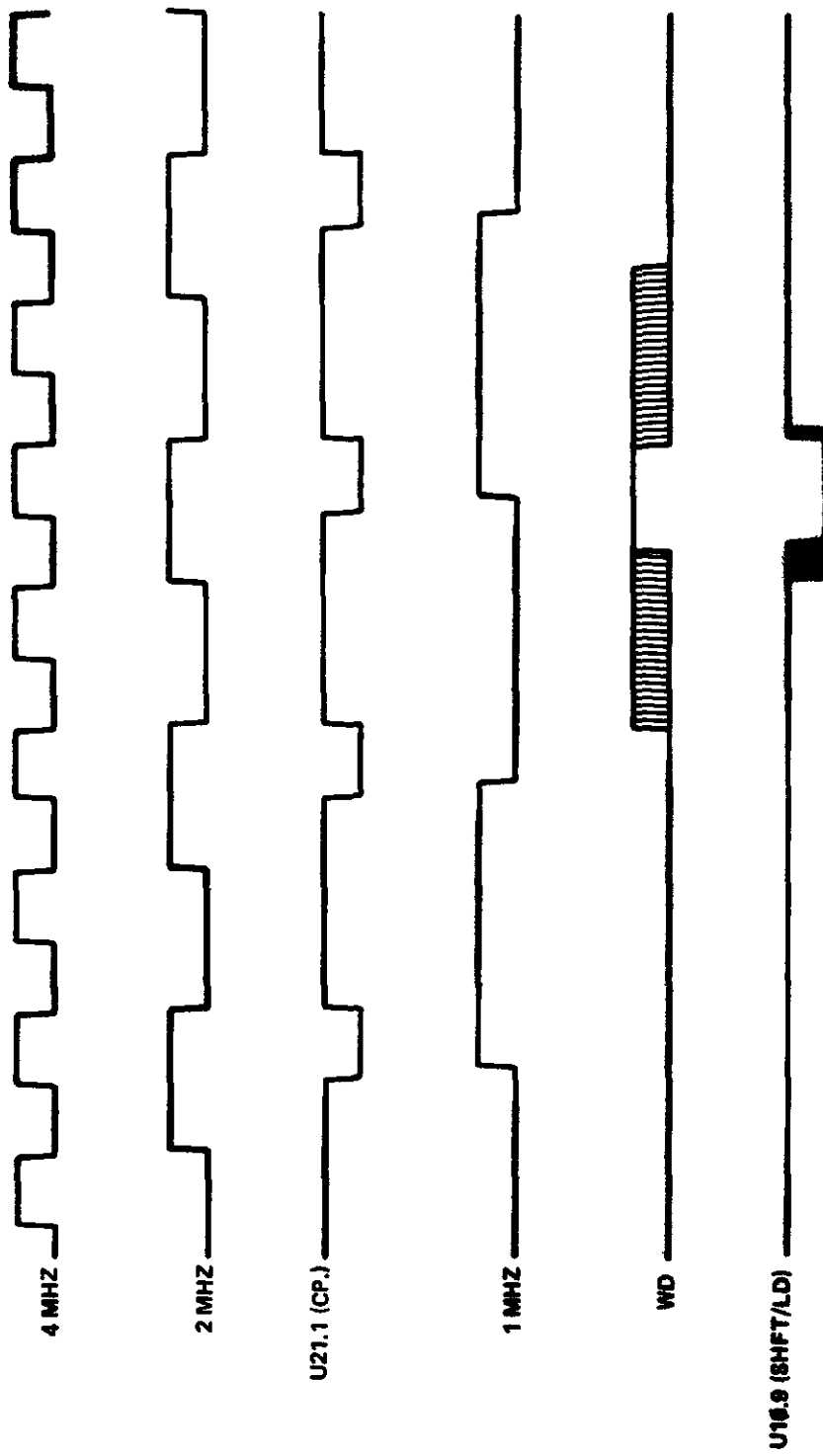


Figure 3-18. Write Precompensation Timing

The Clock and Read Data Recovery Logic is comprised of one chip U18 (FDC9216). The FDC9216 is a Floppy Disk Data Separator (FDDS) which converts a single stream of pulses from the disk drive into separate clock and data pulses for input to the FDC chip. The FDDS consists of a clock divider, a long-term timing corrector, a short-time timing corrector and reclocking circuitry. The reference clock (REFCLK) is a 4 MHz and is divided by the internal clock divider CD0 and CD1 of the FDDS chip control the divisor which divides REFCLK. With DC1 grounded (logic low), CD0 (when a logic low) generates a divide-by-1 for MFM mode and when logic high generates a divide-by-2 for FM mode. CD0 is controlled by the signal DDEN* which is Double Density enable or MFM enable. The FDDS detects the leading edges of RD* pulses and adjusts the phase of the internal clock to generate the separated clock (SEPCLK) to the FDC chip. The separate long and short term timing correctors assure the clock separation to be accurate. The separated Data (SEPD*) is used as the RDD* input to the FDC chip.

Floppy Disk Controller Chip

The 1793 is an MOS LSI device which performs the functions of a floppy disk formatter/controller in a single chip implementation. The following port addresses are assigned to the internal registers of the 1793 FDC chip.

Port No.	Function
F0H	Command Status Register
F1H	Track Register
F2H	Sector Register
F3H	Data Register

3.1.16 RS-232-C Circuit

RS-232C Technical Description

The RS-232C circuit for the Model 4P computer supports asynchronous serial transmissions and conforms to the EIA RS-232C standards at the input-output interface connector (J4). The heart of the circuit is the TR1865 Asynchronous Receiver Transmitter U30. It performs the job of converting the parallel byte data from the CPU to a serial data stream including start stop, and parity bits. For a more detailed description of how this LSI circuit performs these functions, refer to the TR1865 data sheets and application notes. The transmit and receive clock rates that the TR1865 needs are supplied by the Baud Rate Generator U52 (BR1941L) or (BR1943). This circuit takes the 5.0688 MHz supplied by the system timing circuit and the programmed information received from the CPU over the data bus and divides the basic clock rate to provide two clocks. The rates available from the BRG go from 50 Baud to 19200 Baud. See the BRG table for the complete list.

Nibble Loaded	Transmit	16X Clock	Supported by SETCOM
	Receive Baud Rate		
0H	50	0.8 kHz	Yes
1H	75	1.2 kHz	Yes
2H	110	1.76 kHz	Yes
3H	134.5	2.1523 kHz	Yes
4H	150	2.4 kHz	Yes
5H	300	4.8 kHz	Yes
6H	600	9.6 kHz	Yes
7H	1200	19.2 kHz	Yes
8H	1800	28.8 kHz	Yes
9H	2000	32.081 kHz	Yes
AH	2400	38.4 kHz	Yes
BH	3600	57.6 kHz	Yes
CH	4800	76.8 kHz	Yes
DH	7200	115.2 kHz	Yes
EH	9600	153.6 kHz	Yes
FH	19200	307.2 kHz	Yes

The RS-232C circuit is port mapped and the ports used are E8 to EB. Following is a description of each port on both input and output.

Port	Input	Output
E8	Modem status	Master Reset, enables UART control register load
EA	UART status	UART control register load and modem control
E9	Not Used	Baud rate register load enable bit
EB	Receiver Holding register	Transmitter Holding register

Interrupts are supported in the RS-232C circuit by the Interrupt mask register (U92) and the Status register (U44) which allow the CPU to see which kind of interrupt has occurred. Interrupts can be generated on receiver data register full, transmitter register empty, and any one of the errors — parity, framing, or data overrun. This allows a minimum of CPU overhead in transferring data to or from the UART. The interrupt mask register is port E0 (write) and the interrupt status register is port E0 (read). Refer to the IO Port description for a full breakdown of all interrupts and their bit positions.

All Model I, III, and 4 software written for the RS-232-C interface is compatible with the Model 4P RS-232-C circuit, provided the software does not use the sense switches to configure the interface. The programmer can get around this problem by directly programming the BRG and UART for the desired configuration or by using the SETCOM command of the disk operating system to configure the interface. The TRS-80 RS-232C Interface hardware manual has a good discussion of the RS-232C standard and specific programming examples (Catalog Number 26-1145).

Pinout Listing

The following list is a pinout description of the DB-25 connector (P1).

Pin No.	Signal
1	PGND (Protective Ground)
2	TD (Transmit Data)
3	RD (Receive Data)
4	RTS (Request to Send)
5	CTS (Clear To Send)
6	DSR (Data Set Ready)
7	SGND (Signal Ground)
8	CD (Carrier Detect)
19	SRTS (Spare Request to Send)
20	DTR (Data Terminal Ready)
22	RI (Ring Indicate)



SECTION IV

4P GATE ARRAY THEORY OF OPERATION

1



4.2 MODEL 4P GATE ARRAY THEORY OF OPERATION

4.2.1 Introduction

Contained in the following paragraphs is a description of the component parts of the Model 4P CPU Gate Array. It is divided into the logical operational functions of the computer. All components are located on the Main CPU board inside the case housing. Refer to Section 3 for disassembly/assembly procedures.

4.2.2 Reset Circuit

The Model 4P reset circuit provides the necessary reset pulses to all circuits during power up and reset operations. R25 and C214 provide a time constant which holds the input of U121 low during power-up. This allows power to be stable to all circuits before the RESET* and RESET signals are applied. When C214 charges to a logic high, the output of U121 triggers the input of a retriggerable one-shot multivibrator (U1). U1 outputs a pulse with an approximate width of 70 microseconds. When the reset switch is pressed on the front panel, this discharges C214 and holds the input of U121 low until the switch is released. On release of the switch, C214 again charges up, triggering U121 and U1 to reset the microcomputer. Another signal POWRST* is generated to clear drive select circuit immediately when reset switch is pressed.

4.2.3 CPU

The central processing unit (CPU) of the Model 4P microcomputer is a Z80A microprocessor. The Z80A is capable of running in either 2 MHz or 4 MHz mode. The CPU controls all functions of the microcomputer through use of its address lines (A0-A15), data lines (D0-D7), and control lines (/M1, /IOREQ, RD, WR, /MREQ, and /RFSH). The address lines (A0-A15) are buffered to other ICs through two 74LS244s (U67 and U27) which are enabled all the time with their enables pulled to GND. The control lines are buffered to other ICs through a 74F04 (U87). The data lines (D0-D7) are buffered through a bi-directional 74LS245 (U86) which is enabled by BUSEN* and the direction is controlled by BUSDIR*.

4.2.4 System Timing

The main timing reference of the microcomputer, with the exception of the FDC circuit, is generated by a Gate Array U148 and a 20.2752 MHz Crystal. This reference is internally divided in the Gate Array to generate all necessary timing for the CPU, video circuit, and RS-232-C circuit. The CPU clock is generated U148 which can be either 2 or 4 MHz depending on the logic state of FAST input (pin 6 of U148). If FAST is a logic low, the U148 generates a 2.02752 MHz clock. If FAST is a logic high, U148 generates a 4.05504 MHz signal. PCLK (pin 23 of U148) is filtered through a ferrite bead (FB2) and 22Ω Resistor (R9) and then

fed to the CPU U45. PCLK is generated as a symmetrical clock and is never allowed to be short cycled (eg.) Not allowed to generate a low or high pulse under 110 nanoseconds.

4.2.4.1 Video Timing

The video timing is also generated by U148 with the help of a PLL Multiplier Module (PMM) U146. These two ICs generate all the necessary timing signals for the four video modes: 64 x 16, 32 x 16, 80 x 24, and 40 x 24. Two reference clocks are required for the four video modes. One reference clock is 10.1376 MHz. It is generated internally to U148, and is used by the 64 x 16 and 32 x 16 modes. The second reference clock is a 12.672 MHz (12M) clock which is generated by the PMM U146. 12M clock is used by the 80 x 24 and 40 x 24 modes. A 1.2672 MHz (1.2M16) signal is output from pin 3 of U148 and is generated from the master reference clock, the 20.2752 MHz crystal. 1.2M16 is used for a reference clock for the PMM. The PMM is internally set to oscillate at 12.672 MHz which is output as 12M. U148 divides 12M by 10 to generate a second 1.2672 MHz clock (1.2M10) which is fed into pin 5 of U146 (PMM). The two 1.2672 MHz signals are internally compared in the PMM where it corrects the 12.672 MHz output so it is synchronized with the 20.2752 MHz clock.

MODSEL and 8064* signals are used to select the desired video mode. 8064* controls which reference clock is used by U127 and MODSEL controls the single or double character width mode. Refer to the following chart for selecting each video mode.

8064*	MODSEL	Video Mode
0	0	64 x 16
0	1	32 x 16
1	0	80 x 24
1	1	40 x 24

*This is the state to be written to latch U85. Signal is inverted before being input to U148.

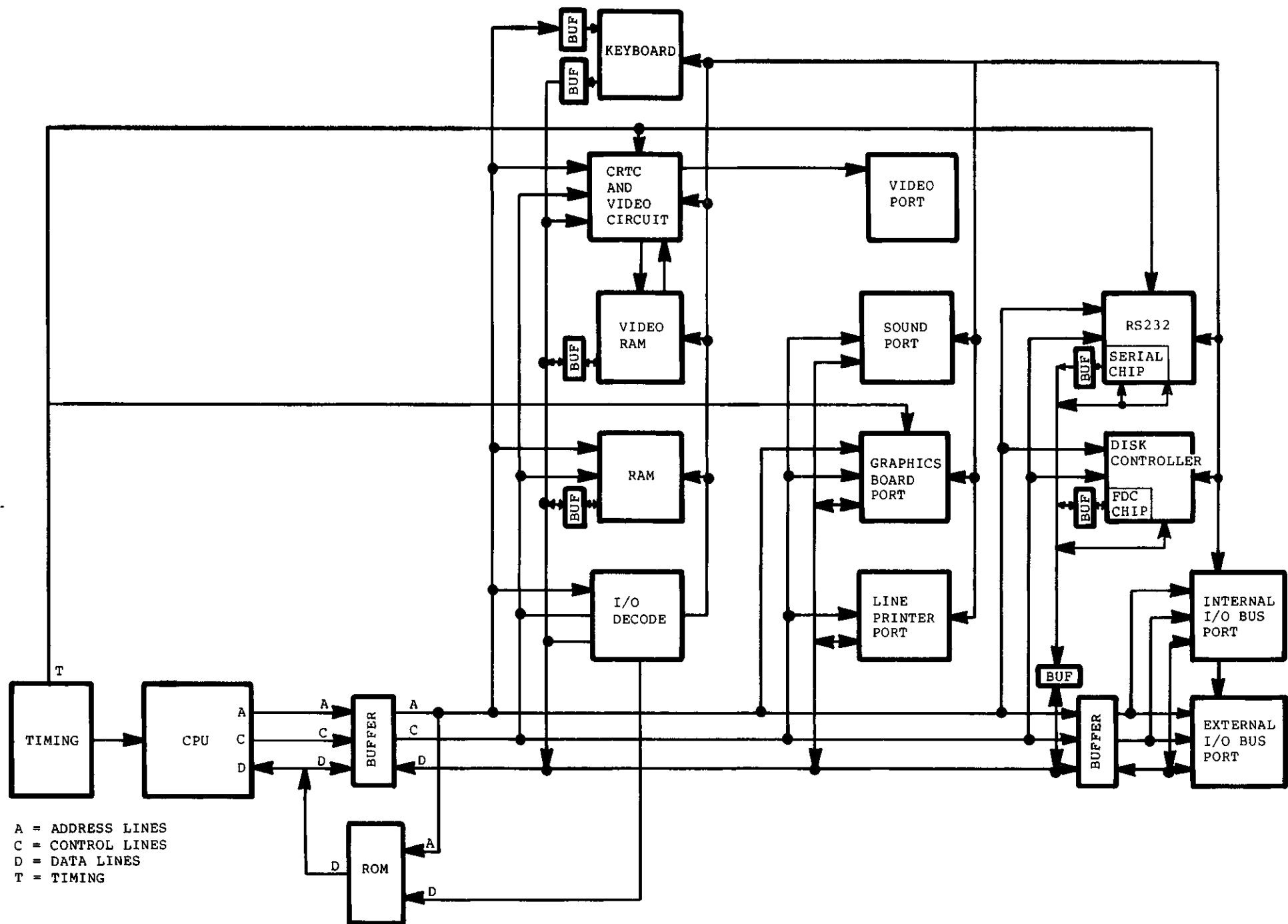


Figure 4-1. Model 4P Functional Block Diagram

DCLK the reference clock selected is output from U148 U148 generates SHIFT* XADR7* CRTCLK LOADS* and LOAD* for proper timing for the four video modes U149 also generated H I and J which are fed to the Graphics Port J7 for reference timings of Hires graphics video Refer to Video Timing Figs 4-2 and 4-3 for timing reference

4.2.5 Address Decode

The Address Decode section will be divided into two subsections Memory Map decoding and Port Map decoding

4.2.5.1 Memory Map Decoding

Memory Map Decoding is accomplished by Gate Array 4 2 (U106) Four memory map modes are available which are compatible with the Model III and Model 4 microcomputers U106 is used for memory map control which also controls page mapping of the 32K RAM pages Refer to Memory Maps below

4.2.5.2 Port Map Decoding

Port Map Decoding is accomplished by Gate Array 4 2 (U106) U106 decodes the low order address (A0 A7) from the CPU and decodes the port being selected The IN* signal allows the CPU to read from a selected port and the OUT* signal allows the CPU to write to the selected port Refer to IO Port Assignment

4.2.6 ROM

The Model 4P contains only a 4K x 8 Boot ROM (U70) This ROM is used only to boot up a Disk Operating System into the RAM memory If Model III operation or DOS is required then the RAM from location 0000-37FFH must be loaded with an image of the Model III or 4 ROM code and then executed A file called MODEL A/III is supplied with the Model 4P which contains the ROM image for proper Model III operation On power-up, the Boot ROM is selected and mapped into location 0000-0FFFH After the Boot Sector or the ROM Image is loaded, the Boot ROM must be mapped out by OUTing to port 9CH with D0 set or by selecting Memory Map modes 2 or 3 In Mode 1 the RAM is write enabled for the full 14K This allows the RAM area mapped where Boot ROM is located to be written to while executing out of the Boot ROM Refer to Memory Maps

The Model 4P Boot ROM contains all the code necessary to initialize hardware detect options selected from the keyboard read a sector from a hard disk or floppy and load a copy of the Model III ROM Image (as mentioned) into the lower 14K of RAM

The firmware is divided into the following routines

- * Hardware Initialization
- * Keyboard Scanner
- * Control
- * Floppy and Hard Disk Driver
- * Disk Directory Searcher
- * File Loader
- * Error Handler and Displayer
- * RS 232 Boot
- * Diagnostic Package

Theory of Operation

This section describes the operation of various routines in the ROM Normally the ROM is not addressable by normal use However there are several routines that are available through fixed calling locations and these may be used by operating systems that are booting

On a power up or RESET condition the Z80's program counter is set to address 0 and the boot ROM is switched in The memory map of the system is set to Mode 0 (See Memory Map for details) This will cause the Z80 to fetch instructions from the boot ROM

The Initialization section of the Boot ROM now performs these functions

- 1 Disables maskable and non maskable interrupts
- 2 Interrupt mode 1 is selected
- 3 Programs the CRT Controller
- 4 Initializes the boot ROM control areas in RAM
- 5 Sets up a stack pointer
- 6 Issues a Force Interrupt to the Floppy Disk Controller to abort any current activity
- 7 Sets the system clock to 4mhz
- 8 Sets the screen to 64 x 16
- 9 Disables reverse video and the alternate character sets
- 10 Tests for key being pressed*
- 11 Clears all 2K of video memory

* This is a special test If the key is being pressed then control is transferred to the diagnostic package in the ROM All other keys are scanned via the Keyboard Scanner

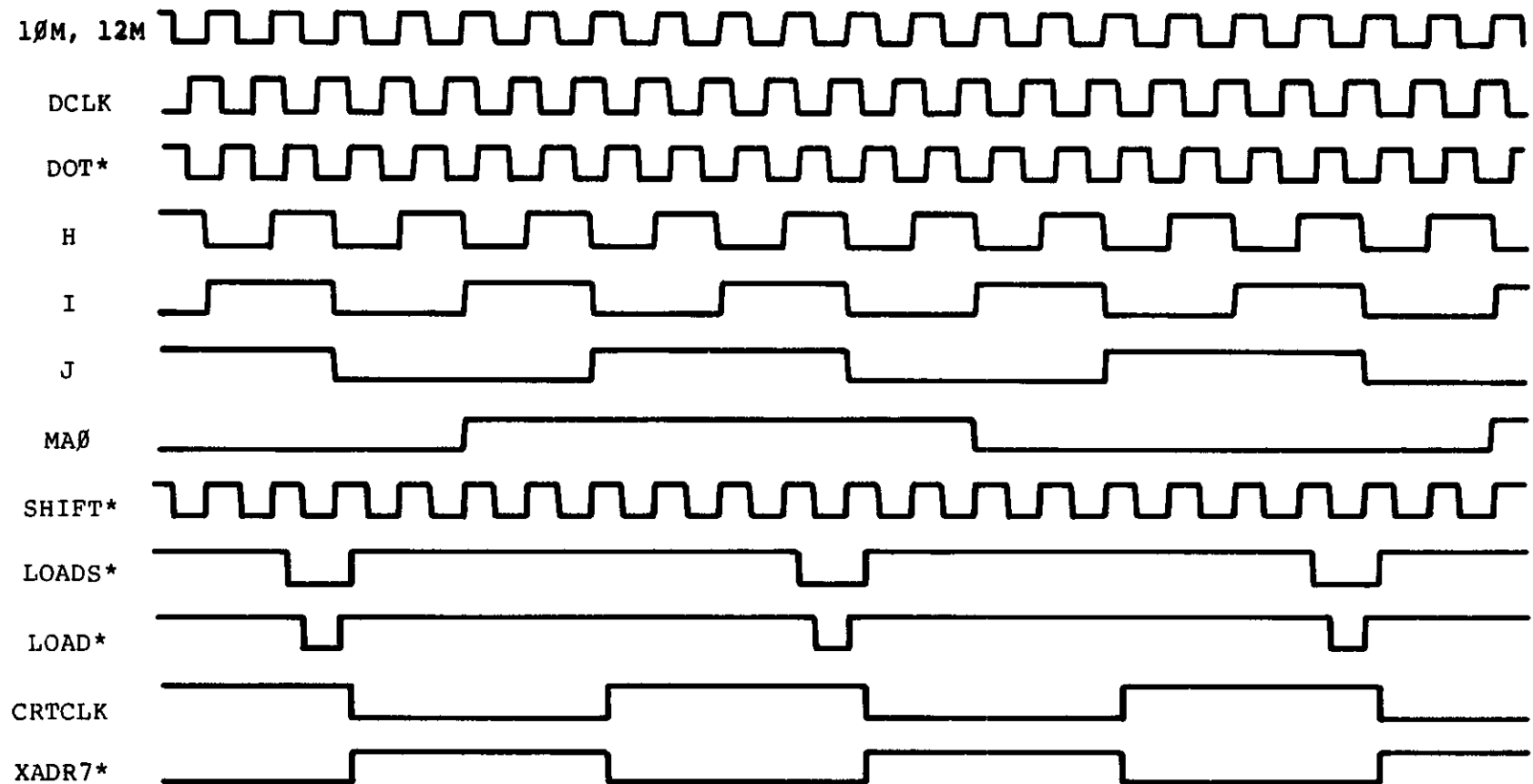


Figure 4-3. Video Timing 32 x 16 Mode 40 x 24 Mode

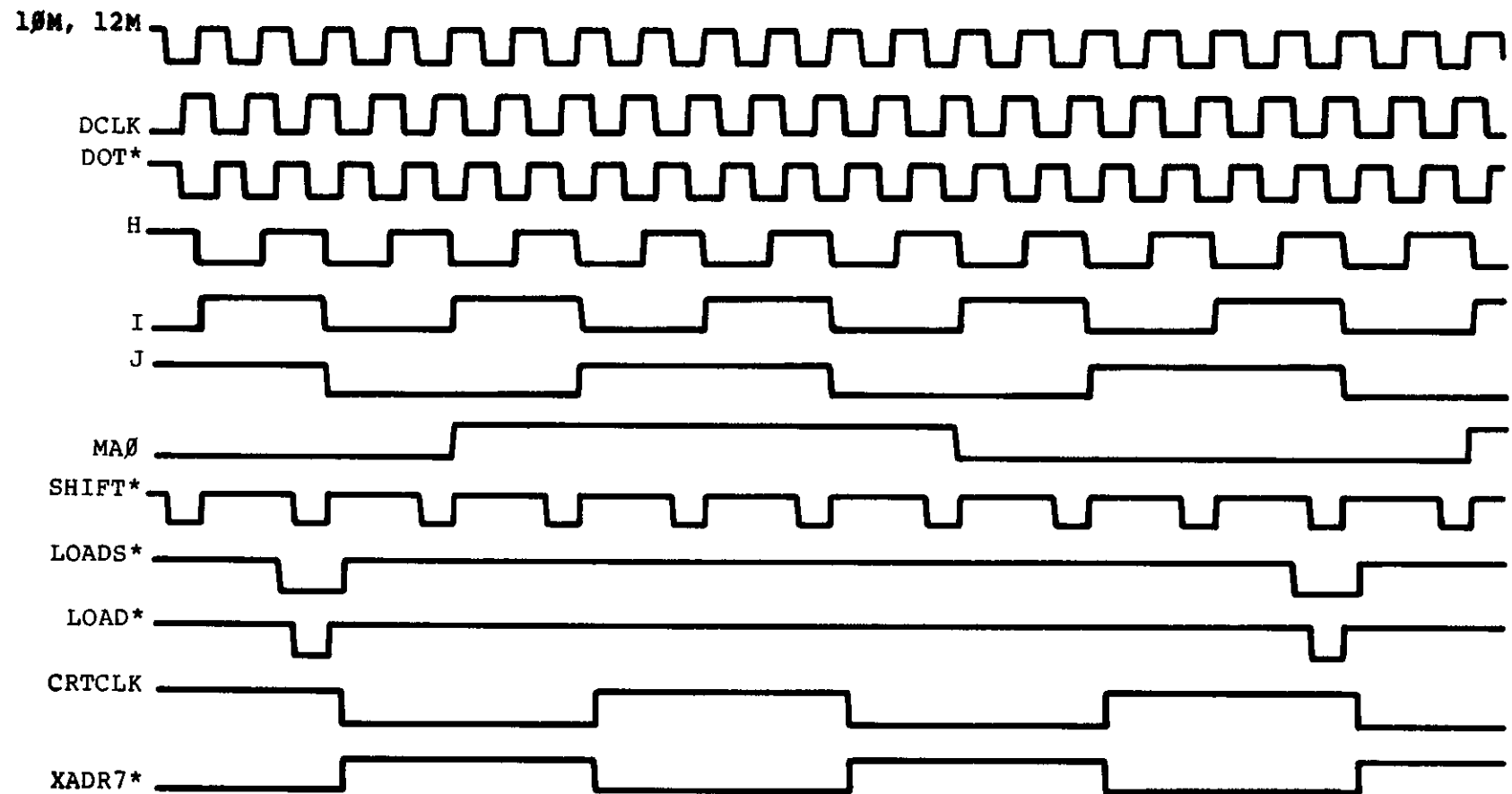


Figure 4-2. Video Timing 64 x 16 Mode 80 x 24 Mode

The Keyboard scanner is now called. It scans the keyboard for a set period of time and returns several parameters based on which, if any, keys were pressed.

The keyboard scanner checks for several different groups of keys. These are shown below:

Function Group	Selection Group
<F1>	A
<F2>	B
<F3>	C
<1>	D
<2>	E
<3>	F
<Left-Shift>	G
<Right-Shift>	
<Ctrl>	
<Caps>	

Special Keys	Misc Keys
<P>	<Enter>
<L>	<Break>
<N>	

When any key in the Function Group is pressed, it is recorded in RAM and will be used by the Control routine in directing the action of the boot. If more than one of these keys are pressed during the keyboard scan, the last one detected will be the one that is used. The Function group keys are currently defined as:

<F1> or <1>	Will cause hard disk boot
<F2> or <2>	Will cause floppy disk boot
<F3> or <3>	Will force Model III mode
<Left-Shift>	Reserved for future use
<Right-Shift>	Boot from RS-232 port
<Ctrl>	Reserved for future use
<Caps>	Reserved for future use

The Special keys are commands to the Control routine which direct handling of the Model III ROM-image. Each key is detected individually.

<P>	When loading the Model III ROM-image, the user will be prompted when the disks can be switched or when ROM BASIC can be entered by pressing <Break>.
<N>	Instructs the Control routine to not load the Model III ROM-image, even if it appears that the operating system being booted requires it.

<L>

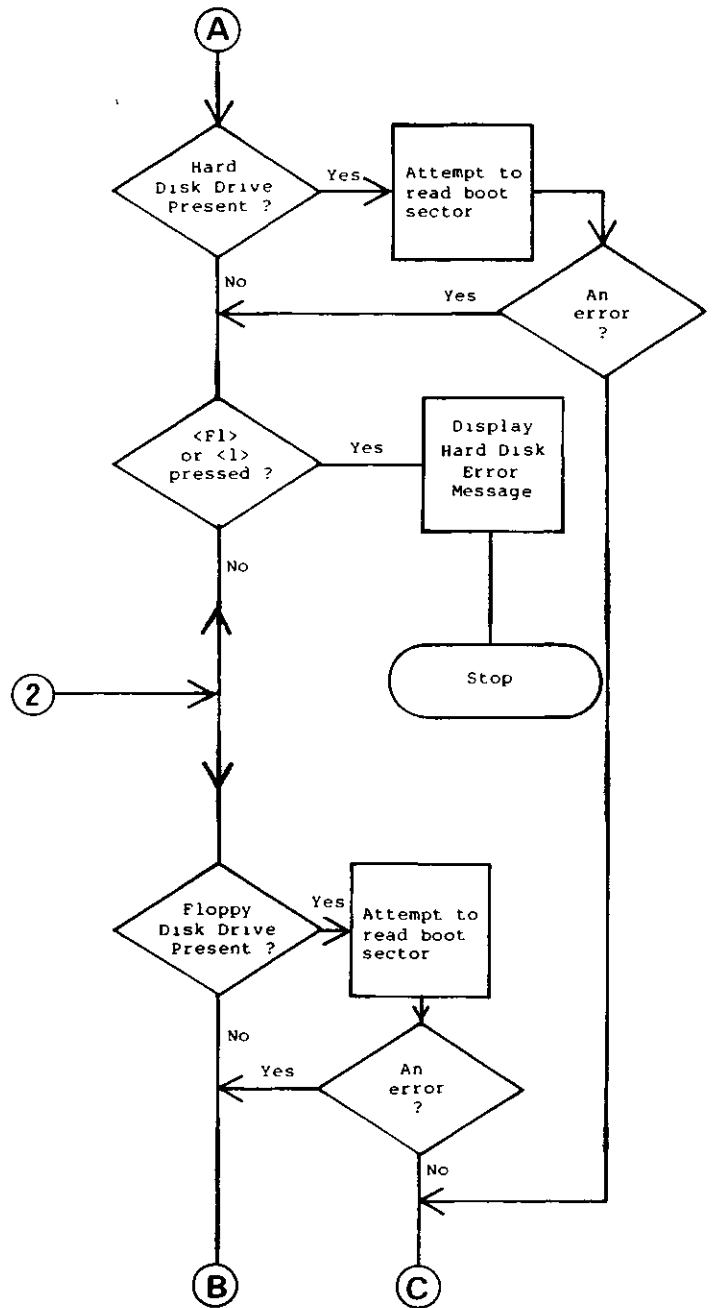
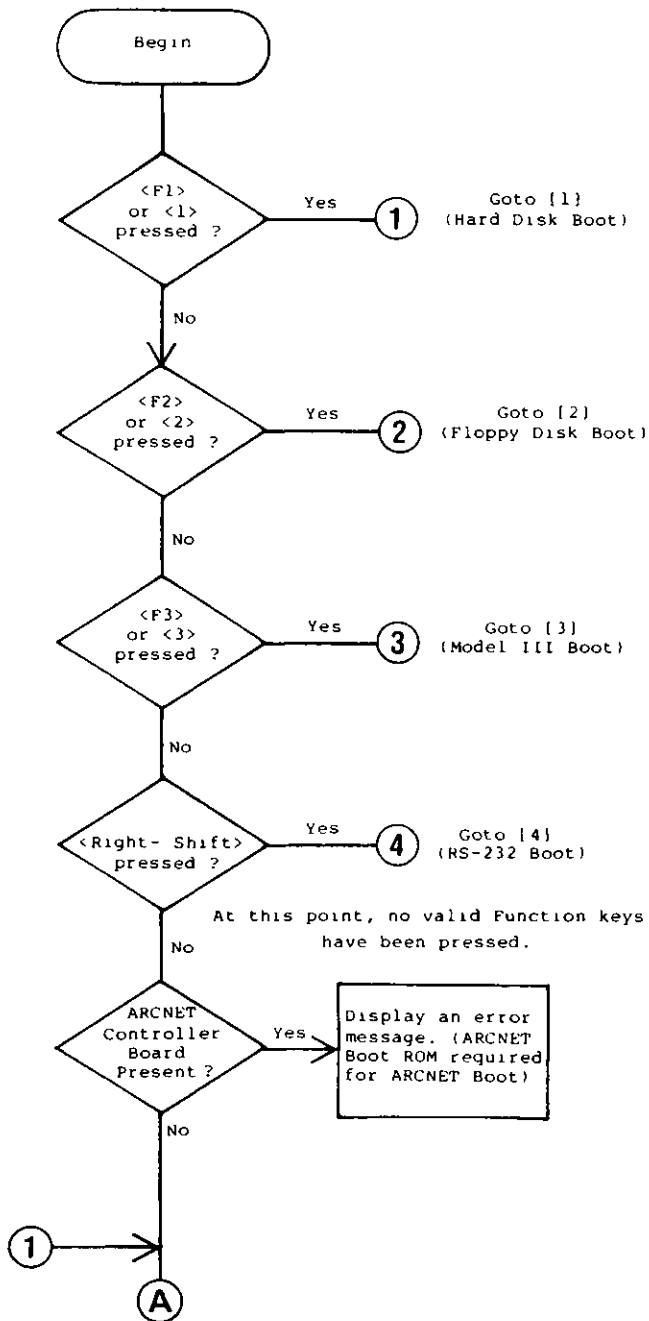
Instructs the Control routine to load the Model III ROM-image, even if it is already loaded. This is useful if the ROM-image has been corrupted or when switching ROM-images. (Note that this will not cause the ROM-image to be loaded if the boot sector check indicates that the Model III ROM image is not needed. Press · F3 · or · F3 · and · L · to accomplish that.

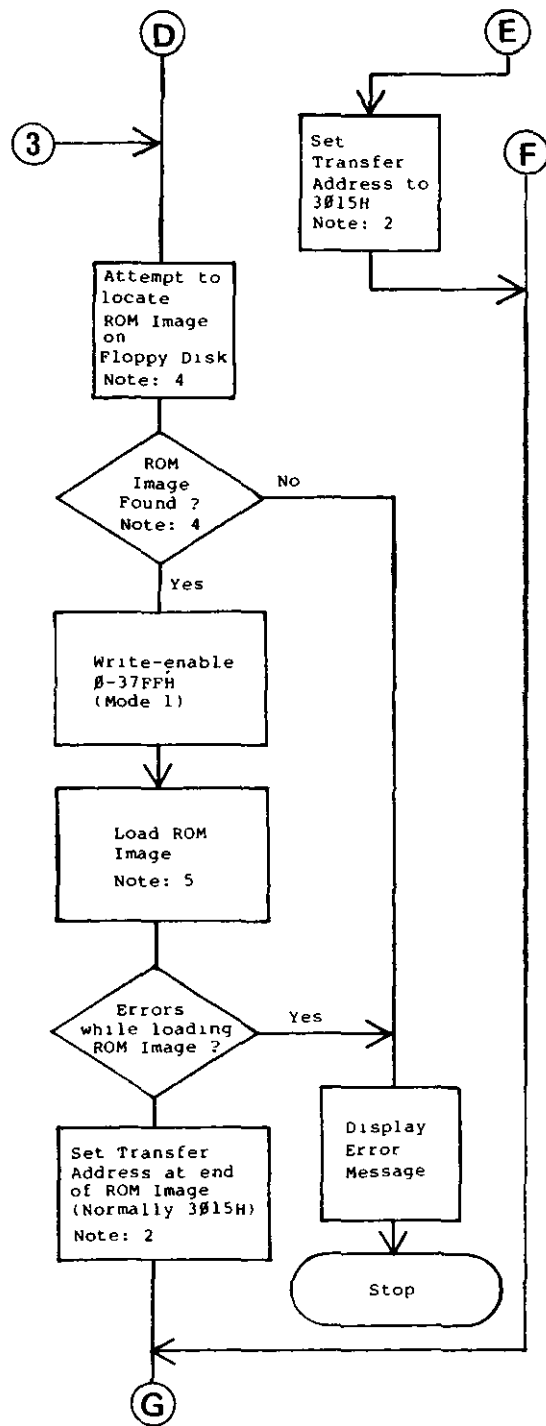
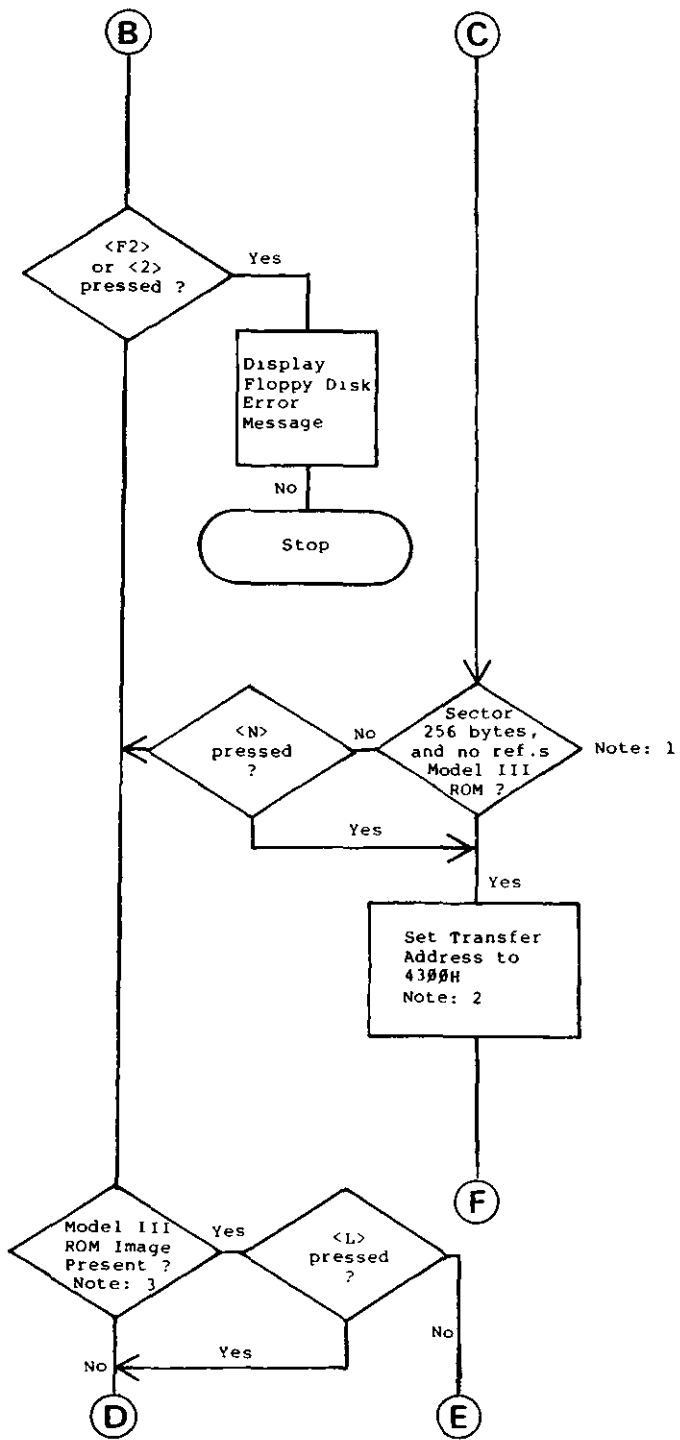
The Selection group keys are used in determining which file will be read from disk when the ROM-image is loaded. For details of this operation, see the Disk Directory Searcher. If more than one of the Selection group keys are pressed, the last one detected will be the one that is used.

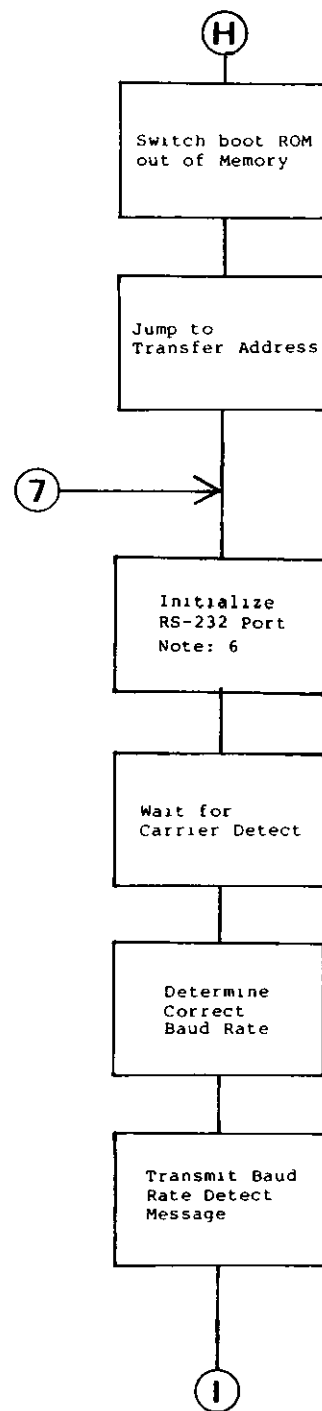
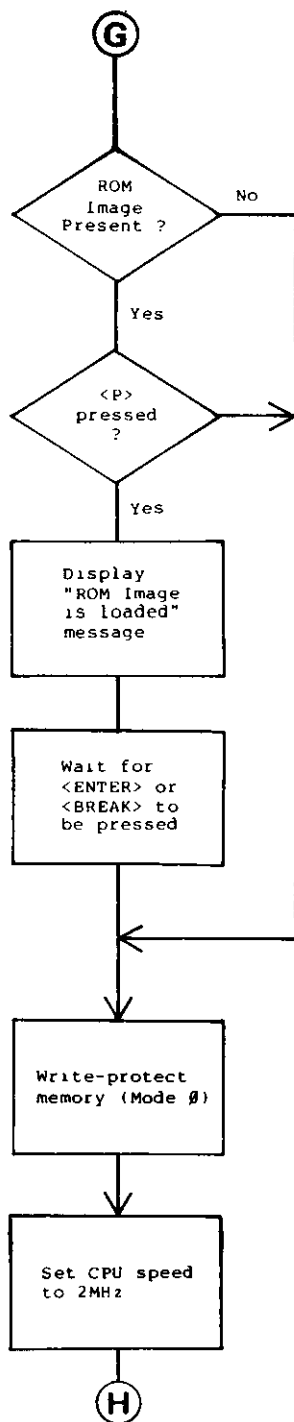
The Miscellaneous keys are:

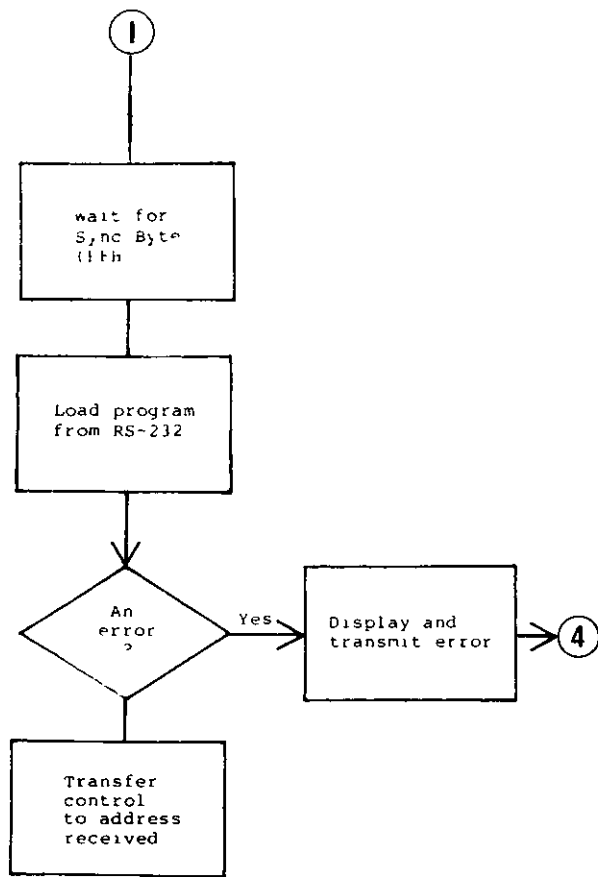
<Break>	Pressing this key is simply recorded by setting location 405BH non-zero. It is up to an operating system to use this flag if desired.
<Enter>	Terminates the Keyboard routine. Any other keys pressed up to that time will be acted upon. <Enter> is useful for experienced users who do not want to wait until the keyboard timer expires.

The Control section now takes over and follows the following flowchart.









Notes:

- (1) If the boot sector was not 256 bytes in length then it is assumed to be a Model III package and the ROM image will be needed. If the sector is 256 bytes in length then the sector is scanned for the sequence CDxx00H. The CD is the first byte of a Z80 unconditional subroutine call. The next byte can have any value. The third byte is tested against a zero. What this check does is test for any references to the first 256 bytes of memory. All Radio Shack Model III operating systems and many other packages all reference the ROM at some point during the boot sector. Most boot sectors will display a message if the system cannot be loaded. To save space these routines use the Model III ROM calls to display the message. Several ROM calls have their entry points in the first 256 bytes of memory and these references are detected by the boot ROM.

Packages that do not reference the Model III ROM in the boot sector can still cause the Model III ROM image to be loaded by coding a CDxx00 somewhere in the boot sector. It does not have to be executable. At the same time Model 4 packages must take care that there is no sequence of bytes in the boot sector that could be mis-interpreted to be a reference to the Boot ROM. An example of this would be sequence 06CD0E00 which is a LD B 0CDH and a LD C 0. If the boot sector cannot be changed then the user must press the F3 key each time the system is started to inform the ROM that the disk contains a Model III package which needs the Model III ROM image.

- (2) If you are loading a Model 4 operating system then the boot ROM will always transfer control to the first byte of the boot sector, which is at 4300H. If you are loading a Model III operating system or about to use Model III ROM BASIC then the transfer address is 3015H. This is the address of a jump vector in the C ROM of the Model III ROM image and this will cause the system to behave exactly like a Model III. If the ROM image file that is loaded has a different transfer address then that address will be used when loading is complete. If the image is already present, the Boot ROM will use 3015H.
- (3) Two different tests are done to insure that the Model III ROM image is present. The first test is to check every third location starting at 3000H for a C3H. This is done for 10 locations. If any of these locations does not contain a C3H then the ROM image is considered to be not present. The next test is to check two bytes at location 000BH. If these addresses contain E9E1H then the ROM image is considered to be present.
- (4) See Disk Director Searcher for more information.
- (5) See File Loader for more information.
- (6) The RS-232 loader is described under RS-232 Boot.

Disk Directory Searcher

When the Model III ROM image is to be loaded it is always read from the floppy in drive 0.

Before the operation begins, some checks are made. First the boot sector is read in from the floppy and the first byte is checked to make sure it is either a 00H or a FEH. If the byte contains some other value no attempt will be made to read the ROM image from that disk. The location of the directory cylinder is then taken from the boot sector and the type of disk is determined. This is done by examining the Data Address Mark that

was picked up by the Floppy Disk Controller (FDC) during the read of the sector. If the DAM equals 1 the disk is a TRSDOS 1 x style disk. If the DAM equals 0 then the disk is a LDOS 5.1 TRSDOS 6 style disk. This is important since TRSDOS 1 x disks number sectors starting with 1 and LDOS style disks number sectors starting with 0.

Once the disk type has been determined, an extra test is made if the disk is a LDOS style disk. This test reads the Granule Address Location Table (GAT) to determine if the disk is single sided or double sided.

The directory is then read one record at a time and a compare is made against the pattern MODEL% for the filename and III for the extension. The % means that any character will match this position. If the user pressed one of the selection keys (A-G) during the keyboard scan, then that character is substituted in place of the % character. For example, if you pressed D, then the search would be for the file MODEL D with the extension III. The searching algorithm searches until it finds the entry or it reaches the end of the directory.

Once the entry has been found, the extent information for that file is copied into a control block for later use.

File Loader

The file loader is actually two modules — the actual loader and a set of routines to fetch bytes from the file on disk. The loader is invoked via a RST 28H. The byte fetcher is called by the loader using RST 20H. Since restart vectors can be re-directed, the same loader is used by the RS 232 boot. The difference is that the RST 20H is redirected to point to the RS 232 data receiving routine. The loader reads standard loader records and acts upon two types:

- 01 Data Load
 - 1 byte with length of block including address
 - 1 word with address to load the data
 - n bytes of data where $n + 2$ equals the length specified
- 02 Transfer Address
 - 1 byte with the value of 02
 - 1 word with the address to start execution at

Any other loader code is treated as a comment block and is ignored. Once an 02 record has been found, the loader stops reading, even if there is additional data, so be sure to place the 02 record at the end of the file.

Floppy and Hard Disk Driver

The disk drivers are entered via RST 8H and will read a sector anywhere on a floppy disk and anywhere on head 1 (top head) in a hard disk drive. Either 256 or 512 byte sectors are readable by these routines and they make the determination of the sector size. The hard disk driver is compatible with both the WD1000 and the WD1010 controllers. The floppy disk driver is written for the WD1793 controller.

Serial Loader

Invoking the serial loader is similar to forcing a boot from hard disk or floppy. In this case the right shift key must be pressed at some time during the first three seconds after reset. The program does not care if the key is pressed forever, making it convenient to connect pins 8 and 10 of the keyboard connector with a shorting plug for bench testing of boards. This assumes that the object program being loaded does not care about the key closure.

Upon entry, the program first asserts DTR (J4 pin 20) and RTS (J4 pin 4) true. Next, Not Ready is printed on the topmost line of the video display. Modem status line CD (J4 pin 8) is then sampled. The program loops until it finds CD asserted true. At that time the message Ready is displayed. Then the program sets about determining the baud rate from the host computer.

To determine the baud rate, the program compares data received by the UART to a test byte equal to 55 hex. The receiver is first set to 19200 baud. If ten bytes are received which are not equal to the test byte, the baud rate is reduced. This sequence is repeated until a valid test byte is received. If ten failures occur at 50 baud, the entire process begins again at 19200 baud. If a valid test byte is received, the program waits for ten more to arrive before concluding that it has determined the correct baud rate. If at this time an improper byte is received or a receiver error (overrun, framing, or parity) is intercepted, the task begins again at 19200 baud.

In order to get to this point, the host or the modem must assert CD true. The host must transmit a sequence of test bytes equal to 55 hex with 8 data bits, odd parity, and 1 or 2 stop bits. The test bytes should be separated by approximately 0.1 second to avoid overrun errors.

When the program has determined the baud rate, the message

Found Baud Rate x

is displayed on the screen, where x is a letter from A to P meaning

A = 50 baud	E = 150	I = 1800	M = 4800
B = 75	F = 300	J = 2000	N = 7200
C = 110	G = 600	K = 2400	O = 9600
D = 134.5	H = 1200	L = 3600	P = 19200

The same message less the character signifying the baud rate is transmitted to the host with the same baud rate and protocol. This message is the signal to the host to stop transmitting test bytes.

After the program has transmitted the baud rate message, it reads from the UART data register in order to clear any overrun error that may have occurred due to the test bytes coming in during the transmission of the message. This is because the receiver must be made ready to receive a sync byte signalling the beginning of the command file. For this reason, it is important that the host wait until the entire baud rate message (16 characters) is received before transmitting the sync byte, which is equal to FF hex.

When the loader receives the sync byte, the message

Loading

is displayed on the screen. Again, the same message is transmitted to the host, and again, the host must wait for the entire transmission before starting into the command file.

If the receiver should intercept a receive error while waiting for the sync byte, the entire operation up to this point is aborted. The video display is cleared and the message

Error x

is displayed near the bottom of the screen, where x is a letter from B to H, meaning:

- B = parity error
- C = framing error
- D = parity & framing errors
- E = overrun error
- F = parity & overrun errors
- G = framing & overrun errors
- H = parity & framing & overrun errors

The message

Error

is then transmitted to the host. The entire process is then repeated from the Not Ready message. A six second delay is inserted before reinitialization. This is longer than the time required to transmit five bytes at 50 baud, so there is no need to be extra careful here.

If the sync byte is received without error, then the Loading message is transmitted and the program is ready to receive the command file. After receiving the Loading message, the host can transmit the file without nulls or delays between bytes.

(Since the file represents Z80 machine code and all 256 combinations are meaningful, it would be disastrous to transmit nulls or other ASCII control codes as fillers, acknowledgement or start/stop bytes. The only control codes needed are the standard command file control bytes.)

Data can be transmitted to the loader at 19200 baud with no delays inserted. Two stop bits are recommended at high baud rates.

See the File Loader description for more information on file loading.

If a receive error should occur during file loading, the abort procedure described above will take place, so when attempting remote control, it is wise to monitor the host receiver during transmission of the file. When the host is near the object board, as is the case in the factory application, or when more than one board is being loaded, it may be advantageous or even necessary to ignore the transmitted responses of the object board(s) and to manually pace the test byte, sync byte, and command file phases of the transmission process using the video display for handshaking.

System Programmers Information

The Model 4P Boot ROM uses two areas of RAM while it is running. These are 4000H to 40FFH and 4300H to 43FFH. (For 512 byte boot sectors, the second area is 4300H to 44FFH.) If the Model III ROM Image is loaded, additional areas are used. See the technical reference manual for the system you are using for a list of these areas.

Operating systems that want to support a software restart by re-executing the contents of the boot ROM can accomplish this in one of two ways. If the operating system relies on the Model III ROM Image, then jump to location 0 as you have in the past. If the operating system is a Model 4 mode package, a simple way is to code the following instructions in your assembly and load them before you want to reset.

Absolute Location	Instruction	
0000	DI	
0001	LD	A 1
0003	OUT	(9CH) A

These instructions cause the boot ROM to become addressable. After executing the OUT instruction, the next instruction executed will be one in the boot ROM. (These instructions also exist in the Model III ROM image at location 0.) The boot ROM has been written so that the first instruction is at address 0005. The hardware must be in memory mode 0 or 1, or else the boot ROM will not be switched in. This operation can be done with an OUT instruction and then a RST 0 can be executed to have the ROM switched in.

Restarts can be redirected at any time while the ROM is switched in. All restarts jump to fixed locations in RAM and these areas may be changed to point to the routine that is to be executed.

Restart	RAM Location	Default Use
0	none	Cold Start Boot
8	4000H	Disk I/O Request
10	4003H	Display string
18	4006H	Display block
20	4009H	Byte Fetch (Called by Loader)
28	400CH	File Loader
30	400FH	Keyboard scanner
38	4012H	Reserved for future use
66	4015H	NMI (Floppy I/O Command Complete)

The above routines have fixed entry parameters. These are described here.

Disk I/O Request (RST 8H)

Accepts

A	1 for floppy, 2 for hard disk
B	Command
	Initialize 1
	Restore 4
	Seek 6
	Read 12 (All reads have an implied seek)
C	Sector number to read
	The contents of the location disktype (405CH) are added to this value before an actual read. If the disk is a two sided floppy, just add 18 to the sector number.
DE	Cylinder number (Only E is used in floppy operations)
HL	Address where data from a read operation is to be stored

Returns

Z	Success, Operation Completed
NZ	Error, Error code in A

Error Codes

3	Hard Disk drive is not ready
4	Floppy disk drive is not ready
5	Hard Disk drive is not available
6	Floppy disk drive is not available
7	Drive Not Ready and no Index (Disk in drive, door open)
8	CRC Error
9	Seek Error
11	Lost Data
12	ID Not Found

Display String (RST 10H)

Accepts

HL	Pointer to text to be displayed
	Text must be terminated with a null (0)
DE	Offset position on screen where text is to be displayed
	(A 0000H will be the upper left-hand corner of the display)

Returns

Success Always

A	Altered
DE	Points to next position on video
HL	Points to the null (0)

Display Block (RST 18H)

Accepts

HL	Points to control vector in the format
	+0 Screen Offset
	+2 Pointer to text, terminated with null
	+4 Pointer to text, terminated with null
	..
	+n word FFFFH End of control vector
or	+n word FFEFH Next word is new Screen Offset

If Z flag is set on entry then the first screen offset is read from DE instead of from the control vector.

Each string is positioned after the previous string, unless a FFEFH entry is found. This is used heavily in the ROM to reduce duplication of words in error messages.

Returns

Success Always

DE	Points to next position on video
-----------	----------------------------------

Byte Fetch (RST 20H)

Accepts None

Returns

Z	Success, byte in A
NZ	Failure, error code in A

Errors

2	Any errors from the disk I/O call and ROM image can't be loaded — Too many extents
10	ROM image can't be loaded — Disk drive is not ready

File Loader (RST 28H)

Accepts None

Returns	
Z	Success
NZ	Failure, error code in A
Errors	
	Any errors from the disk I/O call or the byte fetch call and:
0	The ROM image was not found on drive 0

There are several pieces of information left in memory by the boot ROM which are useful to system programmers. These are shown below:

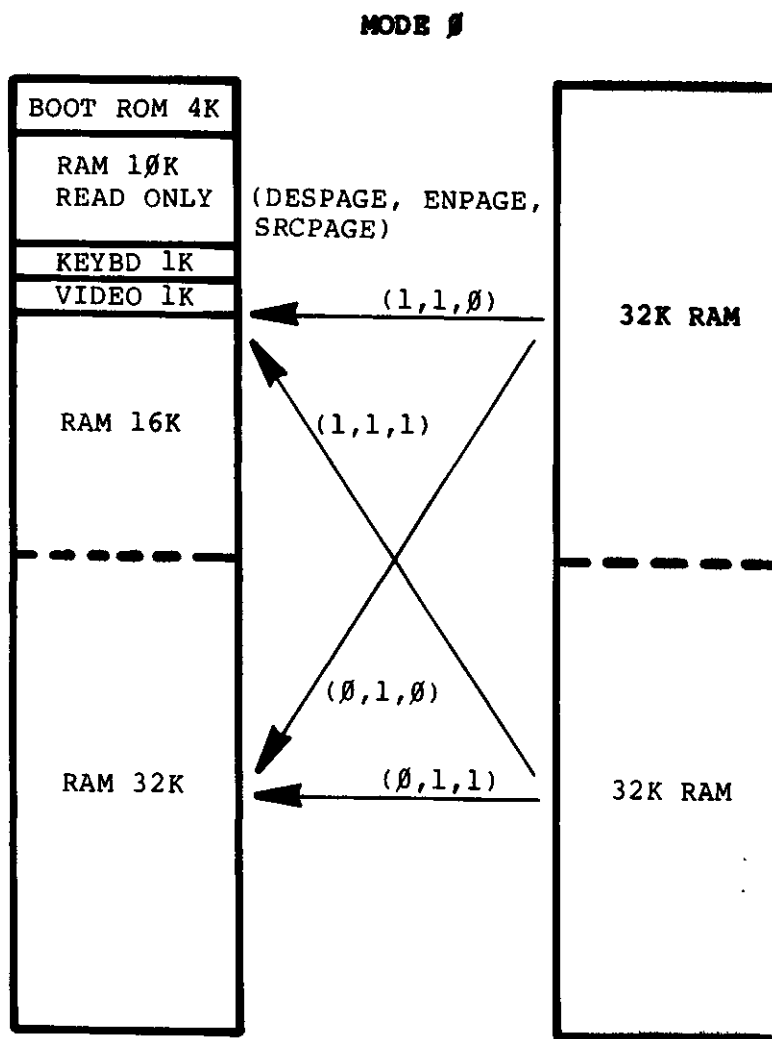
RAM Location	Description
401DH	ROM Image Selected (% for none selected or A-G)
4055H	Boot type 1 = Floppy 2 = Hard disk 3 = ARCNET 4 = RS-232C 5 - 7 = Reserved
4056H	Boot Sector Size (1 for 256, 2 for 512)
4057H	RS-232 Baud Rate (only valid on RS-232 boot)
4059H	Function Key Selected 0 = No function key selected <F1> or <1> 86 <F2> or <2> 87 <F3> or <3> 88 <Caps> 85 <Ctrl> 84 <Left-Shift> 82 <Right-Shift> 83 Reserved 80-81 and 89-90
405BH	Break Key Indication (non-zero if <Break> pressed)
405CH	Disk type (0 for LDOS/ TRSDOS 6,1 for TRSDOS 1.x)

Keep in mind that Model III ROM image will initialize these areas, so this information is useful only to the Model 4 mode programmer.

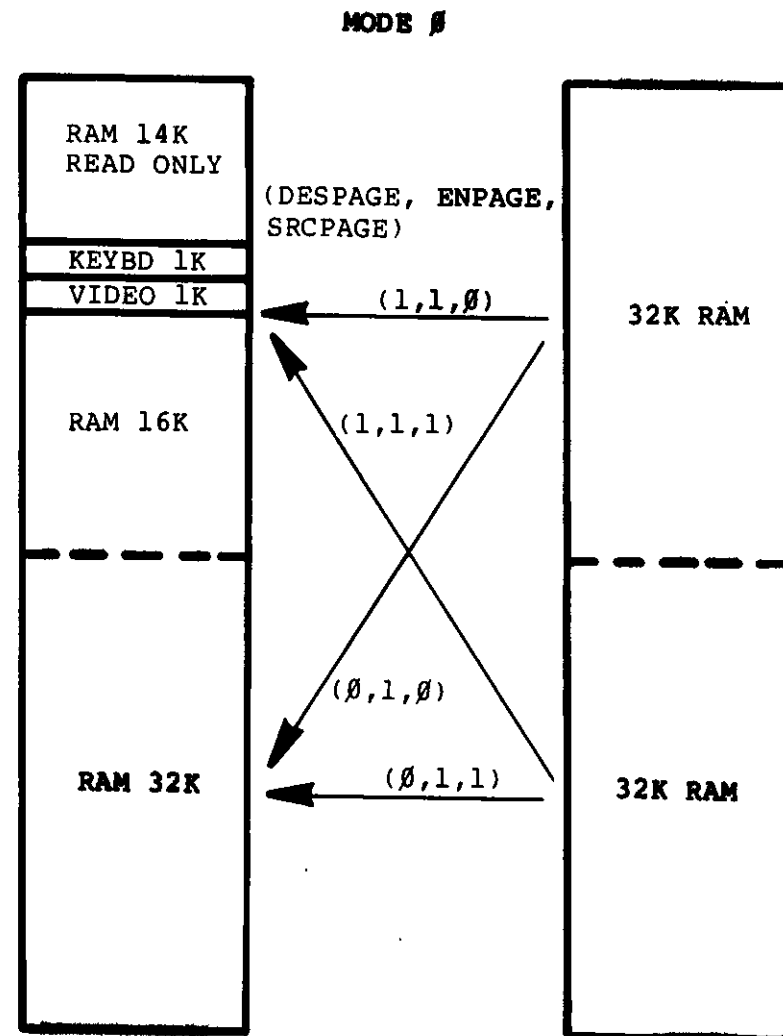
4.2.7 RAM

Two configurations of Random Access Memory (RAM) are available on the Model 4P: 64K and 128K. The 64K and 128K option use the 6665-type 64K x 1 200NS Dynamic RAM, which requires only a single +5v supply voltage.

The DRAMs require multiplexed incoming address lines. This is accomplished by ICs U110 and U111 which are 74LS157 multiplexers. Data to and from the DRAMs are buffered by a 74LS245 (U118) which is controlled by Gate Array 4.2 (U106). The proper timing signals RAS0*, RAS1*, MUX*, and CAS* are generated by a delay line circuit U94. U116 (1 2 of a 74S112) and U117 (1 4 of a 74F08) are used to generate a precharge circuit. During M1 cycles of the Z80A in 4 MHz mode, the high time in MREQ has a minimum time of 110 nanosecs. The specification of 6665 DRAM requires a minimum of 120 nanosecs so this circuit will shorten the MREQ signal during the M1 cycle. The resulting signal PMREQ is used to start a RAM memory cycle through U114 (a 74S64). Each different cycle is controlled at U114 to maintain a fast M1 cycle so no wait states are required. The output of U114 (PRAS*) is ANDed with RFSH to not allow MUX* and CAS* to be generated during a REFRESH cycle. PRAS* also generates either RAS0* or RAS1*, depending on which bank of RAM the CPU is selecting. GCAS* generated by the delay line U94 is latched by U116 (1 2 of a 74S112) and held to the end of the memory cycle. The output of U116 is ANDed with VIDEO signal to disable the CAS* signal from occurring if the cycle is a video memory access. Refer to M1 Cycle Timing (Figure 4-7 and 4-8), Memory Read and Memory Write Cycle Timing (Figure 4-9) and (Figure 4-10).

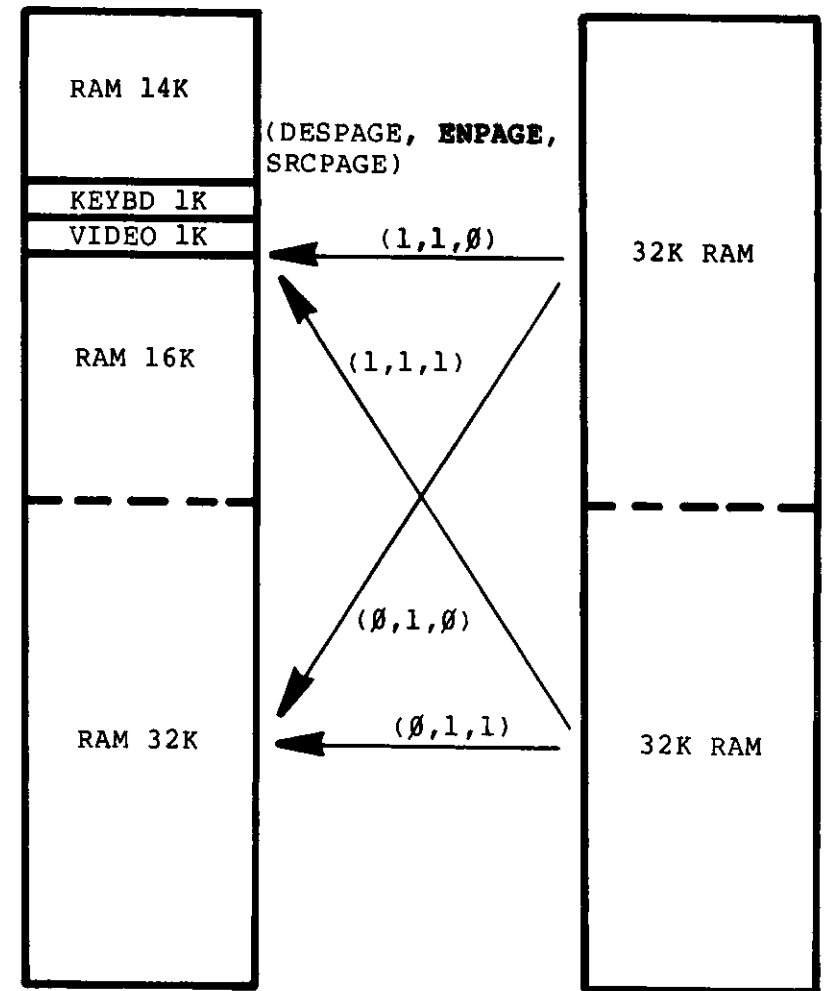
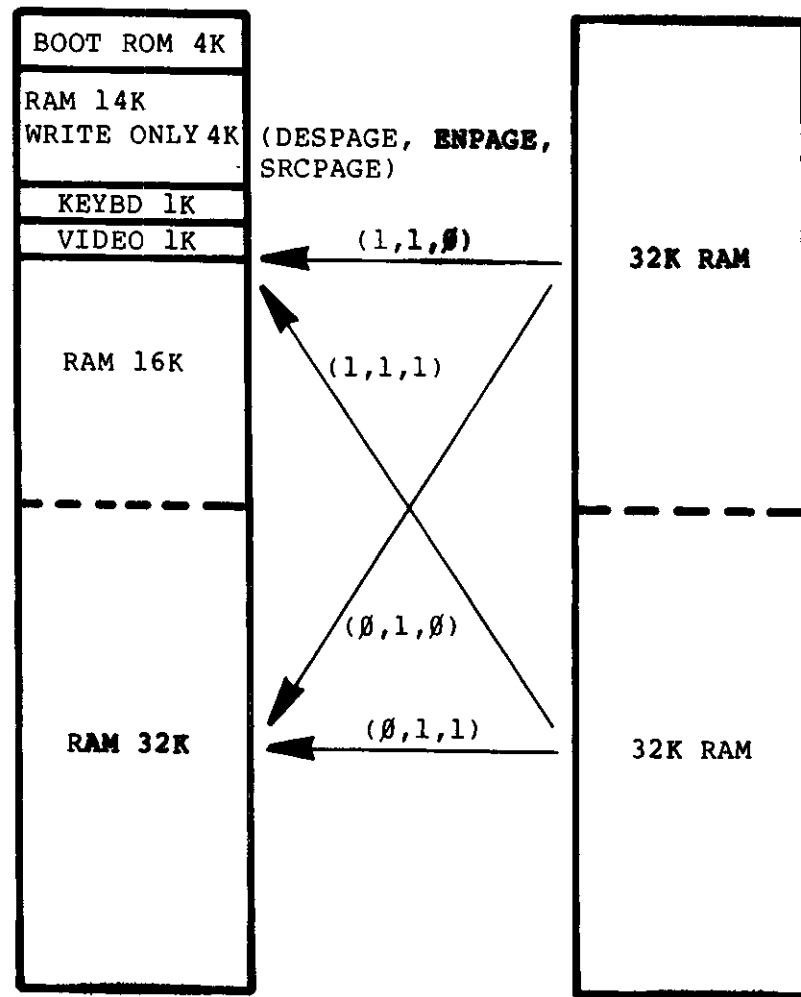


	STATE	LEVEL
SEL0 =	0	0V
SEL1 =	0	0V
ROM =	1	0V



	STATE	LEVEL
SEL0 =	0	0V
SEL1 =	0	0V
ROM =	0	5V

Figure 4-4. Memory

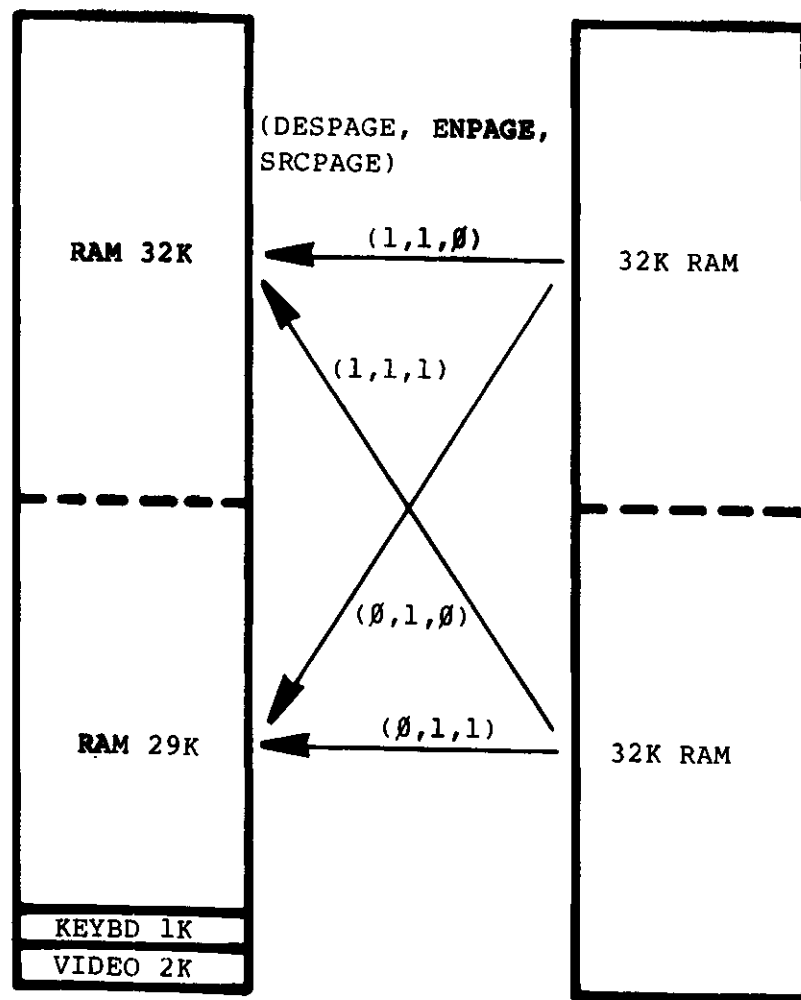


	STATE	LEVEL
SEL0 =	1	5V
SEL1 =	0	0V
ROM =	0	5V

	STATE	LEVEL
SEL0 =	1	5V
SEL1 =	0	0V
ROM =	1	0V

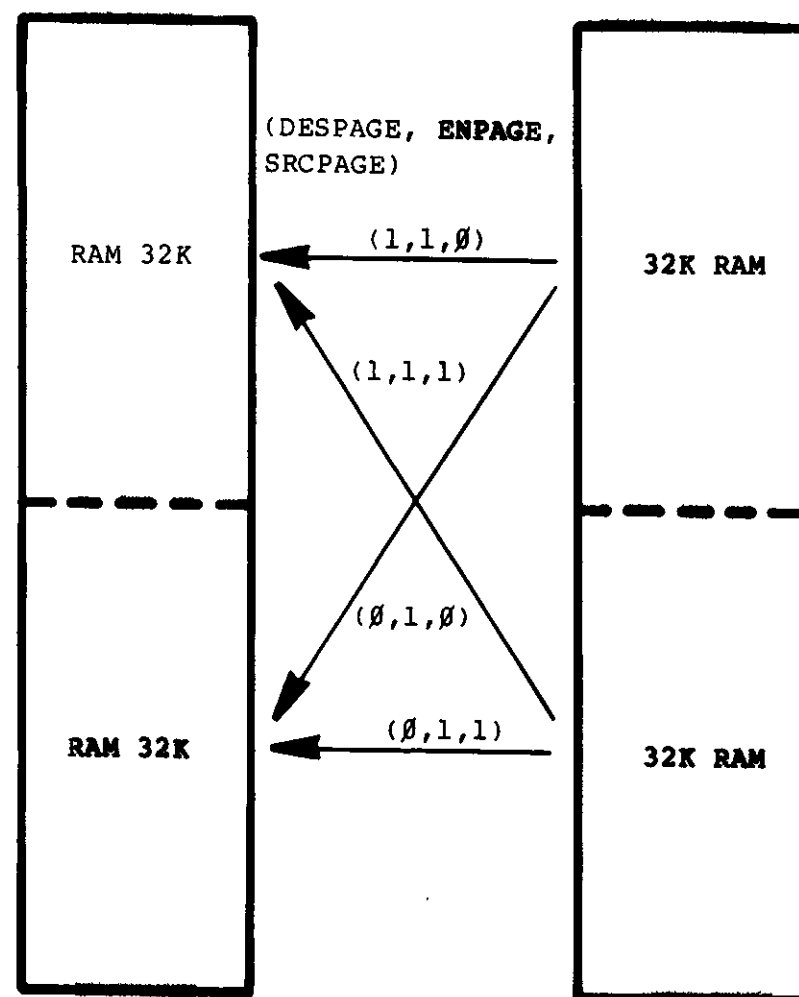
Figure 4-5. Memory

MODE 2



	STATE	LEVEL
SEL0 =	0	0V
SEL1 =	1	5V
ROM =	X	

MODE 3



	STATE	LEVEL
SEL0 =	1	5V
SEL1 =	1	5V
ROM =	X	

Figure 4-6. Memory

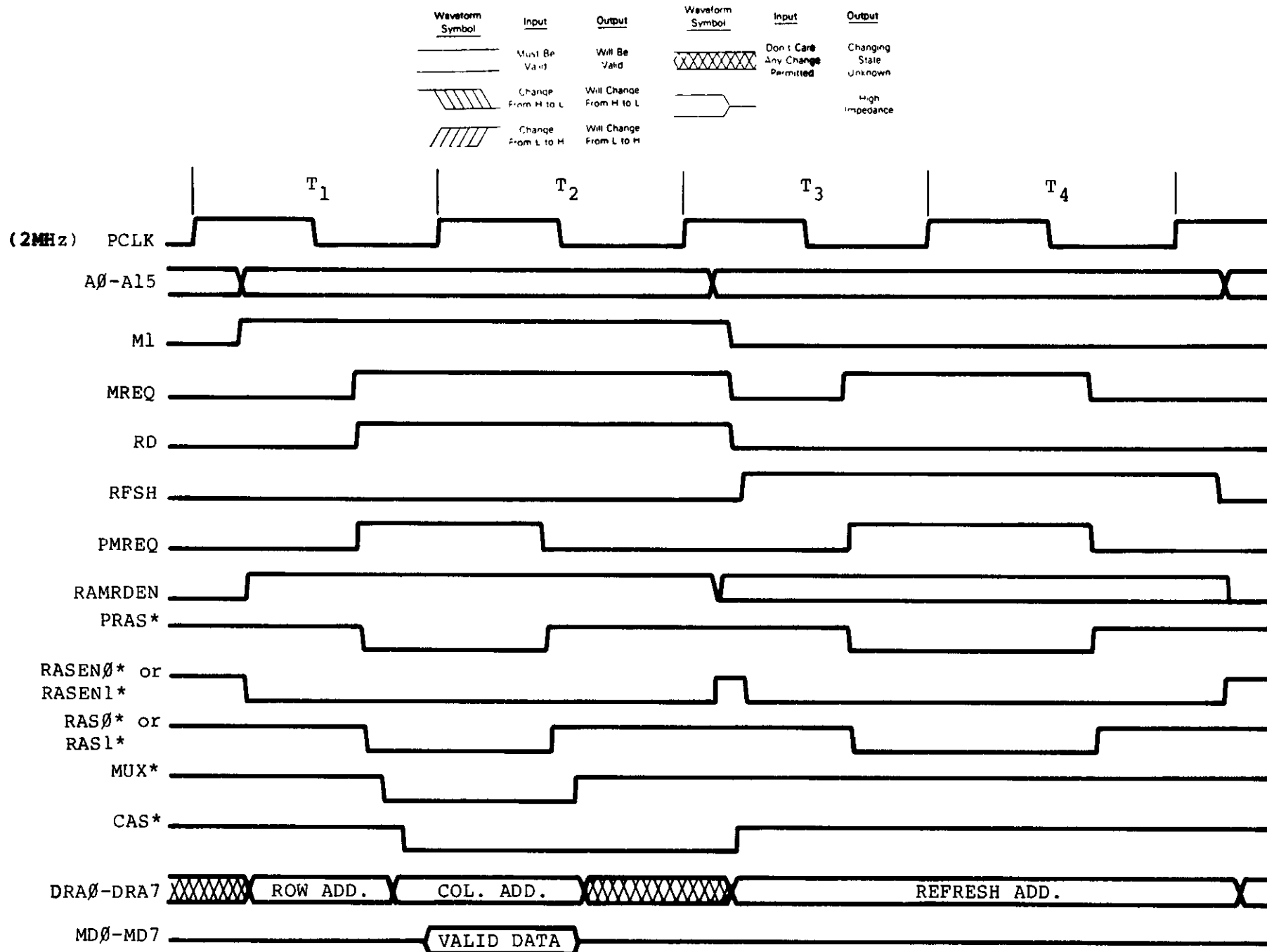


Figure 4-7. M₁ Cycle Timing (2MHZ)

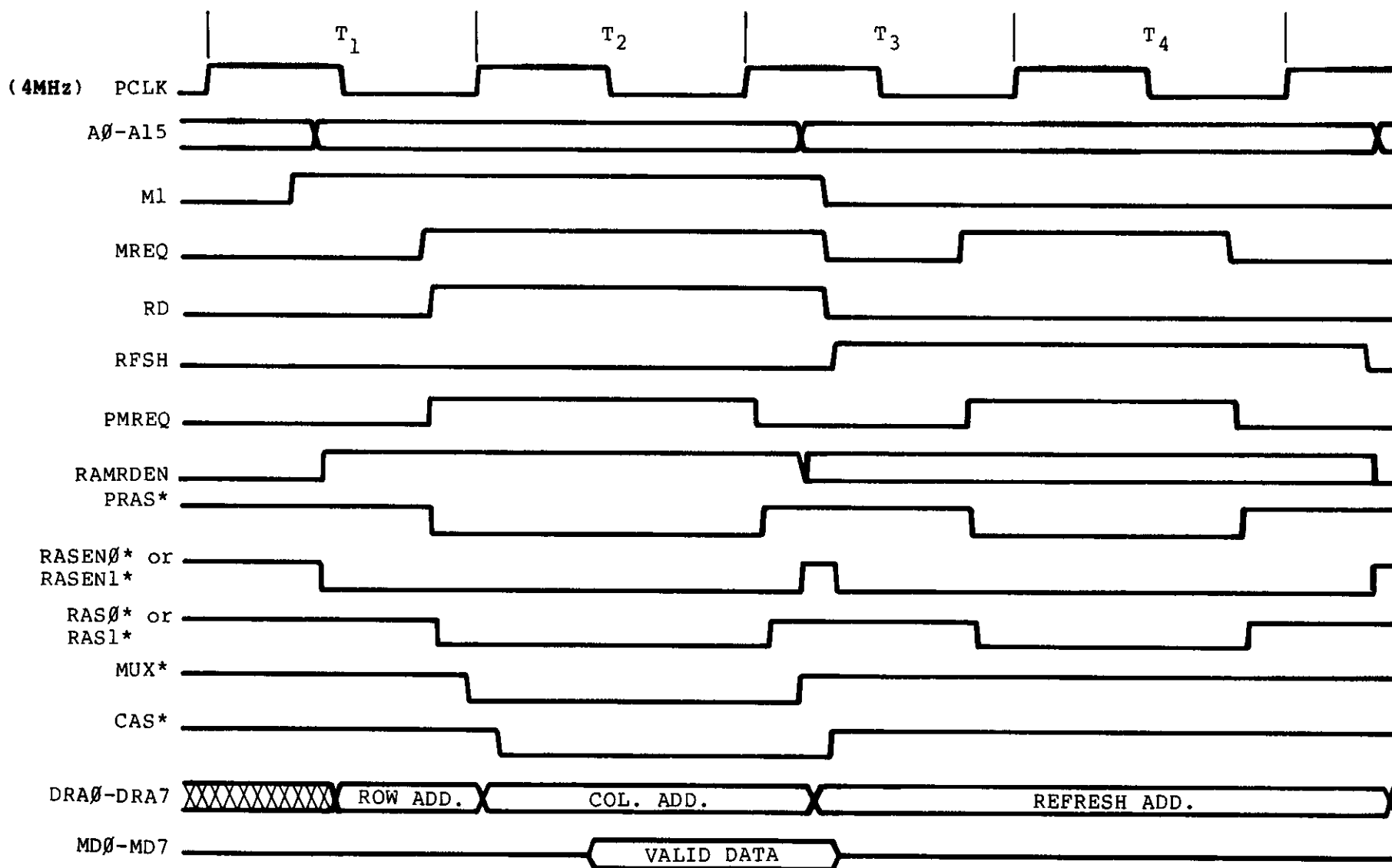
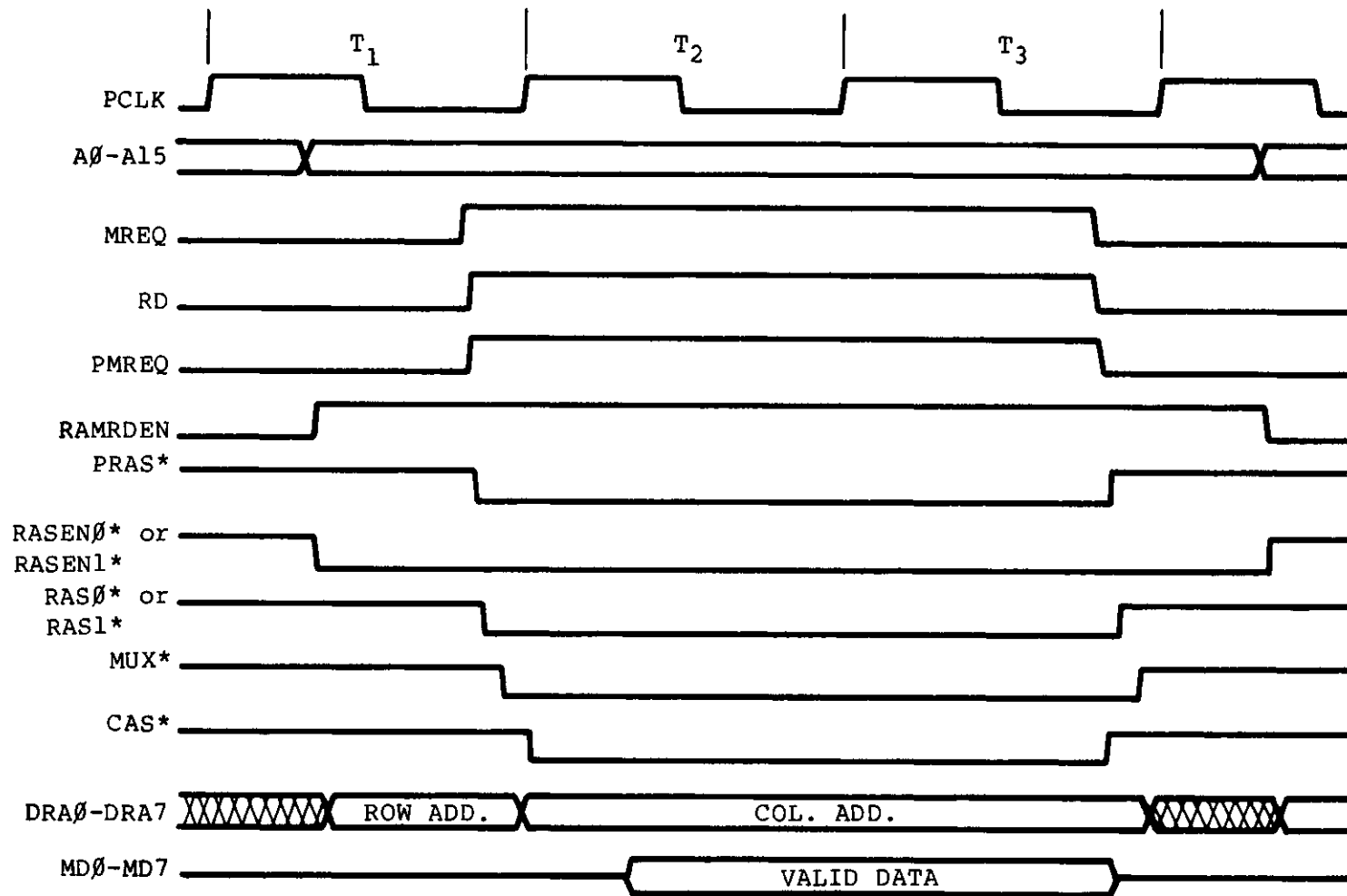


Figure 4-8. M1 Cycle Timing (4MHZ)

**Figure 4-9. Memory Read Cycle Timing**

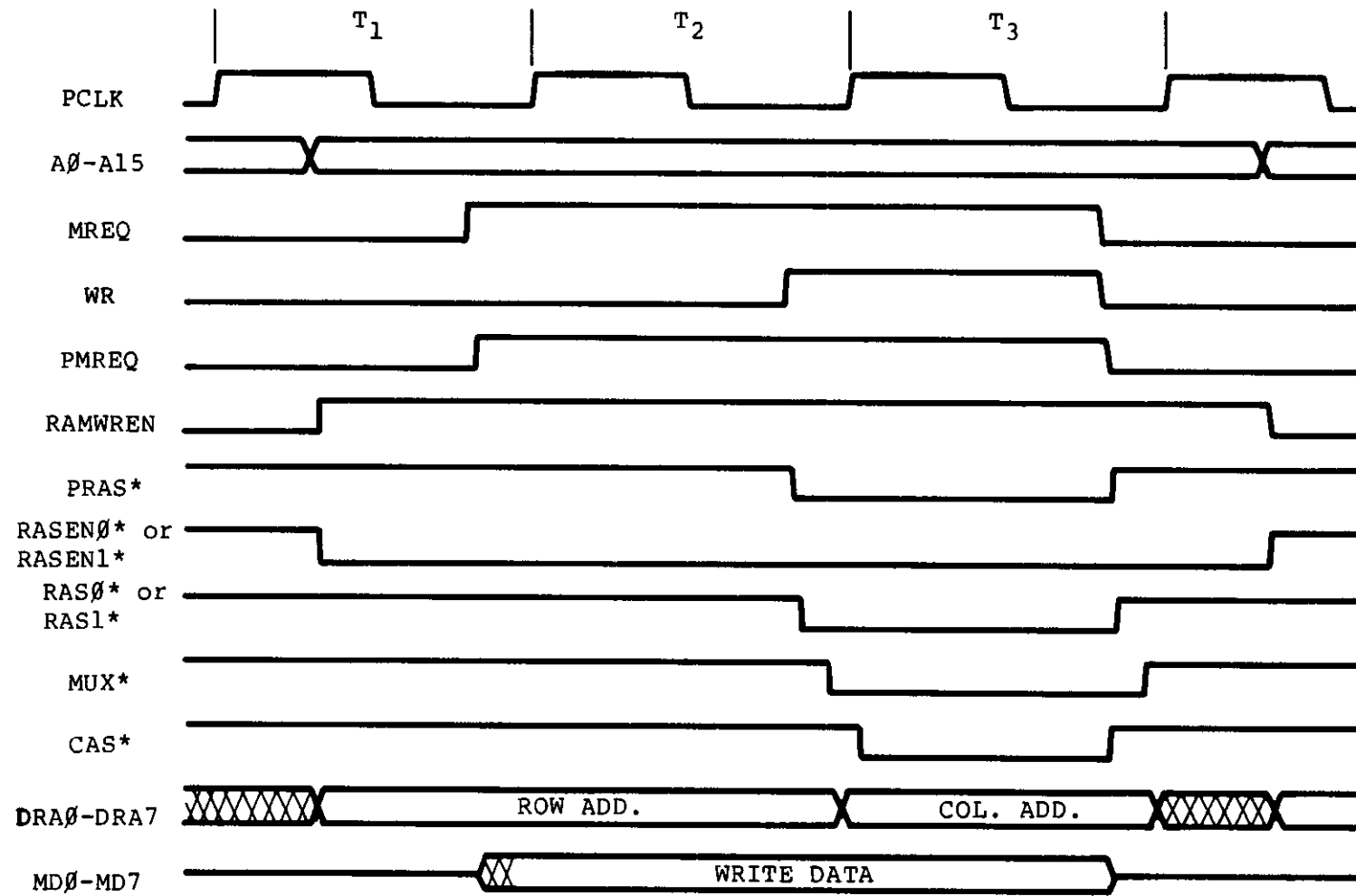


Figure 4-10. Memory Write Cycle Timing

Memory Map — Model 4P

Mode 0 SEL0 - 0 - 0V
SEL1 0 0V
ROM - 1 - 0V

0000 — 0FFF	Boot ROM	4K
1000 — 37FF	RAM (Read Only)	10K
37E8 — 37E9	Printer Status (Read Only)	2
3800 — 3BFF	Keyboard	1K
3C00 — 3FFF	Video	1K
4000 — FFFF	RAM	48K

Mode 0 SEL0 - 0 = 0V
SEL1 - 0 - 0V
ROM - 0 - +5V

0000 — 37FF	RAM (Read Only)	14K
37E8 — 37E9	Printer Status (Read Only)	2
3800 — 3BFF	Keyboard	1K
3C00 — 3FFF	Video	1K
4000 — FFFF	RAM	48K

Mode 1 SEL0 = 1 = +5V
SEL1 = 0 = 0V
ROM = 1 = 0V

0000 — 0FFF	Boot ROM	4K
0000 — 0FFF	RAM (Write Only)	4K
1000 — 37FF	RAM	10K
3800 — 3BFF	Keyboard	1K
3C00 — 3FFF	Video	1K
4000 — FFFF	RAM	48K

Mode 1 SEL0 - 1 - +5V
SEL1 - 0 0V
ROM - 0 - +5V

0000 — 37FF	RAM	14K
3800 — 3BFF	Keyboard	1K
3C00 — 3FFF	Video	1K
4000 — FFFF	RAM	48K

Mode 2 SEL0 - 0 - 0V
SEL1 = 1 = +5V
ROM - X - Don't Care

0000 — F3FF	RAM	61K
F400 — F7FF	Keyboard	1K
F800 — FFFF	Video	2K

Mode 3 SEL0 - 1 - +5V
SEL1 = 1 = +5V
ROM = X = Don't Care

0000 — FFFF	RAM	64K
-------------	-----	-----

I/O Port Assignment

Port #	Normally Used	Out	In
FC — FF	FF	CASSOUT *	MODIN *
F8 — FB	F8	LPOUT *	LPIN *
F4 — F7	F4	DRVSEL *	(RESERVED)
F0 — F3	-	DISKOUT *	DISKIN *
F0	F0	FDC COMMAND REG.	FDC STATUS REG.
F1	F1	FDC TRACK REG.	FDC TRACK REG.
F2	F2	FDC SECTOR REG.	FDC SECTOR REG.
F3	F3	FDC DATA REG.	FDC DATA REG.
EC — EF	EC	MODOUT *	RTCIN *
E8 — EB	-	RS232OUT *	RS232IN *
E8	E8	UART MASTER RESET	MODEM STATUS
E9	E9	BAUD RATE GEN. REG.	(RESERVED)
EA	EA	UART CONTROL AND MODEM CONTROL REG.	UART STATUS REG.
EB	EB	UART TRANSMIT HOLDING REG.	UART HOLDING REG. (RESET D.R.)
E4 — E7	E4	WR NMI MASK REG. *	WD NMI STATUS *
E0 — E3	E0	WR INT MASK REG. *	RD INT MASK REG. *
A0 — DF	-	(RESERVED)	(RESERVED)
9C — 9F	9C	BOOT *	(RESERVED)
94 — 9B	-	(RESERVED)	(RESERVED)
90 — 93	90	SEN *	(RESERVED)
8C — 8F	-	GSEL0 *	GSEL0 *
88 — 8B	-	CRTCCS *	(RESERVED)
88, 8A	88	CRCT ADD. REG.	(RESERVED)
89, 8B	89	CRCT DATA REG.	(RESERVED)
84 — 87	84	OPREG *	(RESERVED)
80 — 83	-	GSEL1 *	GSEL1 *

I/O Port Description

Name: CASSOUT *
Port Address: FC — FF
Access: WRITE ONLY
Description: Output data to cassette or for sound generation

Note: The Model 4P **does not** support cassette storage this port is only used to generate sound that was to be output via cassette port The Model 4P sends data to onboard sound circuit

D0 = Cassette output level (sound data output)

D1 = Reserved

D2 — D7 = Undefined

Name: MODIN * (CASSIN *)
Port Address: FC — FF
Access: READ ONLY
Description: Configuration Status

D0 = 0

D1 = CASSMOTORON STATUS

D2 = MODSEL STATUS

D3 = ENALTSET STATUS

D4 = ENEXTIO STATUS

D5 = (NOT USED)

D6 = FAST STATUS

D7 = 0

Name: LPOUT *
Port Address: F8 — FB
Access: WRITE ONLY
Description: Output data to line printer

D0 — D7 = ASCII BYTE TO BE PRINTED

Name: LPIN *
Port Address: F8 — FB
Access: READ ONLY
Description: Input line printer status

D0 — D3 = (RESERVED)

D4 = FAULT
1 = TRUE
0 = FALSE

D5 = UNIT SELECT
1 = TRUE
0 = FALSE

D6 = OUTPAPER
1 = TRUE
0 = FALSE

D7 = BUSY
1 = TRUE
0 = FALSE

Name: DRVSEL *
Port Address: F4 — F7
Access: WRITE ONLY
Description: Output FDC Configuration

Note: Output to this port will **ALWAYS** cause a 1-2 msc (Microsecond) wait to the Z80

D0 = DRIVE SELECT 0

D1 = DRIVE SELECT 1

D2 = (RESERVED)

D3 = (RESERVED)

D4 = SDSEL
0 = SIDE 0
1 = SIDE 1

D5 = PRECOMPEN
0 = No write precompensation
1 = Write Precompensation enabled

D6 = WSGEN
0 = No wait state generated
1 = wait state generated

Note: This wait state is to sync Z80 with FDC chip during FDC operation

D7 = DDEN *
0 = Single Density enabled (FM)
1 = Double Density enabled (MFM)

Name: DISKOUT *
Port Address: F0 — F3
Access: WRITE ONLY
Description: Output to FDC Control Registers

Port F0 = FDC Command Register

Port F1 = FDC Track Register

Port F2 = FDC Sector Register

Port F3 = FDC Data Register

(Refer to FDC Manual for Bit Assignments)

Name: DISKIN *
Port Address: F0 — F3
Access: READ ONLY
Description: Input FDC Control Registers

Port F0 = FDC Status Register

Port F1 = FDC Track Register

Port F2 = FDC Sector Register

Port F3 = FDC Data Register

(Refer to FDC Manual for Bit Assignment)

Name: MODOUT *
Port Address: EC — EF
Access: WRITE ONLY
Description: Output to Configuration Latch

D0 = (RESERVED)

D1 = CASSMOTORON (Sound enable)
 0 = Cassette Motor Off (Sound enabled)
 1 = Cassette Motor On (Sound disabled)

D2 = MODSEL
 0 = 64 or 80 character mode
 1 = 32 or 40 character mode

D3 = ENALTSET
 0 = Alternate character set disabled
 1 = Alternate character set enabled

D4 = ENEXTIO
 0 = External IO Bus disabled
 1 = External IO Bus enabled

D5 = (RESERVED)

D6 = FAST
 0 = 2 MHZ Mode
 1 = 4 MHZ Mode

D7 = (RESERVED)

Name: RTCIN *
Port Address: EC — EF
Access: READ ONLY
Description: Clear Real Time Clock Interrupt

D0 — D7 = DON'T CARE

Name: RS232OUT *
Port Address: E8 — EB
Access: WRITE ONLY
Description: UART Control, Data Control, Modem Control
 BRG Control

Port E8 = UART Master Reset

Port E9 = BAUD Rate Gen Register

Port EA = UART Control Register (Modem Control Reg)

Port EB = UART Transmit Holding Reg

(Refer to Model III or 4 Manual for Bit Assignments)

Name: RS232IN *
Port Address: E8 — EB
Access: READ ONLY
Description: Input UART and Modem Status

Port E8 = MODEM STATUS

Port E9 = (RESERVED)

Port EA = UART Status Register

Port EB = UART Receive Holding Register (Resets DR)

(Refer to Model III or 4 Manual for Bit Assignments)

Name: WRNMIMASKREG *
Port Address: E4 — E7
Access: WRITE ONLY
Description: Output NMI Latch

D0 — D5 (RESERVED)

D6 = ENMOTOROFFINT
 0 = Disables Motoroff NMI
 1 = Enables Motoroff NMI

D7 = ENINTRQ
 0 = Disables INTRQ NMI
 1 = Enables INTRQ NMI

Name: RDNMISTATUS *
Port Address: E4 — E7
Access: READ ONLY
Description: Input NMI Status

D0 = 0

D2 — D4 (RESERVED)

D5 = RESET (not needed)
 0 = Reset Asserted (Problem)
 1 = Reset Negated

D6 = MOTOROFF
 0 = Motoroff Asserted
 1 = Motoroff Negated

D7 = INTRQ
 0 = INTRQ Asserted
 1 = INTRQ Negated

Name: WRINTMASKREG *
Port Address: E0 — E3
Access: WRITE ONLY
Description: Output INT Latch

D0 — D1 = (RESERVED)

D2 = ENRTC
 0 = Real time clock interrupt disabled
 1 = Real time clock interrupt enabled

D3 = ENIOBUSINT
 0 = External IO Bus interrupt disabled
 1 = External IO Bus interrupt enabled

D4 = ENXMITINT
 0 = RS232 Xmit Holding Reg empty int disabled
 1 = RS232 Xmit Holding Reg empty int enabled

D5 = ENRECINT
 0 = RS232 Rec Data Reg full int disabled
 1 = RS232 Rec Data Reg full int enabled

D6 = ENERRORINT
 0 = RS232 UART Error interrupts disabled
 1 = RS232 UART Error interrupts enabled

D7 = (RESERVED)

Name: RDINTSTATUS *
Port Address: E0 — E3
Access: READ ONLY
Description: Input INT Status

D0 — D1 = (RESERVED)

D2 = RTC INT

D3 = IOBUS INT

D4 = RS232 XMIT INT

D5 = RS232 REC INT

D6 = RS232 UART ERROR INT

D7 = (RESERVED)

Name: BOOT *
Port Address: 9C — 9F
Access: WRITE ONLY
Description: Enable or Disable Boot ROM

D0 = ROM *
 0 = Boot ROM Disabled
 1 = Boot ROM Enabled

D1 — D7 = (RESERVED)

Name: SEN *
Port Address: 90 — 93
Access: WRITE ONLY
Description: Sound output

D0 = SOUND DATA

D1 — D7 = (RESERVED)

Name: OPREG *
Port Address: 84
Access: WRITE ONLY
Description: Output to operation reg.

D0 = SEL0

D1 = SEL1

SEL1	SEL0	MODE
0	0	0
0	1	1
1	0	2
1	1	3

D2 = 8064
0 = 64 character mode
1 = 80 character mode

D3 = INVERSE
0 = Inverse video disabled
1 = Inverse video enabled

D4 = SRCPAGE — Points to the page to be mapped
as new page
0 = U64K, L32K Page
1 = U64K, U32K Page

D5 = ENPAGE — Enables mapping of new page
0 = Page mapping disabled
1 = Page mapping enabled

D6 = DESPAGE — Points to the page where new
page is to be mapped:
0 = L64K, U32K Page
1 = L64K, L32K Page

D7 = PAGE
0 = Page 0 of Video Memory
1 = Page 1 of Video Memory

4.2.8 Video Circuit

The heart of the video display circuit in the Model 4P is the 68045 Cathode Ray Tube Controller (CRTC) U42. The CRTC is a preprogrammed video controller that provides two screen formats 64 by 16 and 80 by 24. The format is controlled by pin 3 of the CRTC (8064*). The CRTC generates all of the necessary signals required for the video display. These signals are VSYNC (Vertical Sync), HSYNC (Horizontal Sync) for proper sync of the monitor. DISPEN (Display Enable) which indicates when video data should be output to the monitor, the refresh memory addresses (MA0-MA13) which addresses the video RAM, and the row addresses (RA0-RA4) which indicates which scan line row is being displayed. The CRTC also provides hardware scrolling by writing to the internal Memory Start Address Register by OUTing to Port 88H. The internal cursor control of the 68045 is not used in the Model 4P video circuit.

Since the 80 by 24 screen requires 1,920 screen memory locations, a 2K by 8 static RAM (U82) is used for the video RAM. Addressing to the video RAM (U82) is provided by the 68045 when refreshing the screen and by the CPU when updating of the data is performed. These two sets of address lines are multiplexed by three 74LS157s (U41, U61, and U81). The multiplexers are switched by CRTCLK which allows the CRTC to address the video RAM during the high state of CRTCLK and the CPU access during the low state. A10 from the CPU is controlled by PAGE* which allows two display pages in the 64 by 16 format. When updates to the video RAM are performed by the CPU, the CPU is held in a WAIT state until the CRTC is not addressing the video RAM. This operation allows reads and writes to video RAM without causing hashing on the screen. The circuit that performs this function is a 74LS244 buffer (U84), an 8 bit transparent latch, 74LS373 (U83) and a Delay line circuit shared with Dynamic RAM timing circuit consisting of a 74LS74 (U98), 74LS32 (U96), 74LS04 (U95), 74LS00 (U92), 74LS02 (U69), and Delay Line (U94). During a CPU Read Access to the Video RAM, the address is decoded by the GA 4 2 and asserts VIDEO* low. This is inverted by U95 (1 6 of 74LS04) which pulls one input of U92 (1 4 of 74LS00) and in turn asserts VWAIT* low to the CPU. RD is high at this time and is latched into U98 (1 2 of 74LS74) on the rising edge of XADR7*, inverse of CRTCLK.

When RD is latched by U98 the Q output goes low releasing WAIT* from the CPU. The same signal also is sent to the Delay Line (U94) through U117 (1 4 of 74F08). The Delay line delays the falling edge 240 ns for VLATCH* which latches the read data from the video RAM at U83. The data is latched so the CRTC can refresh the next address location and prevent any hashing. MRD* decoded by U106 and a memory read is ORed with VIDEO* which enables the data from U83 to the data bus. The CPU then reads the data and completes the cycle. A CPU write is slightly more complex in operation. As in the RD cycle, VIDEO* is asserted low which asserts VWAIT* low to the CPU. WR is high at this time which is Nanded with VIDEO and synced with CRTCLK to create VRAMDIS that disables the video RAM output. On the rising edge of XADR7*, WR is latched into U98 (1 2 of 74LS74) which releases VWAIT* and starts cycle through the Delay Line. After 30ns DLYVWR* (Delayed video write) is asserted low which also asserts VBUFEN* (Video Buffer Enable) low. VBUFEN* enabled data from the Data bus to the video RAM. Approximately 120ns later DLYVWR* is negated high which writes the data to the video RAM and negates VBUFEN* turning off buffer. The CPU then completes WR cycle to the video RAM. Refer to Video RAM CPU Access Timing Figure 5-12 for timing of above RD or WR cycles.

During screen refresh, CRTCLK is high allowing the CRTC to address Video RAM. The data out of the video RAM is latched by LOAD* into Gate Array 4 3 (U102). INVERSE* determines if character should be alpha-numeric only (INVERSE* high) or unchanged (INVERSE* low). A9 is decoded with ENALTSET (Enable Alternate Set) and 7, which controls the alternate set in the character generator ROM. See ENALTSET Control Table below.

ENALTSET	Q7	Q6	A9
0	0	0	0
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	0
1	1	1	0

4.2.8 Video Circuit

The heart of the video display circuit in the Model 4P is the 68045 Cathode Ray Tube Controller (CRTC) U42. The CRTC is a preprogrammed video controller that provides two screen formats 64 by 16 and 80 by 24. The format is controlled by pin 3 of the CRTC (8064*). The CRTC generates all of the necessary signals required for the video display. These signals are VSYNC (Vertical Sync), HSYNC (Horizontal Sync) for proper sync of the monitor. DISPEN (Display Enable) which indicates when video data should be output to the monitor, the refresh memory addresses (MA0-MA13) which addresses the video RAM, and the row addresses (RA0-RA4) which indicates which scan line row is being displayed. The CRTC also provides hardware scrolling by writing to the internal Memory Start Address Register by OUTing to Port 88H. The internal cursor control of the 68045 is not used in the Model 4P video circuit.

Since the 80 by 24 screen requires 1,920 screen memory locations, a 2K by 8 static RAM (U82) is used for the video RAM. Addressing to the video RAM (U82) is provided by the 68045 when refreshing the screen and by the CPU when updating of the data is performed. These two sets of address lines are multiplexed by three 74LS157s (U41, U61, and U81). The multiplexers are switched by CRTCLK which allows the CRTC to address the video RAM during the high state of CRTCLK and the CPU access during the low state. A10 from the CPU is controlled by PAGE* which allows two display pages in the 64 by 16 format. When updates to the video RAM are performed by the CPU, the CPU is held in a WAIT state until the CRTC is not addressing the video RAM. This operation allows reads and writes to video RAM without causing hashing on the screen. The circuit that performs this function is a 74LS244 buffer (U84), an 8 bit transparent latch, 74LS373 (U83) and a Delay line circuit shared with Dynamic RAM timing circuit consisting of a 74LS74 (U98), 74LS32 (U96), 74LS04 (U95), 74LS00 (U92), 74LS02 (U69), and Delay Line (U94). During a CPU Read Access to the Video RAM, the address is decoded by the GA 4 2 and asserts VIDEO* low. This is inverted by U95 (1 6 of 74LS04) which pulls one input of U92 (1 4 of 74LS00) and in turn asserts VWAIT* low to the CPU. RD is high at this time and is latched into U98 (1 2 of 74LS74) on the rising edge of XADR7*, inverse of CRTCLK.

When RD is latched by U98 the Q output goes low releasing WAIT* from the CPU. The same signal also is sent to the Delay Line (U94) through U117 (1 4 of 74F08). The Delay line delays the falling edge 240 ns for VLATCH* which latches the read data from the video RAM at U83. The data is latched so the CRTC can refresh the next address location and prevent any hashing. MRD* decoded by U106 and a memory read is ORed with VIDEO* which enables the data from U83 to the data bus. The CPU then reads the data and completes the cycle. A CPU write is slightly more complex in operation. As in the RD cycle, VIDEO* is asserted low which asserts VWAIT* low to the CPU. WR is high at this time which is Nanded with VIDEO and synced with CRTCLK to create VRAMDIS that disables the video RAM output. On the rising edge of XADR7*, WR is latched into U98 (1 2 of 74LS74) which releases VWAIT* and starts cycle through the Delay Line. After 30ns DLYVWR* (Delayed video write) is asserted low which also asserts VBUFEN* (Video Buffer Enable) low. VBUFEN* enabled data from the Data bus to the video RAM. Approximately 120ns later DLYVWR* is negated high which writes the data to the video RAM and negates VBUFEN* turning off buffer. The CPU then completes WR cycle to the video RAM. Refer to Video RAM CPU Access Timing Figure 5-12 for timing of above RD or WR cycles.

During screen refresh, CRTCLK is high allowing the CRTC to address Video RAM. The data out of the video RAM is latched by LOAD* into Gate Array 4 3 (U102). INVERSE* determines if character should be alpha-numeric only (INVERSE* high) or unchanged (INVERSE* low). A9 is decoded with ENALTSET (Enable Alternate Set) and 7, which controls the alternate set in the character generator ROM. See ENALTSET Control Table below.

ENALTSET	Q7	Q6	A9
0	0	0	0
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	0
1	1	1	0

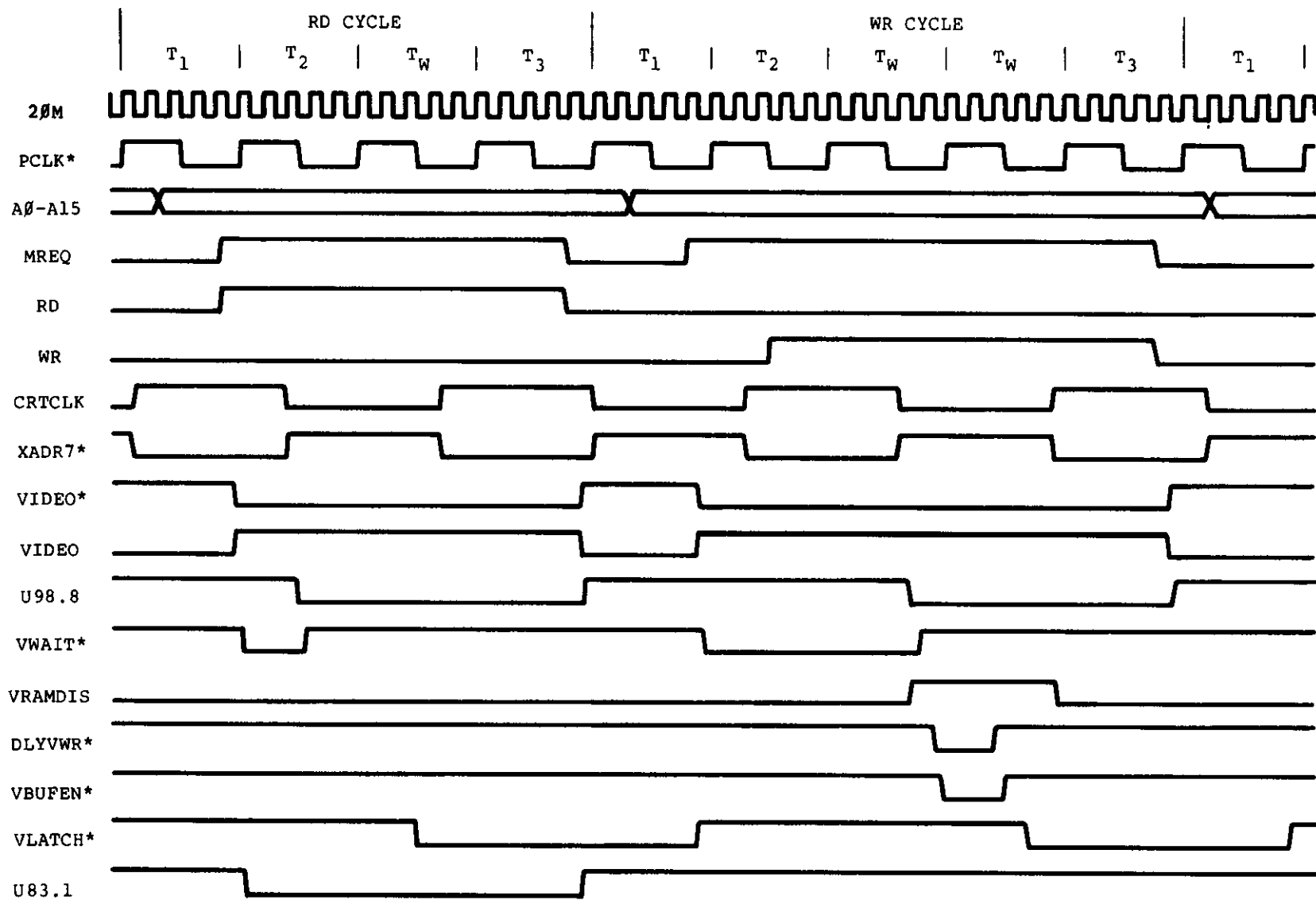


Figure 4-11. Video RAM CPU Access Timing

RA0-RA3, row addresses from the CRTC are used to control which scan line is being displayed. The Model 4P has a 4-bit full adder 74LS283 (U101) to modify the Row address. During a character display DLYGRAPHIC* is high which applies a high to all 4 bits to be added to row address. This will result in subtracting one from Row address count and allow all characters to be displayed one scan line lower. The purpose is so inverse characters will appear within the inverse block. When a graphic block is displayed DLYGRAPHIC* is low which causes the row address to be unmodified. Moving jumper from E14-E15 to E15-E16 will disable this circuit.

DLYCHAR* and DLYGRAPHICS are inverse signals and control which data is to be loaded into the internal shift register of U102. When DLYCHAR* is low and DLYGRAPHIC* is high, the Character Generator ROM (U103) is enabled to output data. When DLYCHAR* is high and DLYGRAPHIC* is low the graphics characters are internally buffered to the shift register. The data is loaded into the internal shift register on the rising edge of SHIFT* when LOADS* is low. Serial video data is output U102 19. The video information is inverted by U142 and F83, is filtered by R14 (47 ohm resistor), and C227 (100 pf Cap) and output to video monitor. VSYNC and HSYNC are buffered by (1/2 of 74LS86) U143 and are also output to video monitor. Refer to Video Circuit Timing Figure 4-12 and Inverse Video Timing Figure 4-13 for timing relationships of Video Circuit.

4.2.9 Keyboard

The keyboard interface of the Model 4P consists of open collector drivers which drive an 8 by 8 key matrix keyboard and an inverting buffer which buffers the key or keys pressed on the data bus. The open collector drivers (U57 and U77 (7416) are driven by address lines A0-A7 which drive the column lines of the keyboard matrix. The ROW lines of the keyboard are pulled up by a 1.5 kohm resistor pack RP2. The ROW lines are buffered and inverted onto the data bus by U78 (74LS240) which is enabled when KEYBD* is a logic low. KEYBD* is a memory mapped decode of addresses 3800-3BFF in Model III Mode and F400-F7FF in Model 4/4P mode. Refer to the Memory Map under Address Decode for more information. During real time operation, the CPU will scan the keyboard periodically to check if any keys are pressed. If no key is pressed, the resistor pack RP2 keeps the inputs of U78 at a logic high. U78 inverts the data to a logic low and buffers it to the data bus which is read by the CPU. If a key is pressed when the CPU scans the correct column line, the key pressed will pull the corresponding row to a logic low. U78 inverts the signal to a logic high which is read by the CPU.

4.2.10 Real Time Clock

The Real Time Clock circuit in the Model 4P provides a 30 Hz (in the 2 MHz CPU mode) or 60 Hz (in the 4 MHz CPU mode) interrupt to the CPU. By counting the number of interrupts that have occurred, the CPU can keep track of the time. The 60 Hz vertical sync signal (VSYNC) from the video circuitry is used for the Real Time Clock's reference. In the 2 MHz mode, FAST is a logic low which sets the Preset input (pin 4 of U23 (74LS74)) to a logic high. This allows the 60 Hz (VSYNC) to be divided by 2 to 30 Hz. The output of 1/2 of U23 is ORed with the original 60 Hz and then clocks another 74LS74 (1/2 of U23). If the real time clock is enabled (ENRTC at a logic high), the interrupt is latched and pulls the INT* line low to the CPU. When the CPU recognizes the interrupt, the pulse is counted and the latch reset by pulling RTCIN* low. In the 4 MHz mode, FAST is a logic high which keeps the first half of U23 in a preset state (the Q* output at a logic low). The 60 Hz is used to clock the interrupts.

NOTE: If interrupts are disabled, the accuracy of the real time clock will suffer.

4.2.11 Line Printer Port

The Line Printer Port Interface consists of a pulse generator, an eight-bit latch, and a status line buffer. The status of the line printer is read by the CPU by enabling buffer U3 (74LS244). This buffer is enabled by LPRD* which is a memory map and port map decode. In Model III mode, only the status can be read from memory location 37E8 or 37E9. The status can be read in all modes by an input from ports F8-FB. For a listing of the bit status, refer to Port Map section.

After the printer driver software determines that the printer is ready for printing (by reading the correct status), the characters to be printed are output to Port F8-FB. U2, a 74LS374 eight-bit latch, latches the character byte and outputs to the line printer. One-half of U1 (74LS123), a one-shot, is then triggered which generates an appropriate strobe signal to the printer which signifies a valid character is ready. The output of the one-shot is buffered by 1/6th of the U51 (74LS04) to prevent noise from the printer cable from false-triggering the one-shot.

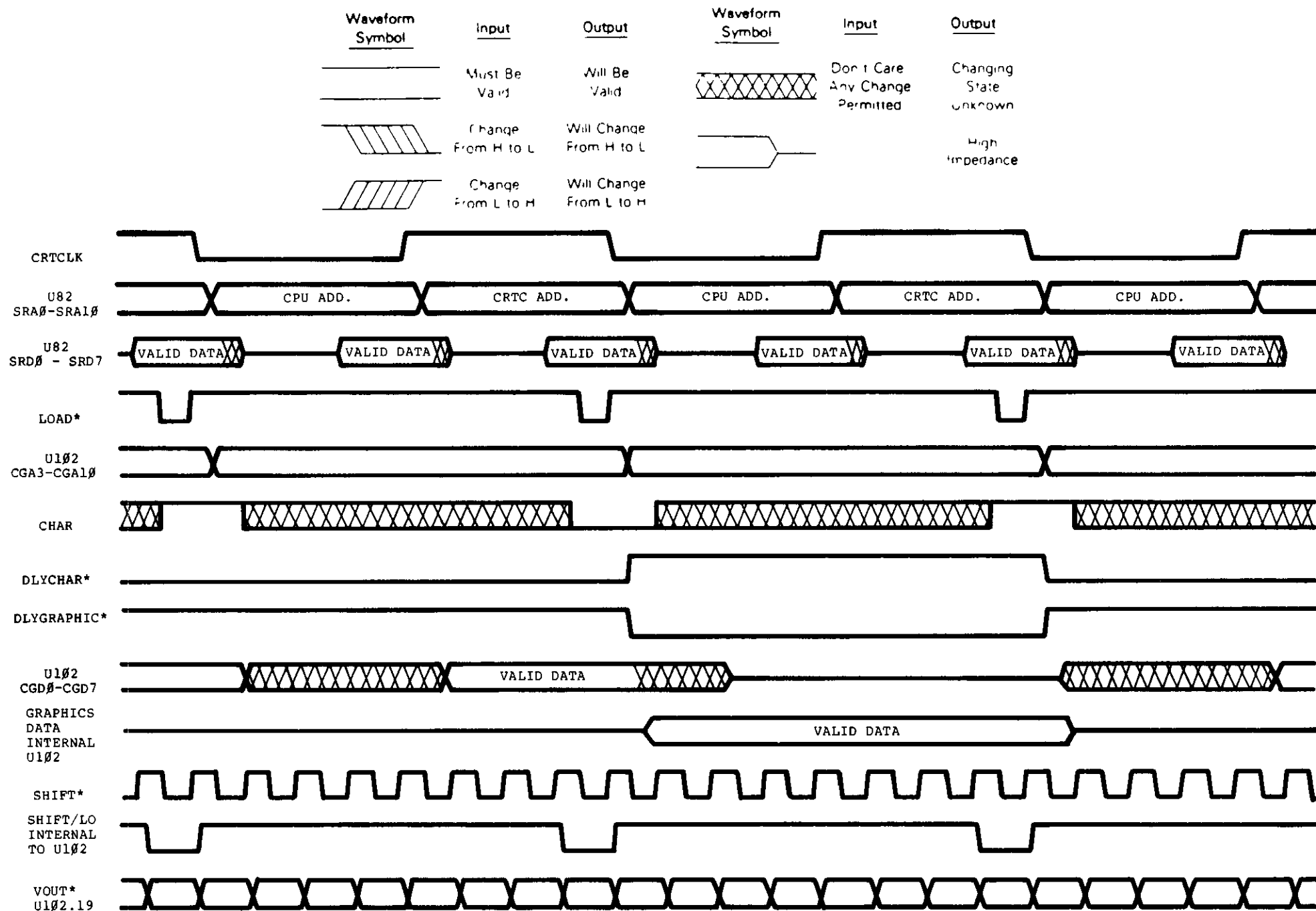


Figure 4-12. Video Circuit Timing

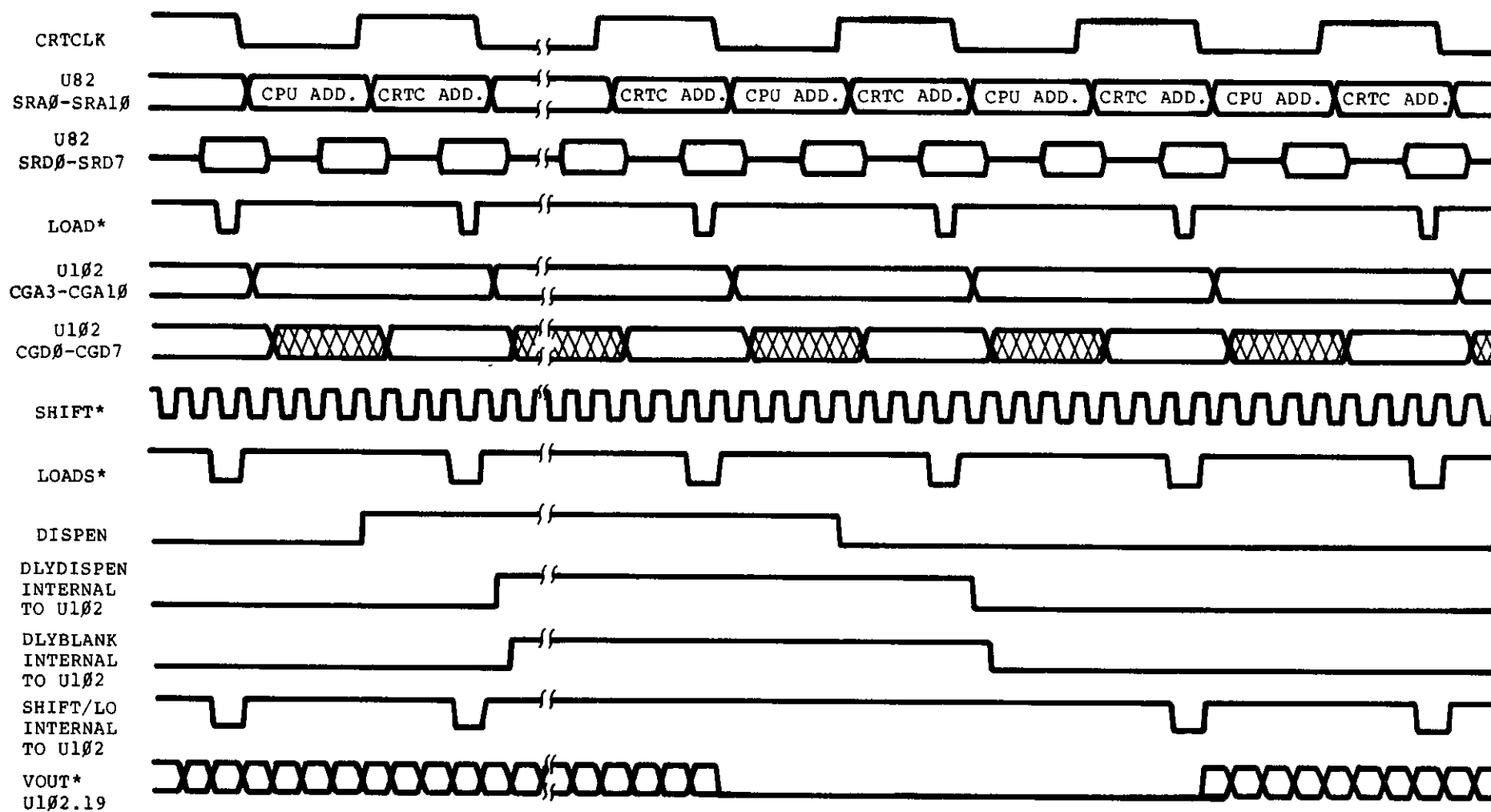


Figure 4-13. Video Blanking Timing

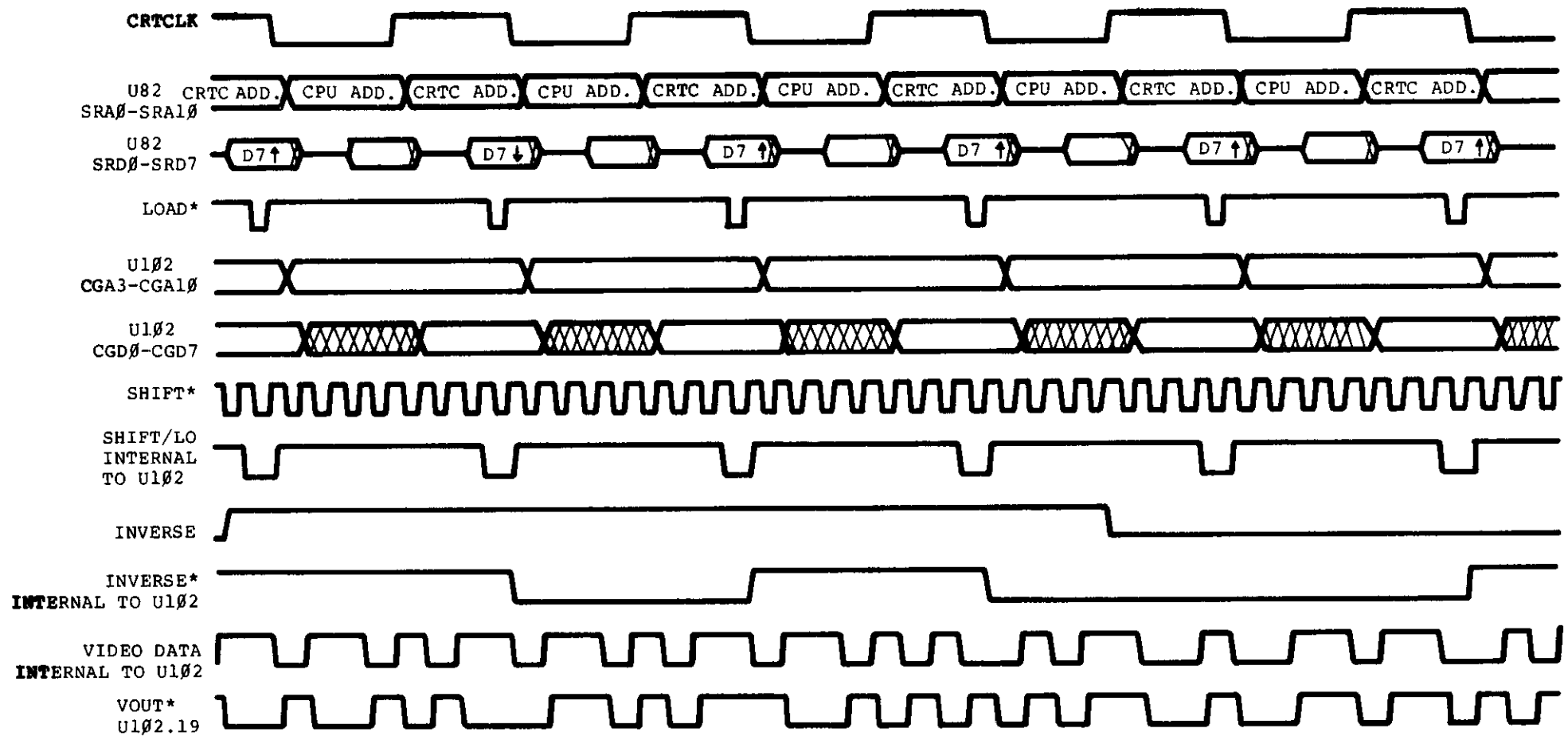


Figure 4-14. Inverse Video Timing

4.2.12 Graphics Port

The Graphics Port (J7) on the Model 4P is provided to attach the optional Graphics Board. The port provides D0-D7 (Data Lines), A0-A3 (Address Lines), IN*, GEN*, and RESET* for the necessary interface signals for the Graphics Board. GEN* is generated by negative ORing Port selects GSEL0* (8C-8FH) and GSEL1* (80-83H) together by (1 4 of 74LS08) U4. The resulting signal is negative ANDed with IORQ* by (1 4 of 74S32) U24. Seven timing signals are provided to allow synchronization of Main Logic Board Video and Graphics Board Video. These timing signals are VSYNC, HSYNC, DISPEN, DCLK, H, I, and J. Three control signals from the Graphics Board are used to sync to CPU access and select different video modes. WAIT* controls the CPU access by causing the CPU to WAIT till video is in retrace area before allowing any writes or reads to Graphics Board RAM. ENGRAF is asserted when Graphics video is displayed. ENGRAF also disables inverse video mode on Main Logic Board Video. CL166* (Clear 74L166) is used to enable or disable mixing of Main Logic Board Video and Graphics Board Video. If CL166* is negated high, then mixing is allowed in all four video modes 80 x 24, 40 x 24, 64 x 16, and 32 x 16. If CL166* is asserted low, this will clear the video shift register U63, which allows no video from the Main Logic Board. In this state 8064* is automatically asserted low to put screen in 80 x 24 video mode. Refer to Figure 4-15 Graphic Board Video Timing for timing relationships. Refer to the Model 4/4P Graphics Board Service information for service or technical information on the Graphics Board.

4.2.13 Sound

The sound circuit in the Model 4P is compatible with the Sound Board which was optional in the Model 4. Sound is generated by alternately setting and clearing data bit D0 during an OUT to port 90H. The state of D0 is latched by U129 (1 2 of a 74LS74) and the output is amplified by Q2 which drives a 8Ω speaker. The speed of the software loop determines the frequency, and thus, the pitch of the resulting tone. Since the Model 4P does not have a cassette circuit, some existing software that used the cassette output for sound would have been lost. The Model 4P routes the cassette latch to the sound board through U109. When the CASSMOTORON signal is a logic low, the cassette motor is off, then the cassette output is sent to the sound circuit.

4.2.14 I/O Bus Port

The Model 4P Bus is designed to allow easy and convenient interfacing of I/O devices to the Model 4P. The I/O Bus supports all the signals necessary to implement a device compatible with the Z80s I/O structure.

Addresses

A0 to A7 allow selection of up to 256* input and 256 output devices if external I/O is enabled.

*Ports 80H to 0FFH are reserved for System use.

Data

DB0 to DB7 allow transfer of 8-bit data onto the processor data bus if external I/O is enabled.

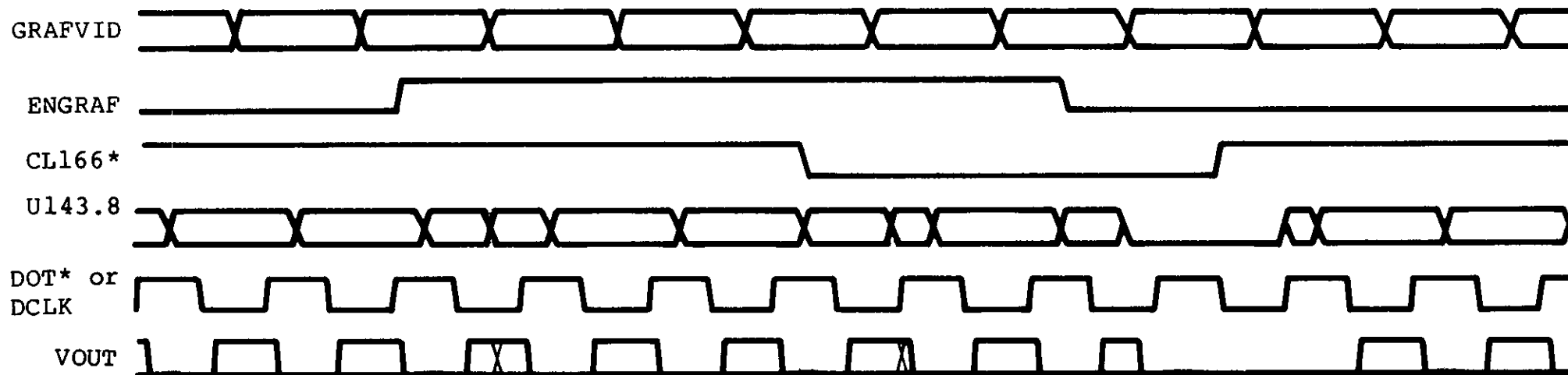
Control Lines

- 1 M1* — Z80A signal specifying an M1 or Operation Code Fetch Cycle or with IOREQ* it specifies an Interrupt acknowledge.
- 2 IN* — Z80A signal specifying that an input is in progress. Logic AND of IOREQ* and WR*.
- 3 OUT* — Z80A signal specifying that an output is in progress. Logic AND of IOREQ* and WR*.
- 4 IOREQ* — Z80A signal specifying that an input or output is in progress or with M1* it specifies an interrupt acknowledge.
- 5 RESET* — system reset signal.
- 6 IOBUSINT* — input to the CPU signaling an interrupt from an I/O Bus device if I/O Bus interrupts are enabled.
- 7 IOBUSWAIT* — input to the CPU wait line allowing I/O Bus device to force wait states on the Z80 if external I/O is enabled.
- 8 EXTIOSEL* — input to I/O Bus Port circuit which switches the I/O Bus data bus transceiver and allows an INPUT instruction to read I/O Bus data.

The address line, data line, and all control lines except RESET* are enabled only when the ENEXIO bit in port EC is set to one.

To enable I/O interrupts, the ENIOBUSINT bit in the PORT E0 (output port) must be a one. However, even if it is disabled from generating interrupts, the status of the IOBUSINT* line can still read on the appropriate bit of CPU IOPORT E0 (input port).

See Model 4P Port Bit assignments for port 0FF, 0EC, and 0E0.

**Figure 4-15. Graphic Board Video Timing**

The Model 4P CPU board is fully protected from foreign I/O devices in that all the I/O Bus signals are buffered and can be disabled under software control. To attach and use an I/O device on the I/O Bus, certain requirements (both hardware and software) must be met.

For input port device use, you must enable external I/O devices by writing to port 0E0H with bit 4 on in the user software. This will enable the data bus address lines and control signals to the I/O Bus edge connector. When the input device is selected, the hardware should acknowledge by asserting EXTIOSEL* low. This switches the data bus transceiver and allows the CPU to read the contents of the I/O Bus data lines. See Figure 4-16 for the timing. EXTIOSEL* can be generated by NANDing IN and the I/O port address.

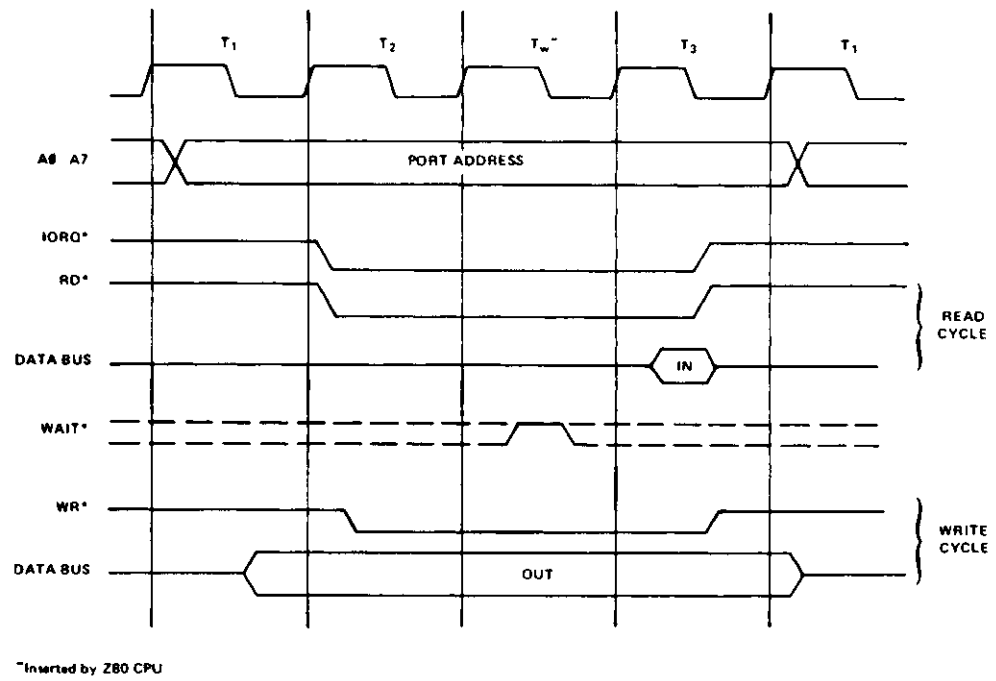
Output port device use is the same as the input port device in use, in that the external I/O devices must be enabled by writing to port 0E0H with bit 4 on in the user software — in the same fashion.

For either input or output devices, the IOBUSWAIT* control line can be used in the normal way for synchronizing slow devices to the CPU. Note that since dynamic memories are used in the Model 4P, the wait line should be used with caution. Holding the CPU in a wait state for 2 msec or more may cause loss of memory contents since refresh is inhibited during this time. It is recommended that the IOBUSWAIT* line be held active no more than 500 μ sec with a 25% duty cycle.

The Model 4P will support Z80 Mode 1 interrupts. A RAM jump table is supported by the LEVEL II BASIC ROMs image and the user must supply the address of his interrupt service routine by writing this address to locations 403E and 403F. When an interrupt occurs, the program will be vectored to the user-supplied address if I/O Bus interrupts have been enabled. To enable I/O Bus interrupts, the user must set bit 3 of Port 0E0H.

4.2.15 FDC Circuit

The TRS-80 Model 4P Floppy Disk Interface provides a standard 5-1/4 floppy disk controller. The Floppy Disk Interface supports both single and double density encoding schemes. Write precompensation can be software enabled or disabled beginning at any track, although the system software enables write precompensation for all tracks greater than twenty-one. The amount of write precompensation is 125 nsec and is not adjustable. One or two drives may be controlled by the interface. All data transfers are accomplished by CPU data requests. In double density operation, data transfers are synchronized to the CPU by forcing a wait to the CPU and clearing the wait by a data request from the FDC chip. The end of the data transfer is indicated by generation of a non-maskable interrupt from the interrupt request output of the FDC chip. A hardware watchdog timer insures that any error condition will not hang the wait line to the CPU for a period long enough to destroy RAM contents.



Input or Output Cycles with Wait States

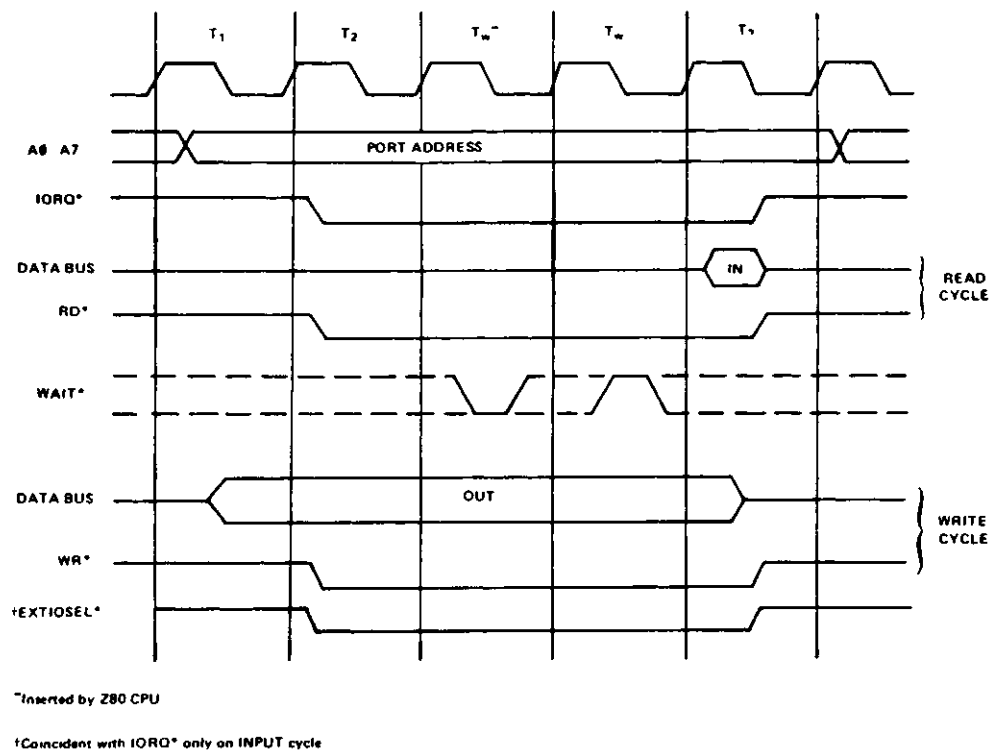


Figure 4-16. I/O Bus Timing Diagram

Control and Data Buffering

The Floppy Disk Controller Board is an I/O port mapped device which utilizes ports E4H, F0H, F1H, F2H, F3H, and F4H. The decoding logic is implemented on the CPU board. (Refer to Paragraph 5.1.5 Address Decoding for more information on Port Map). U70 is a bi-directional 8-bit transceiver used to buffer data to and from the FDC and RS-232 circuits. The direction of data transfer is controlled by the combination of control signals DISKIN*, RS232IN*, RDINT*, and RDNMI*. If any of these signals is active (logic low), U70 is enabled to drive data onto the CPU data bus. If both signals are inactive (logic high), U70 is enabled to receive data from the CPU board data bus. A second buffer (U36) is used to buffer the FDC chip data to the FDC RS232 Data Bus (BD0-BD7). U36 is enabled all the time and its direction controlled by DISKIN*. Again, if DISKIN* is active (logic low), data is enabled to drive from the FDC chip to the Main Data Busses. If DISKIN* is inactive (logic high), data is enabled to be transferred to the FDC chip.

Nonmaskable Interrupt Logic

Gate Array 4.4 (U18) is used to latch data bits D6 and D7 on the rising edge of the control signal WRNMI*. This enables the conditions which will generate a non-maskable interrupt to the CPU. The NMI interrupt conditions which are programmed by doing an OUT instruction to port E4H with the appropriate bits set. If data bit 7 is set, an FDC interrupt is enabled to generate an NMI interrupt. If data bit 7 is reset, interrupt requests from the FDC are disabled. If data bit 6 is set, a Motor Time Out is enabled to generate an NMI interrupt. If data bit 6 is reset, interrupts on Motor Time Out are disabled. An IN instruction from port E4H enables the CPU to determine the source of the non-maskable interrupt. Data bit 7 indicates the status of FDC interrupt request (INTRQ) (0 = true, 1 = false). Data bit 6 indicates the status of Motor Time Out (0 = true, 1 = false). Data bit 5 indicates the status of the Reset signal (0 = true, 1 = false). The control signal RDNMI* gates this status onto the CPU data bus when active (logic low).

Drive Select Latch and Motor ON Logic

Selecting a drive prior to disk I/O operation is accomplished by doing an OUT instruction to port F4H with the proper bit set. The following table describes the bit allocation of the Drive Select Latch.

Data Bit	Function
D0	Selects Drive 0 when set*
D1	Selects Drive 1 when set*
D2	Selects Drive 2 when set*
D3	Selects Drive 3 when set*
D4	Selects Side 0 when reset Selects Side 1 when set
D5	Write precompensation enabled when set, disabled when reset
D6	Generates WAIT if set
D7	Selects MFM mode if set Selects FM mode if reset

*Only one of these bits should be set per output

Hex D flip-flop U54 (74LS174) latches the drive select bits, side select and FM* MFM bits on the rising edge of the control signal DRVSEL*. Gate Array 4.4 (U18) is used to latch the Wait Enable and Write precompensation enable bits on the rising edge of DRVSEL*. The rising edge of DRVSEL* also triggers a one-shot (1.2 of U54, 74LS123) which produces a Motor On to the disk drives. The duration of the Motor On signal is approximately three seconds. The spindle motors are not designed for continuous operation. Therefore, the inactive state of the Motor On signal is used to clear the Drive Select Latch, which de-selects any drives which were previously selected. The Motor On one-shot is retriggerable by simply executing another OUT instruction to the Drive Select Latch.

Wait State Generation and WAITIMOUT Logic

As previously mentioned, a wait state to the CPU can be initiated by an OUT to the Drive Select Latch with D6 set. Pin 18 of U18 will go high after this operation. This signal is inverted by 1/4th of U15 and is routed to the CPU where it forces the Z80A into a wait state. The Z80A will remain in the wait state as long as WAIT* is low. Once initiated, the WAIT* will remain low until one of five conditions is satisfied. If INTRQ, DRQ, and RESET, inputs become active (logic high), it causes WAIT* to go high which allows the Z80 to exit the wait state. An internal timer in U18 serves as a watchdog timer to insure that a wait condition will not persist long enough to destroy dynamic RAM contents. This internal watchdog timer logic will limit the duration of a wait to 1024µsec, even if the FDC chip should fail to generate a DRQ or an INTRQ.

If an OUT to Drive Select Latch is initiated with D6 reset (logic low), a WAIT is still generated. The internal timer in U18 will count to 2 which will clear the WAIT state. This allows the WAIT to occur only during the OUT instruction to prevent violating any Dynamic RAM parameters.

NOTE: This automatic WAIT will cause a 5-1 µsec wait each time an out to Drive Select Latch is performed.

Clock Generation Logic

A 16 MHz crystal oscillator and a Gate Array 4.4 (U18) are used to generate the clock signals required by the FDC board. The 6 MHz oscillator is implemented internal to U18 and a quartz crystal (Y2). The output of the oscillator is divided by 2 to generate an 8 MHz clock. This is used by the FDC 1773 for all internal timing and data separation. U18 further divides the 16 MHz clock to drive the watchdog timer circuit.

Disk Bus Output Drivers

High current open collector drivers U15 and U34 are used to buffer the output signals from the FDC circuit to the disk drives.

Write Precompensation and Write Data Pulse Shaping Logic

All Write Precompensation is generated internal to the FDC chip 1773 (U17). Write Precompensation is enabled when W6 goes high and Write Precompensation is enabled from software. This signal is multiplexed with RDY by W6 is fed into pin 20 of U17. Write Data is output pin 22 of U17 and is shaped by a one-shot (1/2 of U56) which stretches the data pulses to approximately 500 nsec.

Floppy Disk Controller Chip

The 1773 is an MOS LSI device which performs the functions of a floppy disk formatter controller in a single chip implementation. The following port addresses are assigned to the internal registers of the 1773 FDC chip.

Port No.	Function
F0H	Command Status Register
F1H	Track Register
F2H	Sector Register
F3H	Data Register

4.2.16 RS-232-C Circuit

RS-232C Technical Description

The RS-232C circuit for the Model 4P computer supports asynchronous serial transmissions and conforms to the EIA RS-232C standards at the input-output interface connector (J4). The heart of the circuit is the TR1865 Asynchronous Receiver/Transmitter U33. It performs the job of converting the parallel byte data from the CPU to a serial data stream including start, stop, and parity bits. For a more detailed description of how this LSI circuit performs these functions, refer to the TR1865 data sheets and application notes. The transmit and receive clock rates that the TR1865 needs are supplied by the Baud Rate Generator U73 (BR1943). This circuit takes the 5.0688 MHz supplied by the system timing circuit and the programmed information received from the CPU over the data bus and divides the basic clock rate to provide two clocks. The rates available from the BRG go from 50 Baud to 19200 Baud. See the BRG table for the complete list.

BRG Programming Table

Nibble Loaded	Transmit Receive Baud Rate	16X Clock	Supported by SETCOM
0H	50	0.8 kHz	Yes
1H	75	1.2 kHz	Yes
2H	110	1.76 kHz	Yes
3H	134.5	2.1523 kHz	Yes
4H	150	2.4 kHz	Yes
5H	300	4.8 kHz	Yes
6H	600	9.6 kHz	Yes
7H	1200	19.2 kHz	Yes
8H	1800	28.8 kHz	Yes
9H	2000	32.081 kHz	Yes
AH	2400	38.4 kHz	Yes
BH	3600	57.6 kHz	Yes
CH	4800	76.8 kHz	Yes
DH	7200	115.2 kHz	Yes
EH	9600	153.6 kHz	Yes
FH	19200	307.2 kHz	Yes

The RS-232C circuit is port mapped and the ports used are E8 to EB. Following is a description of each port on both input and output.

Port	Input	Output
E8	Modem status	Master Reset, enables UART control register load
EA	UART status	UART control register load and modem control
E9	Not Used	Baud rate register load enable bit
EB	Receiver Holding register	Transmitter Holding register

Interrupts are supported in the RS-232C circuit by the Interrupt mask register and the Status register internal to GA 4.5 (U31) which allow the CPU to see which kind of interrupt has occurred. Interrupts can be generated on receiver data register full, transmitter register empty, and any one of the errors — parity, framing, or data overrun. This allows a minimum of CPU overhead in transferring data to or from the UART. The interrupt mask register is port E0 (write) and the interrupt status register is port E0 (read). Refer to the IO Port description for a full breakdown of all interrupts and their bit positions.

All Model I, III, and 4 software written for the RS-232-C interface is compatible with the Model 4P RS-232-C circuit, provided the software does not use the sense switches to configure the interface. The programmer can get around this problem by directly programming the BRG and UART for the desired configuration or by using the SETCOM command of the disk operating system to configure the interface. The TRS-80 RS-232C Interface hardware manual has a good discussion of the RS-232C standard and specific programming examples (Catalog Number 26-1145).

Pinout Listing

The following list is a pinout description of the DB-25 connector (P1).

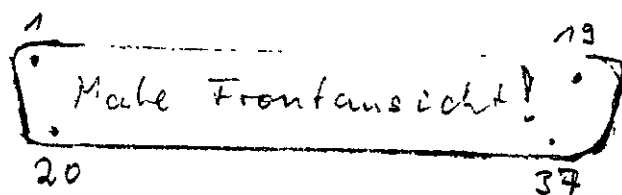
Pin No.	Signal
1	PGND (Protective Ground)
2	TD (Transmit Data)
3	RD (Receive Data)
4	RTS (Request to Send)
5	CTS (Clear To Send)
6	DSR (Data Set Ready)
7	SGND (Signal Ground)
8	CD (Carrier Detect)
19	SRTS (Spare Request to Send)
20	DTR (Data Terminal Ready)
22	RI (Ring Indicate)

**Model 4P Gate Array
I/O Pin Assignments**

J1		J2		J3	
Pin No.	Signal	Pin No.	Signal	Pin No.	Signal
1.	DATA STROBE	1.	XD0	1.	XD0
2.	GND	2.	GND	2.	GND
3.	PD0	3.	XD1	3.	XD1
4.	GND	4.	GND	4.	GND
5.	PD1	5.	XD2	5.	XD2
6.	GND	6.	GND	6.	GND
7.	PD2	7.	XD3	7.	XD3
8.	GND	8.	GND	8.	GND
9.	PD3	9.	XD4	9.	XD4
10.	GND	10.	GND	10.	GND
11.	PD4	11.	XD5	11.	XD5
12.	GND	12.	GND	12.	GND
13.	PD5	13.	XD6	13.	XD6
14.	GND	14.	GND	14.	GND
15.	PD6	15.	XD7	15.	XD7
16.	GND	16.	GND	16.	GND
17.	PD7	17.	XA0	17.	XA0
18.	GND	18.	GND	18.	GND
19.	N/A	19.	XA1	19.	XA1
20.	GND	20.	GND	20.	GND
21.	BUSY	21.	XA2	21.	XA2
22.	GND	22.	GND	22.	GND
23.	OUTPAPER	23.	XA3	23.	XA3
24.	GND	24.	GND	24.	GND
25.	UNIT SELECT	25.	XA4	25.	XA4
26.	NC	26.	GND	26.	GND
27.	GND	27.	XA5	27.	XA5
28.	FAULT	28.	GND	28.	GND
29.	N/A	29.	XA6	29.	XA6
30.	N/A	30.	GND	30.	GND
31.	NC	31.	XA7	31.	XA7
32.	N/A	32.	GND	32.	GND
33.	NC	33.	XIN*	33.	XIN*
34.	GND	34.	GND	34.	GND
35.		35.	XOUT*	35.	XOUT*
36.		36.	GND	36.	GND
37.		37.	XRESET*	37.	XRESET*
38.		38.	GND	38.	GND
39.		39.	IOBUSINT*	39.	IOBUSINT*
40.		40.	GND	40.	GND
41.		41.	IOBUSWAIT*	41.	IOBUSWAIT*
42.		42.	GND	42.	GND
43.		43.	EXTIOSEL*	43.	EXTIOSEL*
44.		44.	GND	44.	GND
45.		45.	NC	45.	NC
46.		46.	GND	46.	GND
47.		47.	XMI*	47.	XMI*
48.		48.	GND	48.	GND
49.		49.	XIOREQ*	49.	XIOREQ*
50.		50.	GND	50.	GND

J4		J5		J7		J9	
Pin No.	Signal	Pin No.	Signal	Pin No.	Signal	Pin No.	Signal
1.	PGND	1.	GND	1.	D0	1.	GND
2.	TD	2.	GND	2.	D1	2.	VOUT
3.	RD	3.	GND	3.	D2	3.	GND
4.	CTS	4.	GND	4.	D3	4.	VERTSYNC*
5.	DSR	5.	GND	5.	D4	5.	GND
6.	CD	6.	GND	6.	D5	6.	HORZSYNC
7.	SGND	7.	DIP*	7.	D6	7.	
8.	CD	8.	DIP*	8.	D7	8.	
9.		9.	GND	9.	GEN*	9.	
10.		10.	DS0*	10.	DCLK	10.	
11.		11.	GND	11.	A0	11.	
12.		12.	DS1*	12.	A1	12.	
13.		13.	GND	13.	A2	13.	
14.		14.	GND	14.	J	14.	
15.		15.	GND	15.	GRAFVID	15.	
16.		16.	MOTORON*	16.	ENGRAF	16.	
17.		17.	GND	17.	DISPEN	17.	
18.		18.	DIR*	18.	VSYN	18.	
19.	SRTS	19.	GND	19.	HSYN	19.	
20.	DTR	20.	STEP*	20.	RESET*	20.	
21.		21.	GND	21.	WAIT*	21.	
22.	RI	22.	WD*	22.	H	22.	
23.		23.	GND	23.	I	23.	
24.		24.	WG*	24.	IN*	24.	
25.		25.	GND	25.	GND	25.	
26.		26.	DTRK0*	26.	+5V	26.	
27.		27.	GND	27.		27.	
28.		28.	DWPRT*	28.	CL166*	28.	
29.		29.	GND	29.	GND	29.	
30.		30.	DRRD*	30.	+5V	30.	
31.		31.	GND	31.	GND	31.	
32.		32.	SDSEL	32.	+5V	32.	
33.		33.	GND	33.	GND	33.	
34.		34.		34.	+5V	34.	

Herangeführt auf externen Sub-D-Stecker 37pol.



$$1 \leq 1$$

$$3 \leq 2$$

$$5 \leq 3 \text{ usw.}$$

$$2 \leq 20$$

$$4 \leq 21$$

$$6 \leq 22 \text{ usw.}$$



SECTION V

CHIP SPECIFICATIONS



CHIP SPECIFICATIONS

4	4P	4 GATE ARRAY	4P GATE ARRAY
Motorola MC 6835	Motorola MC 6835	Motorola MC 6835	Motorola MC 6835
	Western Digital BR 1943 (BR 1941L) FD 1793 (WD 179X) FDC 9216 TR 1865 WD 1943-00	Western Digital BR 1943 TR 1865 WD 1773	Western Digital BR 1943 TR 1865 WD 1773
MMI PAL 16RGA (166) PAL 10L8 (208) PAL 16L8 (268) PAL 16L8 (368)	MMI PAL 16RGA S.T. PAL 10L8 V.T. PAL 10L8 C.T. PAL 16L8 MeMep PAL 16L8 Page Mep	MATRA Timing A. (4.1.1) Address A. (4.2.0) Video A. (4.3.0) VTI FDC A. (4.4.0) RS-232 A. (4.5.0)	MATRA Timing A. (4.1.1) Address A. (4.2.0) Video A. (4.3.0) VTI FDC A. (4.4.0) RS-232 A. (4.5.0)
Zilog 280 A	Zilog 280 A	Zilog 280 A	Zilog 280 A



ARRAY #: 4.1.1

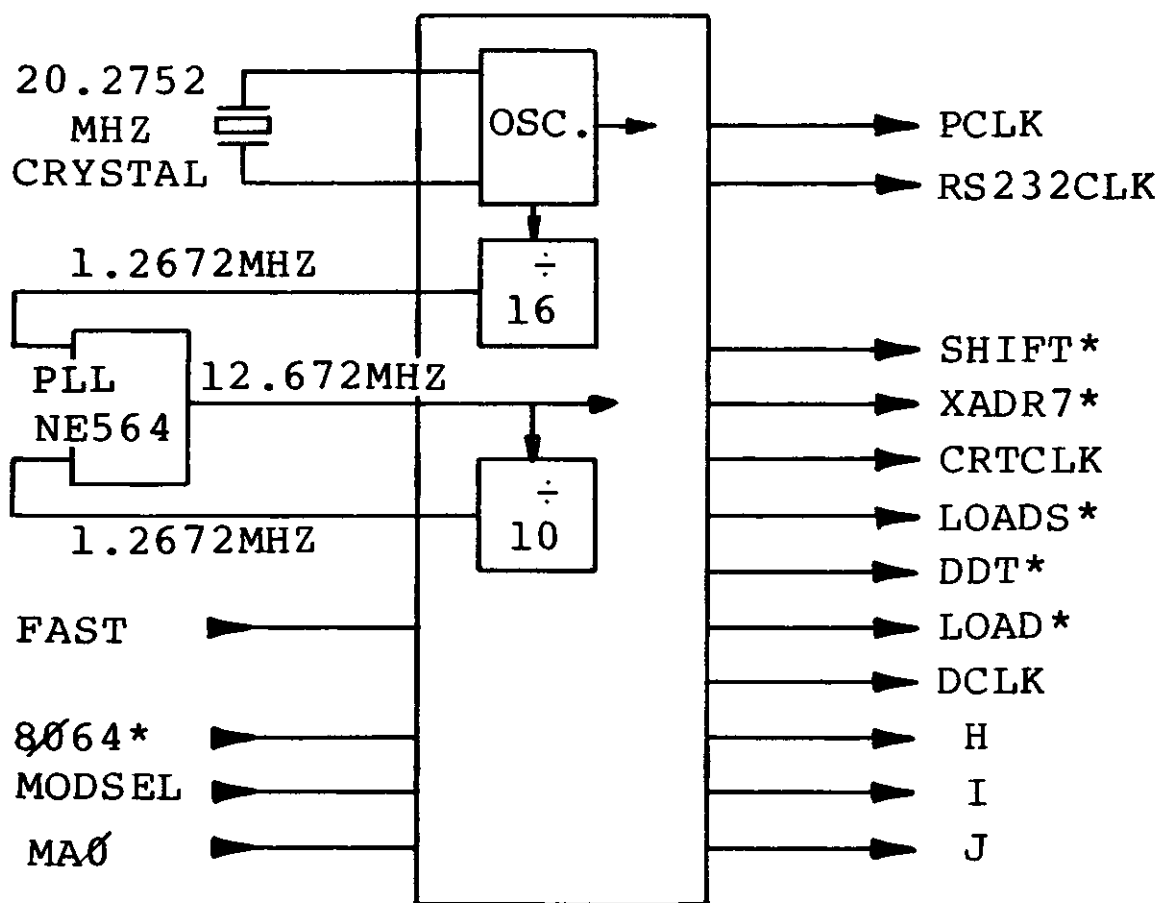
CIRCUIT NAME: System Timing

NO. OF PINS: 24

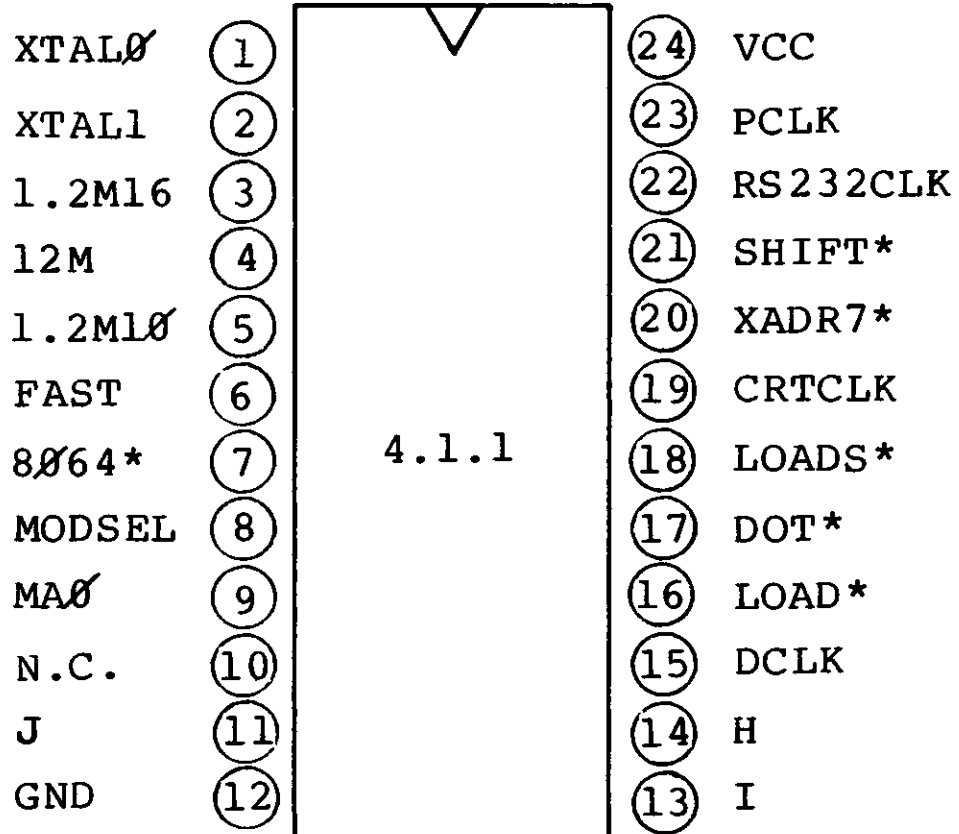
MAX. CLOCK FREQ.: 20.2752 MHz

OPER TEMP.: 0° C to 70° C

OPERATING VOLTAGE & RANGE: 5 V \pm 5%



24 PIN CHIP



SYSTEM TIMING SPECS

NUMBER	PARAMETER	MIN.	TYP.	MAX.	UNITS
1	20M Cycle Time		49.3		ns
2	20M Pulse Width (High)	20			ns
3	20M Pulse Width (Low)	20			ns
4	10M Cycle Time		98.6		ns
5	10M Pulse Width (High)	45 40			ns
6	10M Pulse Width (Low)	45 40			ns
7	RS232CLK Cycle Time		197.2		ns
8	RS232CLK Pulse Width (High)	92			ns
9	RS232CLK Pulse Width (Low)	92			ns
10	PCLK* (Fast) Cycle Time		246.6		ns
11	PCLK* (Fast) Pulse Width (High)	110			ns
12	PCLK* (Fast) Pulse Width (Low)	110			ns
13	PCLK* (/Fast) Cycle Time		493.2		ns
14	PCLK* (/Fast) Pulse Width (High)	180			ns
15	PCLK* (/Fast) Pulse Width (Low)	180			ns
16	PCLK* Rise Time			13	ns
17	PCLK* Fall Time			13	ns

DC CHARACTERISTICS (ALL PINS)

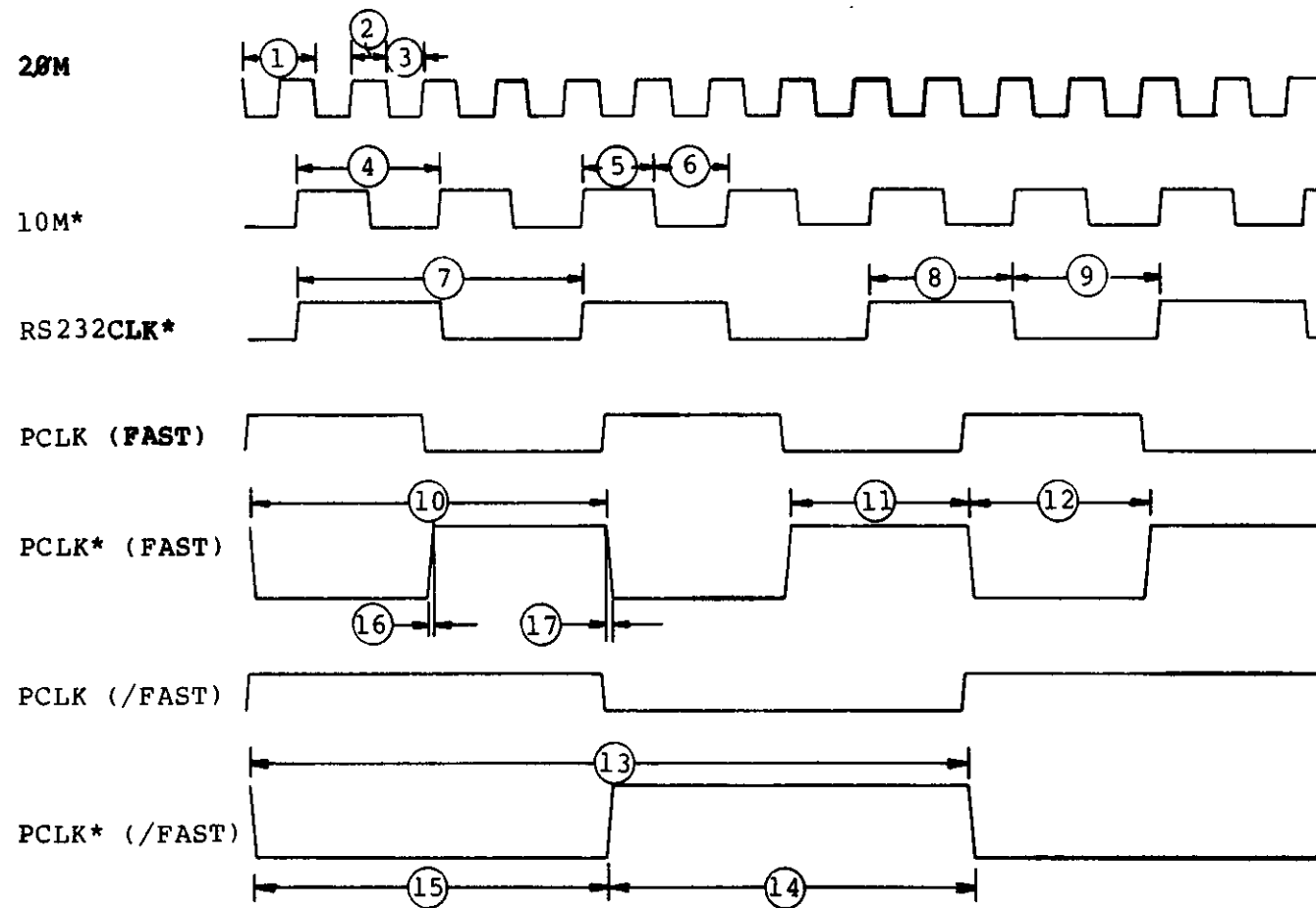
—	Input Voltage Level (High)	2.0			V
—	Input Voltage Level (Low)			.8	V
—	Output Voltage Level (High)	2.8	3.5		V
—	Output Voltage Level (Low)		.35	.5	V

(ALL PINS EXCEPT CRTCLK OUTPUT)

—	Input Current Level (High)			40	µa
—	Input Current Level (Low)			—1.6	ma
—	Output Current Level (High)	—160			µa
—	Output Current Level (Low)	3.2			ma

(CRTCLK OUTPUT)

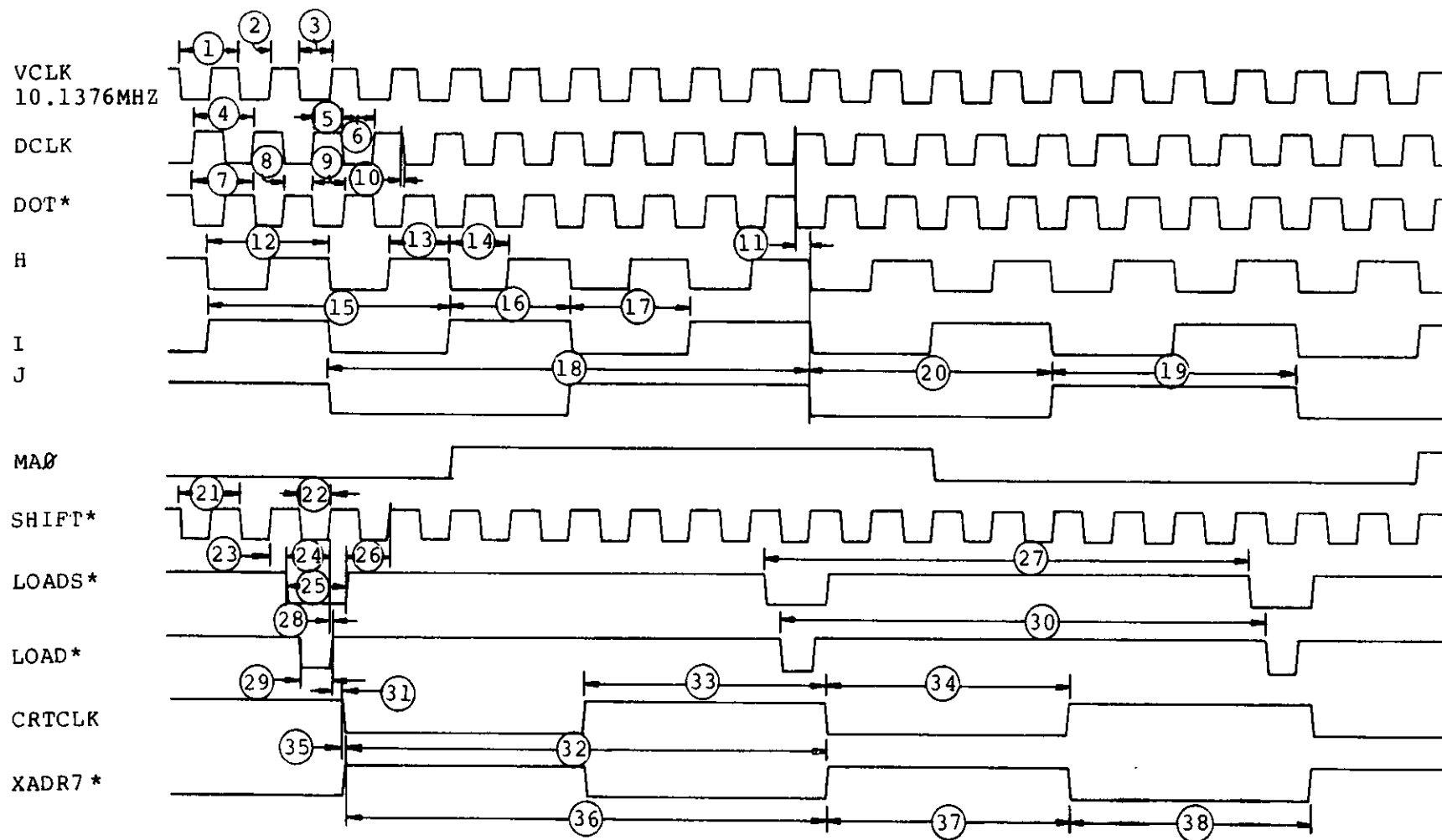
—	Output Current Level (High)	—400			µa
—	Output Current Level (Low)	8			ma



SYSTEM TIMING

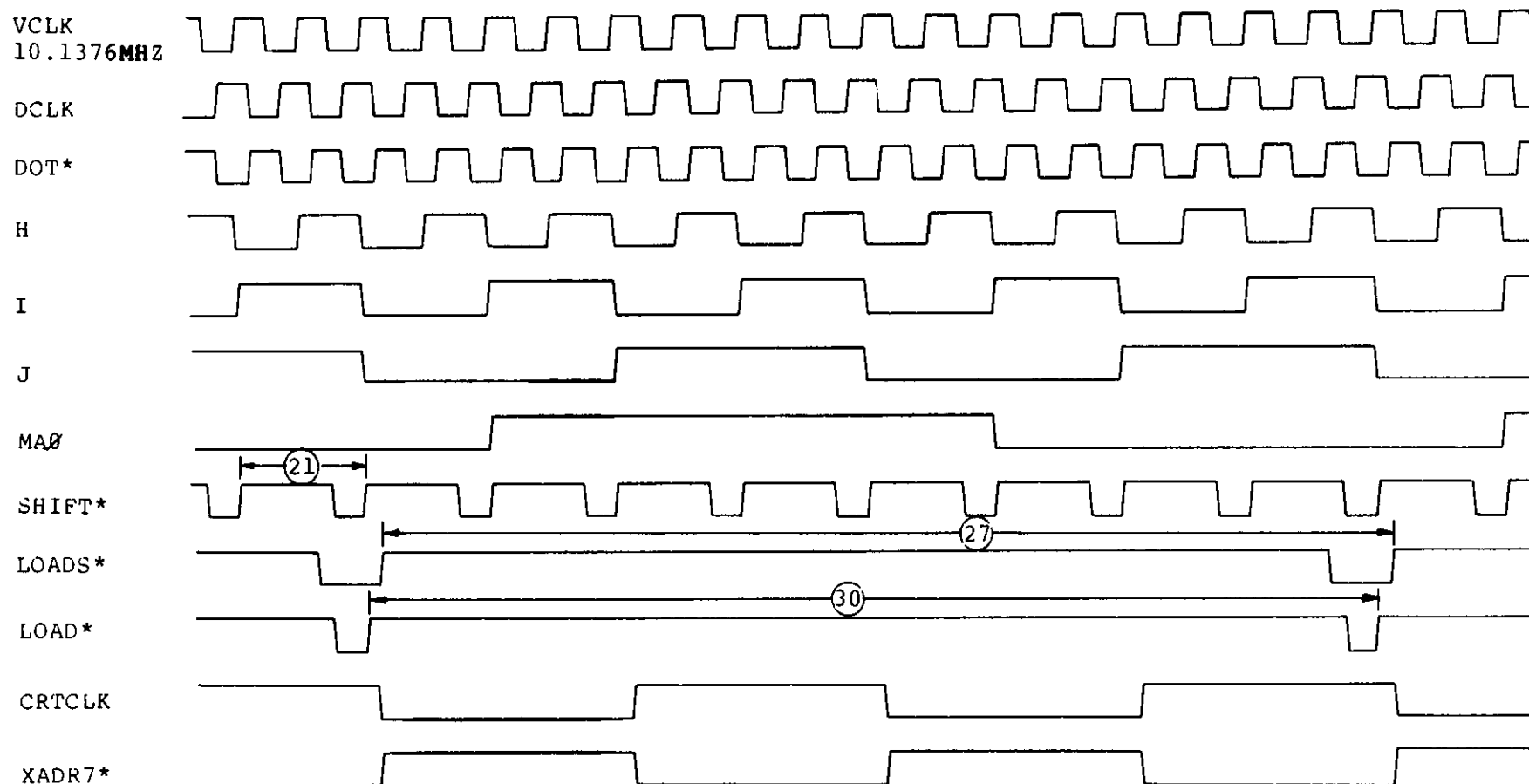
VIDEO TIMING SPECS

NUMBER	PARAMETER	10.1376 MHz			12.672 MHz			UNITS
		MIN.	TYP.	MAX.	MIN.	TYP.	MAX.	
1	VCLK Cycle Time		98.6			78.9		ns
2	VCLK Pulse Width (High)	40			30			ns
3	VCLK Pulse Width (Low)	40			30			ns
4	DCLK Cycle Time		98.6			78.9		ns
5	DCLK Pulse Width (High)	40			30			ns
6	DCLK Pulse Width (Low)	40			30			ns
7	DOT Cycle Time		98.6			78.9		ns
8	DOT Pulse Width (High)	40			30			ns
9	DOT Pulse Width (Low)	40			30			ns
10	DCLK ↓ to DOT ↑			5			5	ns
11	DCLK ↑ to H, I, J ↑↓			27			27	ns
12	H Cycle Time		197.2			157.8		ns
13	H Pulse Width (High)	90			70			ns
14	H Pulse Width (Low)	90			70			ns
15	I Cycle Time		394.4			315.6		ns
16	I Pulse Width (High)	190			150			ns
17	I Pulse Width (Low)	190			150			ns
18	J Cycle Time		788.8			631.2		ns
19	J Pulse Width (High)	385			305			ns
20	J Pulse Width (Low)	385			305			ns
21	SHIFT Cycle Time							
	(64x16 & 80x24 Mode)		98.6			78.9		ns
	(32x16 & 40x24 Mode)		197.2			157.8		ns
22	SHIFT Pulse Width (Low)	30			30			ns
23	SHIFT ↑ to LOADS ↓	0		27*	0		27*	ns
24	LOADS ↓ to SHIFT ↑	50*			50*			ns
25	LOADS Pulse Width (Low)	70	98.6		70	78.9		ns
26	LOADS ↑ to SHIFT ↑	50*			50*			ns
27	LOADS Cycle Time							
	(64x16 & 80x24 Mode)		788.8			631.2		ns
	(32x16 & 40x24 Mode)		1577.6			1262.4		ns
28	SHIFT ↑ to LOAD ↑			5			5	ns
29	LOAD Pulse Width (Low)	40			30			ns
30	LOAD Cycle Time							
	(64x16 & 80x24 Mode)		788.8			631.2		ns
	(32x16 & 40x24 Mode)		1577.6			1262.4		ns
31	LOAD ↑ to CRTCLK ↓	0		27	0		27	ns
32	CRTCLK Cycle Time		788.8			631.2		ns
33	CRTCLK Pulse Width (High)	385			305			ns
34	CRTCLK Pulse Width (Low)	385			305			ns
35	CRTCLK ↑↓ to XADR7 ↑↓			5			5	ns
36	XADR7 Cycle Time		788.8			631.2		ns
37	XADR7 Pulse Width (High)	385			305			ns
38	XADR7 Pulse Width (Low)	385			305			ns



VIDEO TIMING

64 X 16 MODE
80 X 24 MODE



VIDEO TIMING

32 X 16 MODE
40 X 24 MODE

4.1

<u>PIN</u>	<u>SIGNAL</u>	<u>MAX. CAPACITANCE</u>
23	PCLK	35 pf
22	RS232CK	105 pf
21	SHIFT*	35 pf
20	XADR7*	35 pf
19	CRTCLK	35 pf
18	LOADS*	35 pf
17	DOT*	35 pf
16	LOAD*	35 pf
15	DCLK	35 pf
14	H	35 pf
13	I	35 pf
11	J	35 pf

ARRAY #: 4.2.1

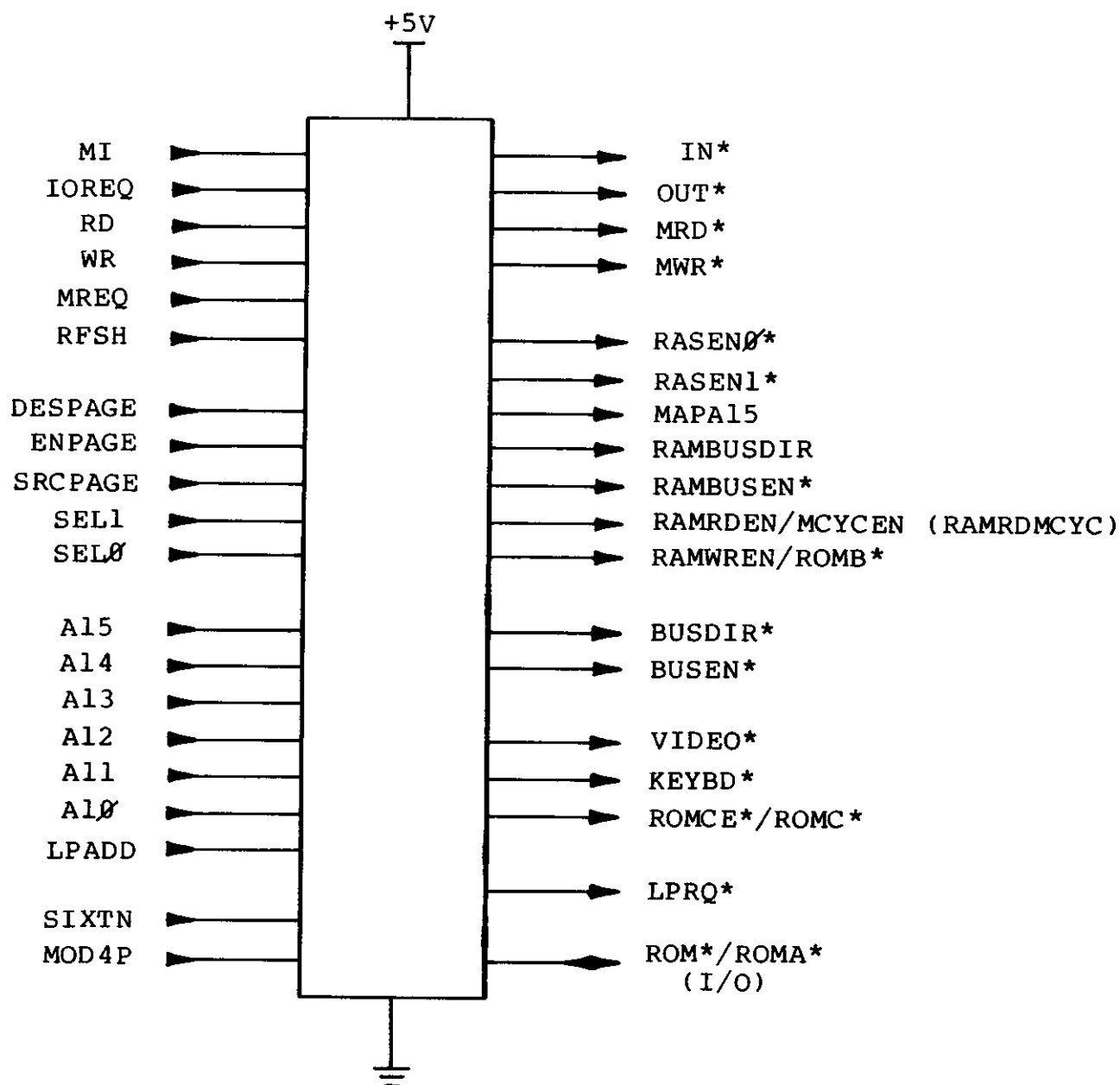
CIRCUIT NAME: Address Decode

NO. OF PINS: 40

MAX. CLOCK FREQ.: 4 MHz

OPER. TEMP.: 0° C to 70° C

OPERATING VOLTAGE & RANGE: $5 \pm 5\%$

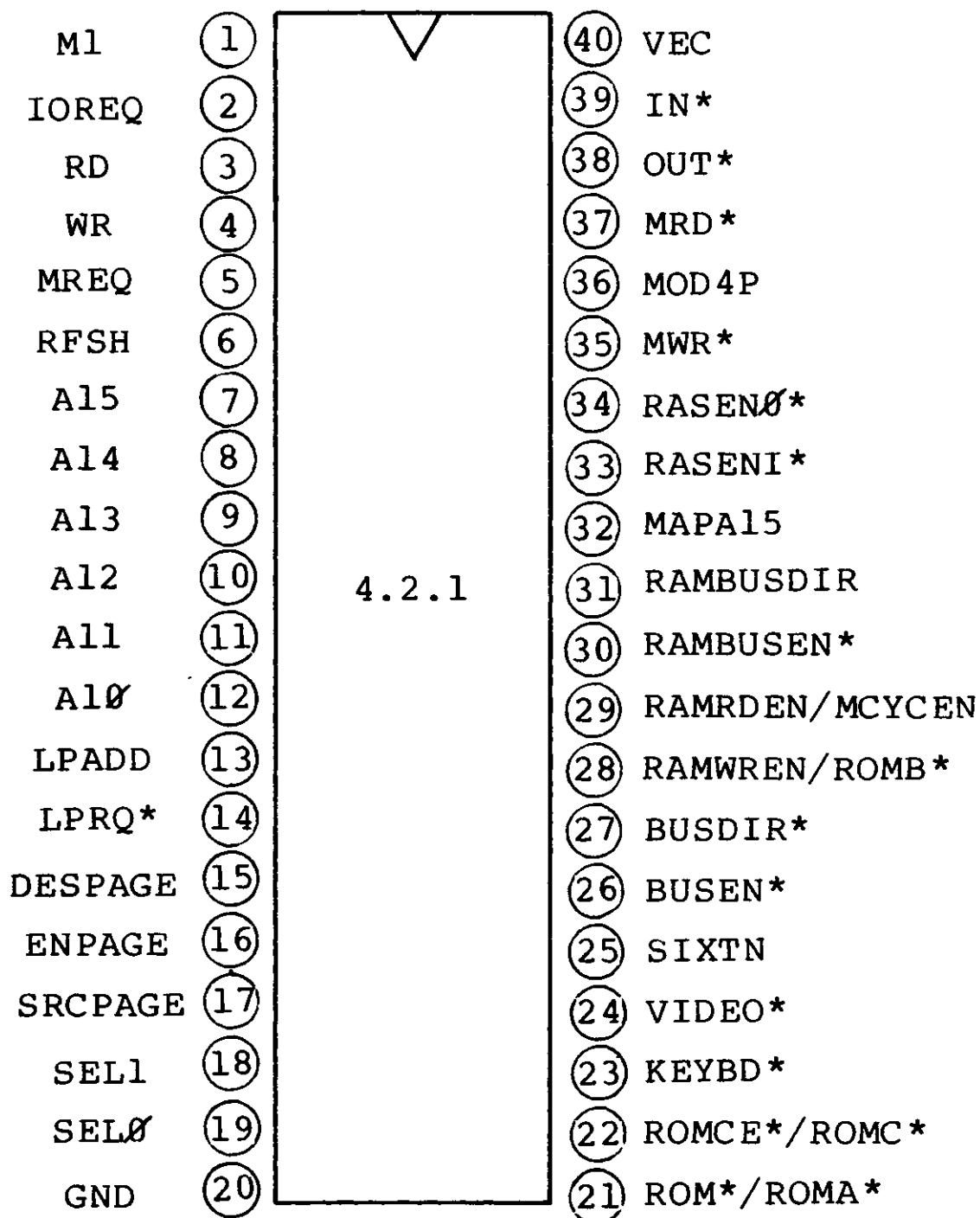


40 PINS USED

40 PIN CHIP

4.2.0

ADDRESS DECODE



SIGNAL NAME

MODEL A MODE

MODEL 4 MODE

MDD4P

"I" = +SV

"0" = GND

MI
 IOREQ
 RD
 WR
 MREQ
 RFSH
 DESPAGE
 ENPAGE
 SRCPAGE
 SEL1
 SEL0
 A15
 A14
 A13
 A12
 A11
 A10
 LPADD
 SIXTN

 IN*
 OUT*
 MRD*
 MWR*
 RASEN0*
 RASEN1*
 MAPA15
 RAMBUSDIR
 RAMBUSEN*
 (RAMRDMCYC) RAM RDEN/MCYCEN
 RAM WREN/ROMB*
 BUSDIR*
 BUSEN*
 VIDEO*
 KEYBD*
 ROMCE*/ROMC*
 LPRQ*
 ROM*/ROMA*

MI I
 IOREQ I
 RD I
 WR I
 MREQ I
 RFSH I
 DESPAGE I
 ENPAGE I
 SRCPAGE I
 SEL1 I
 SEL0 I
 A15 I
 A14 I
 A13 I
 A12 I
 A11 I
 A10 I
 LPADD I
 SIXTN I

 IN* O
 OUT* O
 MRD* O
 MWR* O
 RASEN0* O
 RASEN1* O
 MAPA15 O
 RAMBUSDIR O
 RAMBUSEN* O
 RAMRDEN O
 RAMWREN O
 BUSDIR* O
 BUSEN4P* O
 VIDEO4P* O
 KEYBD4P* O
 ROMCE* O
 LPRQ* O
 ROM* I

MI I
 IOREQ I
 RD I
 WR I
 IOREQ I
 RFSH I
 DESPAGE I
 ENPAGE I
 SRCPAGE I
 SEL1 I
 SEL0 I
 A15 I
 A14 I
 A13 I
 A12 I
 A11 I
 A10 I
 LPADD I
 SIXTN I

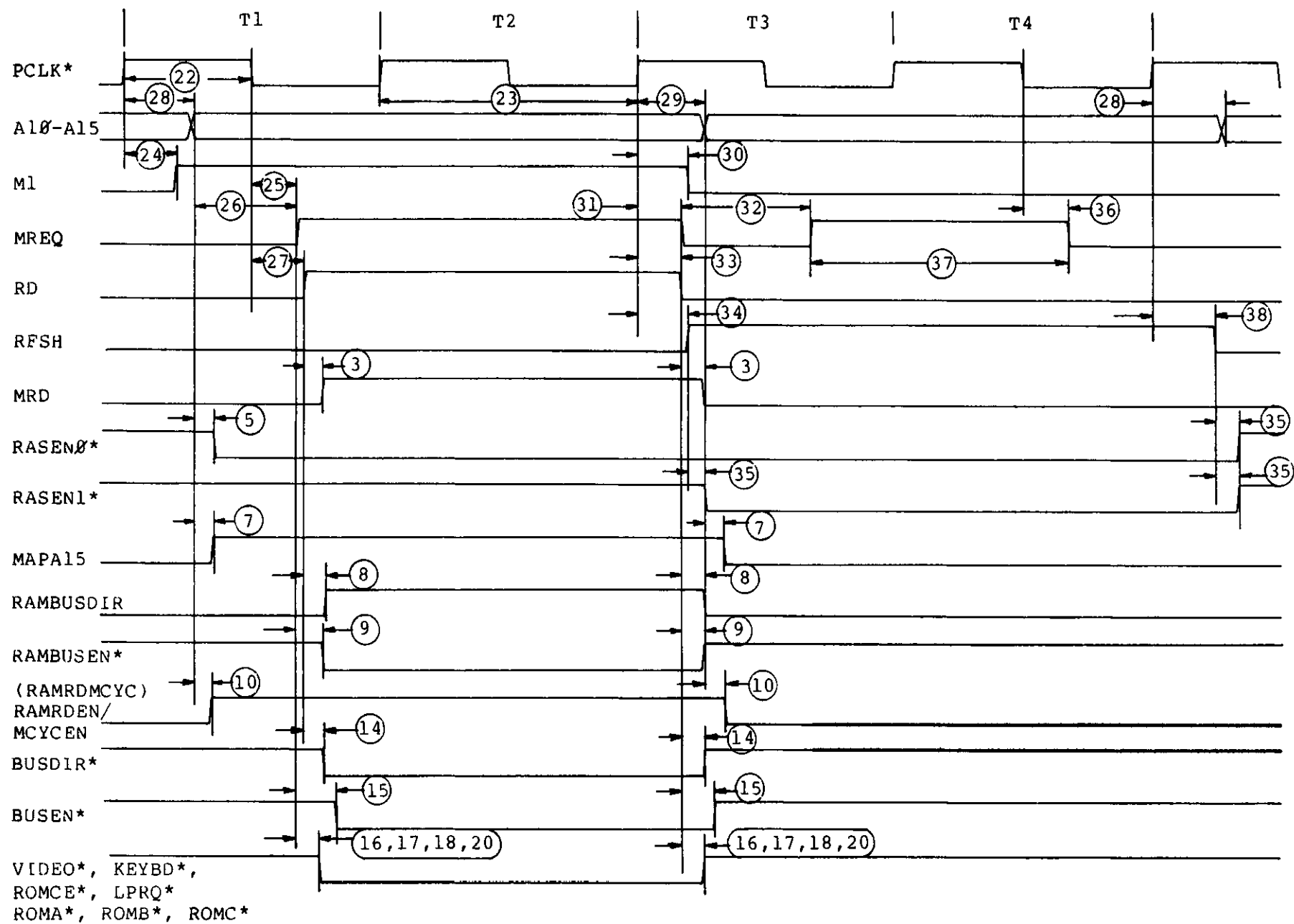
 IN* O
 OUT* O
 MRD* O
 MWR* O
 RASEN0* O
 RASEN1* O
 MAPA15 O
 RAMBUSDIR O
 RAMBUSEN* O
 MCYCEN O
 ROMB* O
 BUSDIR* O
 DATAcnt* O
 VIDEO4* O
 KEYBD4* O
 ROMC* O
 LPRQ* O
 ROMA* O

I = INPUT

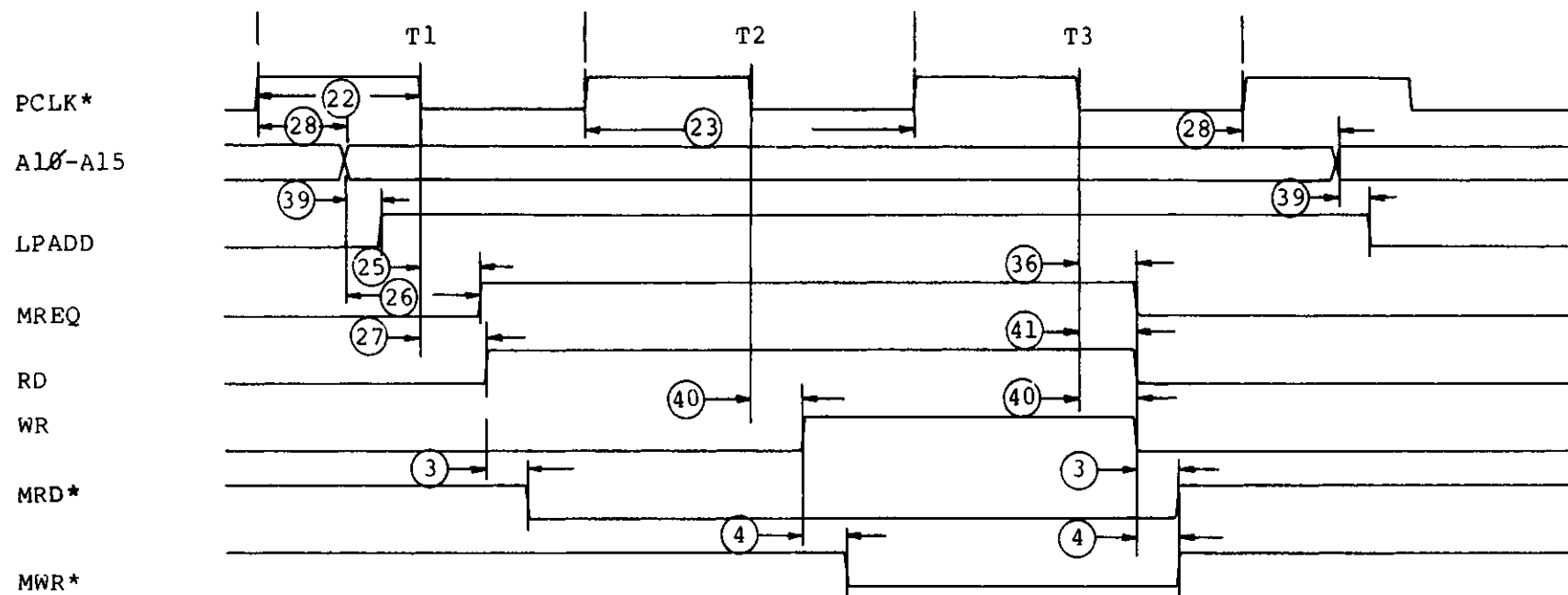
O = OUTPUT

	PARAMETER	SPECS			UNITS
		MIN.	TYP.	MAX.	
1	IOREQ $\uparrow\downarrow$ * RD $\uparrow\downarrow$ to \overline{IN} $\downarrow\uparrow$			35	ns
2	IOREQ $\uparrow\downarrow$ * WR $\uparrow\downarrow$ to \overline{OUT} $\downarrow\uparrow$			35	ns
3	RD $\uparrow\downarrow$ to \overline{MRD} $\downarrow\uparrow$			35	ns
4	WR $\uparrow\downarrow$ to \overline{MWR} $\downarrow\uparrow$			35	ns
5	A15 $\uparrow\downarrow$ to RASEN0 $\uparrow\downarrow$			50	ns
6	A15 $\uparrow\downarrow$ to RASEN1 $\uparrow\downarrow$			50	ns
7	A15 $\uparrow\downarrow$ to MAPA15 $\uparrow\downarrow$			50	ns
8	RD $\downarrow\uparrow$ to RAMBUSDIR $\downarrow\uparrow$			35	ns
9	MREQ $\uparrow\downarrow$ to RAMBUSEN $\downarrow\uparrow$			35	ns
10	A15-A10 $\uparrow\downarrow$ to RAMRDMCYC $\uparrow\downarrow$			50	ns
11	A15-A14 $\uparrow\downarrow$ to RAMWREN $\uparrow\downarrow$			50	ns
12	MREQ $\uparrow\downarrow$ to \overline{ROMB} $\downarrow\uparrow$			35	ns
13	IOREQ $\uparrow\downarrow$ to \overline{BUSDIR} $\downarrow\uparrow$			35	ns
14	RD $\uparrow\downarrow$ to \overline{BUSDIR} $\downarrow\uparrow$			35	ns
15	MREQ $\uparrow\downarrow$ to \overline{BUSEN} $\downarrow\uparrow$			50	ns
16	MREQ $\uparrow\downarrow$ to \overline{VIDEO} $\downarrow\uparrow$			35	ns
17	MREQ $\uparrow\downarrow$ to \overline{KEYBD} $\downarrow\uparrow$			35	ns
18	MREQ $\uparrow\downarrow$ to \overline{ROMCE} $\downarrow\uparrow$			35	ns
19	MREQ $\uparrow\downarrow$ to \overline{ROMC} $\downarrow\uparrow$			35	ns
20	MREQ $\uparrow\downarrow$ to \overline{LPRQ} $\downarrow\uparrow$			35	ns
21	MREQ $\uparrow\downarrow$ to \overline{ROMA} $\downarrow\uparrow$			35	ns
22	\overline{PCLK} $\uparrow\downarrow$ to \overline{PCLK} $\downarrow\uparrow$	110	123		ns
23	PCLK Cycle Time		246		ns
24	\overline{PCLK} \uparrow to M1 \uparrow			106	ns
25	\overline{PCLK} \downarrow to MREQ \uparrow			91	ns
26	A10-A15 $\uparrow\downarrow$ to MREQ \uparrow	50			ns
27	\overline{PCLK} \downarrow to RD \uparrow			101	ns
28	\overline{PCLK} \uparrow to A10-A15 $\uparrow\downarrow$			128	ns
29	\overline{PCLK} \uparrow to A10-A15 $\uparrow\downarrow$			128	ns
30	\overline{PCLK} \uparrow to M1 \downarrow			136	ns
31	\overline{PCLK} \uparrow to MREQ \downarrow			91	ns
32	MREQ \downarrow to MREQ \uparrow	110			ns
33	\overline{PCLK} \uparrow to RD \downarrow			91	ns
34	\overline{PCLK} \uparrow to RFSH \uparrow			136	ns
35	RFSH $\uparrow\downarrow$ to $\overline{RASEN0}$ or $\overline{RASEN1}$ $\uparrow\downarrow$			35	ns
36	\overline{PCLK} \downarrow to MREQ \downarrow			91	ns
37	MREQ Pulse Width (High)	220		126	ns
38	\overline{PCLK} \uparrow to RFSH \downarrow				
39	A1-A9 $\uparrow\downarrow$ to LPADD $\uparrow\downarrow$			30	ns
40	\overline{PCLK} \downarrow to WR $\uparrow\downarrow$			86	ns
41	\overline{PCLK} \downarrow to RD \downarrow			91	ns
42	Control Lines $\uparrow\downarrow$ to Affected Signals $\uparrow\downarrow$			35	ns
43	A0-A15 $\uparrow\downarrow$ to IOREQ \uparrow	200			ns
44	\overline{PCLK} \uparrow to IOREQ \uparrow			81	ns
45	\overline{PCLK} \uparrow to RD \uparrow			91	ns
46	\overline{PCLK} \uparrow to WR \uparrow			71	ns

M1 CYCLE

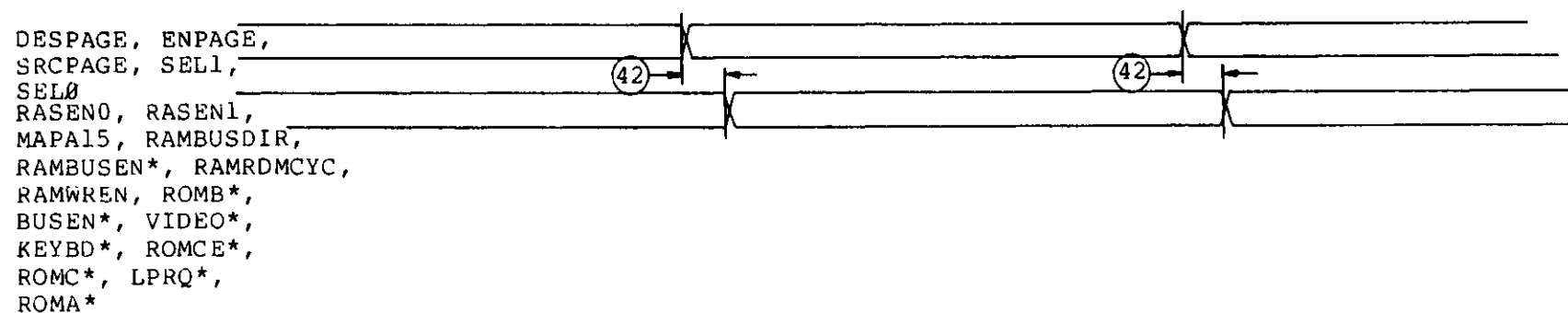


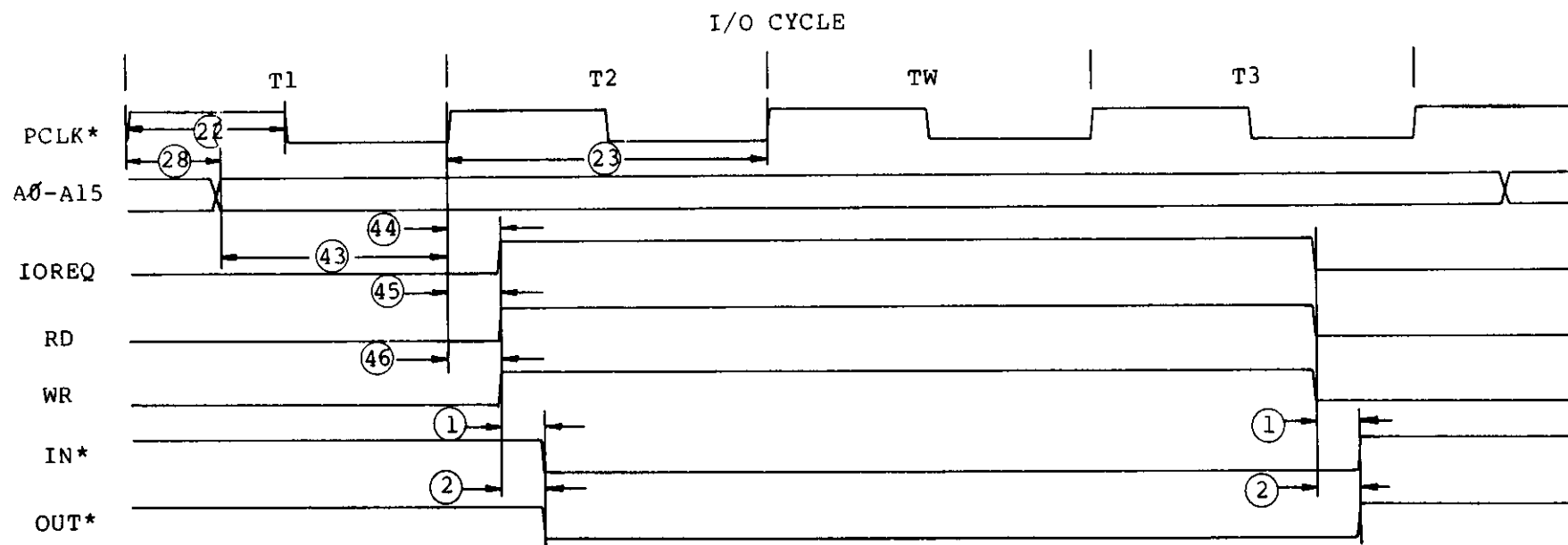
READ OR WRITE CYCLE



RAS_{EN0}, RAS_{EN1}, MAPA15, RAMBUSDIR, RAMBUSEN*, RAMRDMCYC, BUSDIR*, BUSEN*,
VIDEO*, KEYBD*, ROMCE*, LPRQ, ROMA*, ROMB*, ROMC* - Refer to M1 Cycle

CONTROL LINES





DC CHARACTERISTICS (ALL PINS) 0° – 70° C

PARAMETER	MIN.	TYP.	MAX.	UNITS
Input Voltage Level (High)	2.0			V
Input Voltage Level (Low)			.8	V
Output Voltage Level (High)	2.7	3.5		V
Output Voltage Level (Low)		.35	.5	V

(ALL PINS EXCEPT OUT*, RAMRDEN/MCYCEN)

Input Current Level (High)			20	μa
Input Current Level (Low)			-.4	ma
Output Current Level (High)	-200			μa
Output Current Level (Low)	4			ma

(OUT*, RAMRDEN/MCYCEN)

Output Current Level (High)	-400			μa
Output Current Level (Low)	8			ma

	<u>PIN</u>	<u>SIGNAL</u>	<u>MAX. CAPACITANCE</u>
	39	IN*	35 pf
	38	OUT*	35 pf
	37	MRD*	35 pf
	35	MWR*	128 pf
	34	RASEN0*	35 pf
	33	RASEN1*	35 pf
	32	MAPA15	35 pf
	31	RAMBUSDIR	35 pf
	30	RAMBUSEN*	35 pf
	29	RAMRDEN/MCYCEN	35 pf
	28	RAMWREN/ROMB*	35 pf
	27	BUSDIR*	35 pf
	26	BUSEN*	35 pf
	24	VIDEO*	35 pf
	23	KEYBD*	35 pf
	22	ROMCE*/ROMC*	35 pf
(OUTPUT)	21	ROMA*	35 pf
	14	LPRQ*	35 pf

ARRAY #: 4.3.0

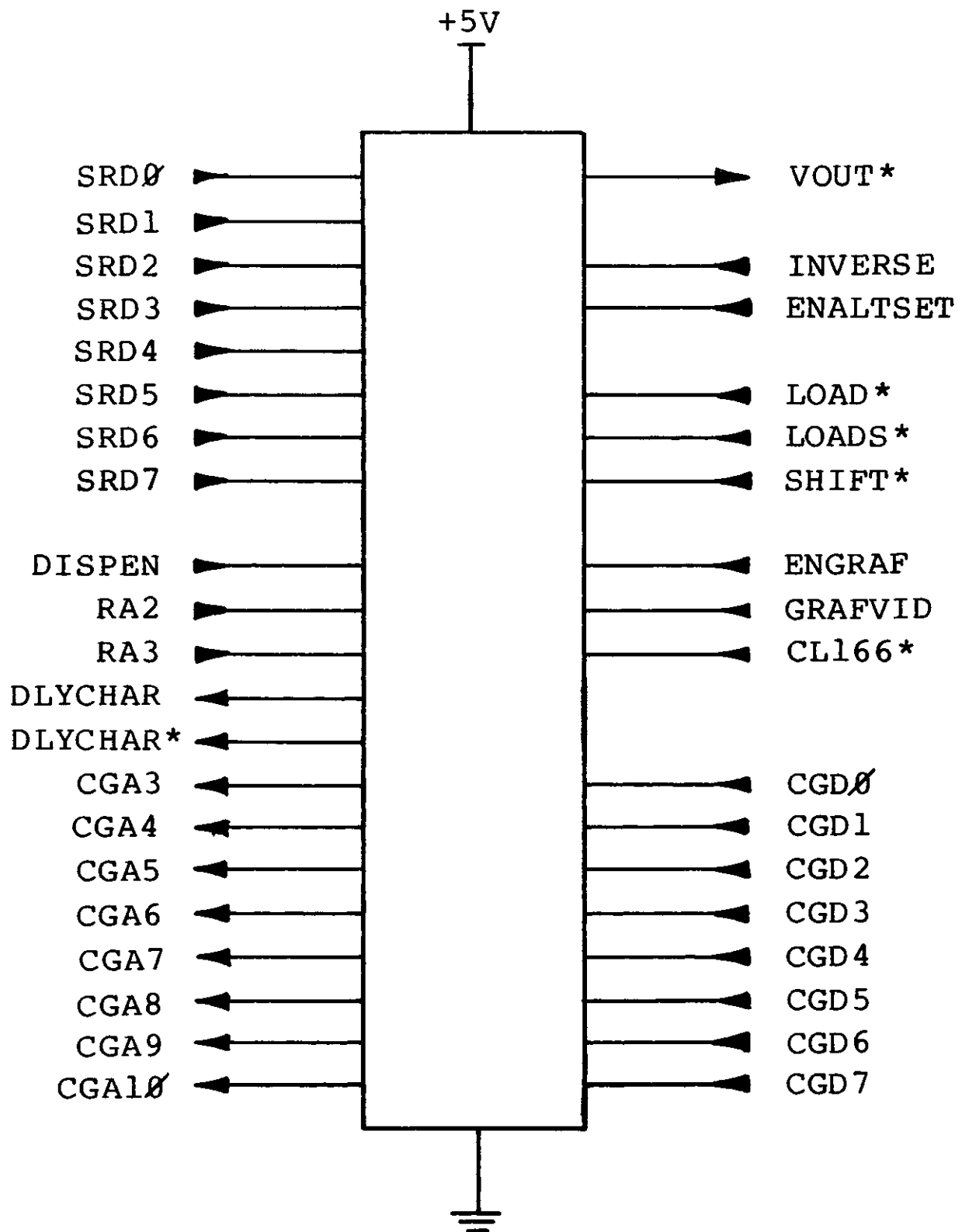
CIRCUIT NAME: Video Support

NO. OF PINS: 40

MAX. CLOCK FREQ.: 12.672 MHz

OPER. TEMP.: 0° C to 70° C

OPERATING VOLTAGE & RANGE: $5 \pm 5\%$



39 PINS USED

40 PIN CHIP

4.3.0

VIDEO SUPPORT

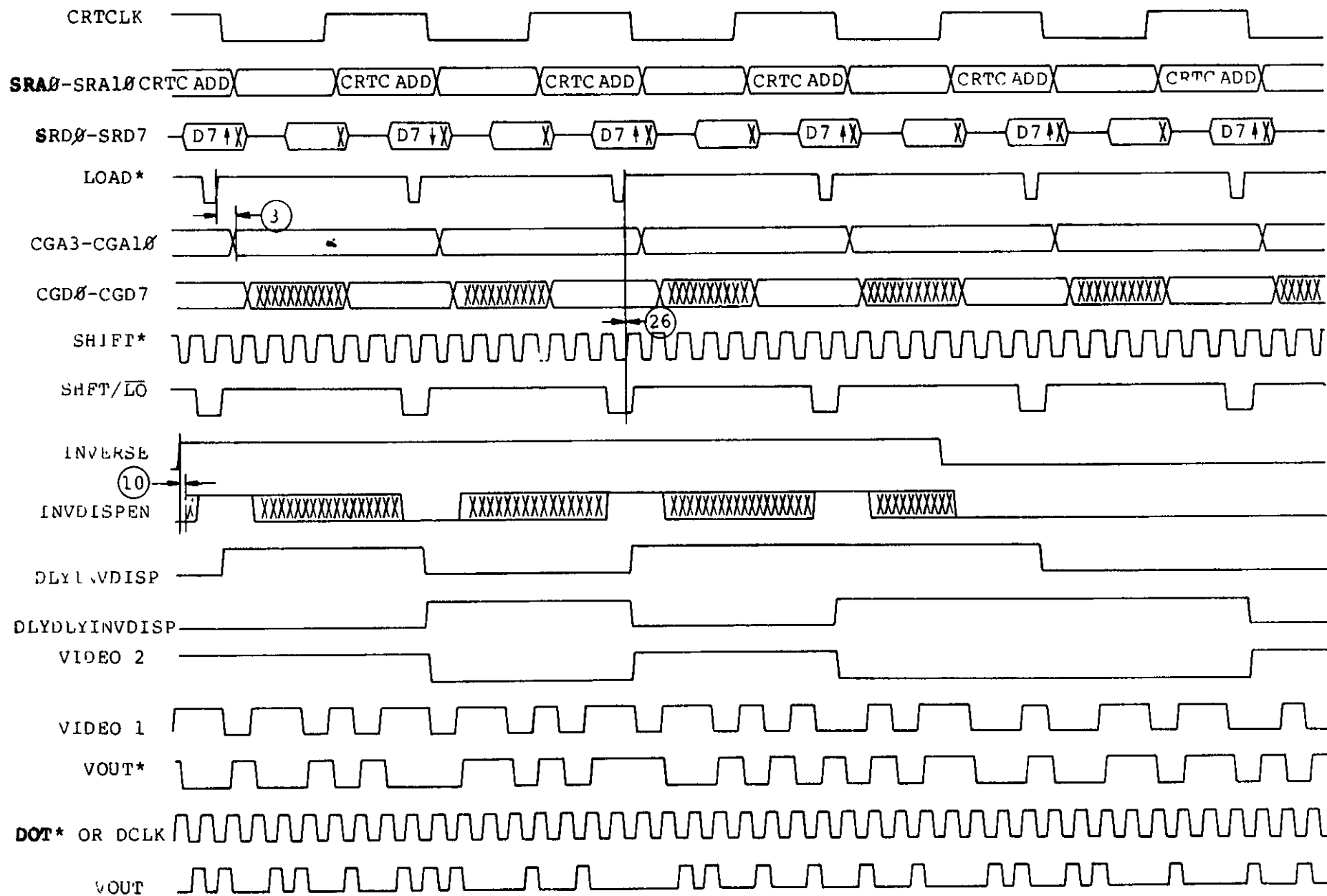
CGA7	1		40	+5V
CGA8	2		39	CGA6
CGA9	3		38	CGA5
CGA10	4		37	CGA4
SRD7	5		36	CGA3
SRD6	6		35	RA3
SRD5	7		34	RA2
SRD4	8		33	CGD7
SRD3	9		32	CGD6
SRD2	10		31	CGD5
SRD1	11		30	CGD4
SRD0	12		29	CGD3
DLYCHAR*	13		28	CGD2
DLYCHAR	14		27	CGD1
DISPEN	15		26	CGD0
CL166 *	16		25	INVERSE
ENGRAF	17		24	ENALTSET
GRAFVID	18		23	LOAD*
VOUT *	19		22	LOADS*
GND	20		21	SHIFT*

		SPECS			UNITS
PARAMETER		MIN.	TYP.	MAX.	
1**	SRD0—SRD7 ↑↓ to $\overline{\text{LOAD}}$ ↑	61			ns
2*	Inputs D0—D7 of LS273 ↑↓ to $\overline{\text{LOAD}}$ ↑	20			ns
3	$\overline{\text{LOAD}}$ ↑ to CGA3—CGA10 ↑↓	0		60	ns
4	RA2, RA3 ↑↓ to Outputs of LS153 ↑↓	0		38	ns
5	Inputs CGA3—CGA10 of LS153 ↑↓ to Outputs ↑↓	0		30	ns
6	$\overline{\text{DLYGRAPHIC}}$ ↓ to Outputs of LS244 ↑↓	0		30	ns
7	$\overline{\text{DLYGRAPHIC}}$ ↑ to Outputs of LS244 Tristate	0		30	ns
8	ENALTSET ↑↓ to CGA9 ↑↓	0		35	ns
9	INVERSE ↑↓ to Inputs D7 of LS273 ↑↓	0		35	ns
10	INVERSE ↑↓ to INVDISPEN, CHAR ↑↓	0		40	ns
11	INVERSE ↑↓ to Input to 51 ↑↓	0		20	ns
12	SRD6 ↑↓ to CHAR ↑↓	0		40	ns
13	DISPEN ↑↓ to Input D0 of LS175 ↑↓	0		20	ns
14	DISPEN ↑↓ to INVDISPEN ↑↓	0		40	ns
15	ENGRAF ↑↓ to INVDISPEN ↑↓	0		40	ns
16	ENGRAF ↑↓ to Inputs of 51 ↑↓	0		20	ns
17	GRAFVID ↑↓ to Input of 51 ↑↓	0		5	ns
20**	CGD0—CGD7 ↑↓ to LOADS ↓ & SHIFT ↑	100			ns
21	RA3 ↑↓ to DLYBLANK ↑↓	0	27	50	ns
22	$\overline{\text{LOAD}}$ ↑ to DLYBLANK ↑↓	0	27	50	ns
23**	LOADS ↓ to $\overline{\text{SHIFT}}$ ↑	50			ns
24*	$\overline{\text{SHIFT/LD}}$ ↓ to $\overline{\text{SHIFT}}$ ↑	30			ns
25	$\overline{\text{CL166}}$ ↑↓ to QH ↑↓	0		30	ns
26*	$\overline{\text{LOAD}}$ ↑ to $\overline{\text{SHIFT}}$ ↑			± 5	ns
27 ¹	$\overline{\text{LOAD}}$ ↑ to VIDEO2 ↑↓ = $\overline{\text{SHIFT}}$ ↑ to VIDEO1 ↑↓			± 5	ns
28	GRAFVID ↑↓ to VIDEO2 ↑↓	0		15	ns
29	VIDEO2 ↑↓, VIDEO1 ↑↓ to $\overline{\text{VOUT}}$ ↑↓	0		20	ns
30	ENGRAF ↑↓ to VIDEO2 ↑↓	0		15	ns
31	DLYCHAR* ↑ to CGD0—CGD7 Tristate			150	ns
32	CRTCLK ↓ to DISPEN			300	ns

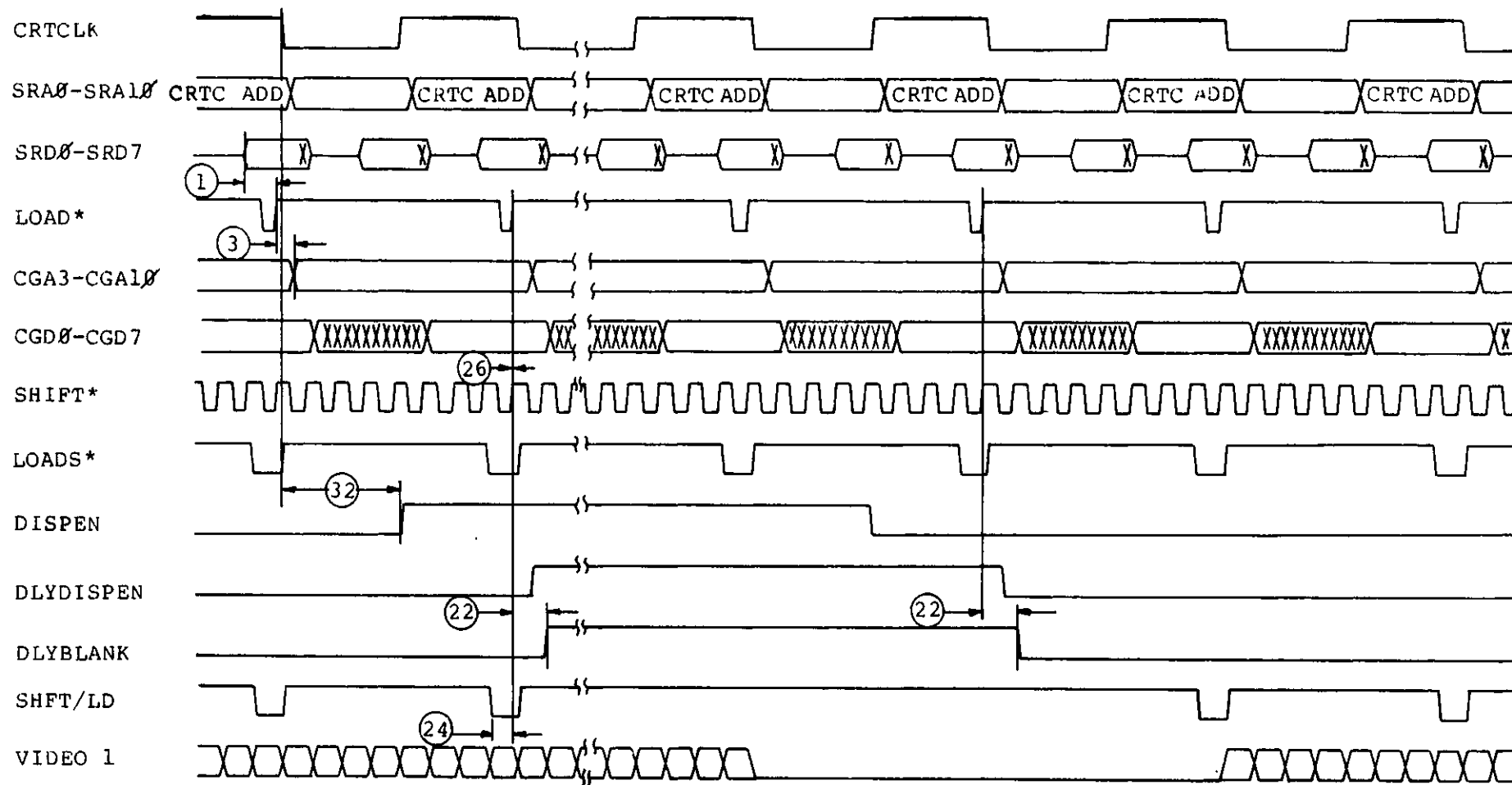
¹ The delay from $\overline{\text{LOAD}}$ ↑ to VIDEO2 ↑↓ should equal the delay from $\overline{\text{SHIFT}}$ ↑ to VIDEO1 ↑↓.

* Specs required for TLL components—can be changed to meet the setup & hold time specs of array logic.

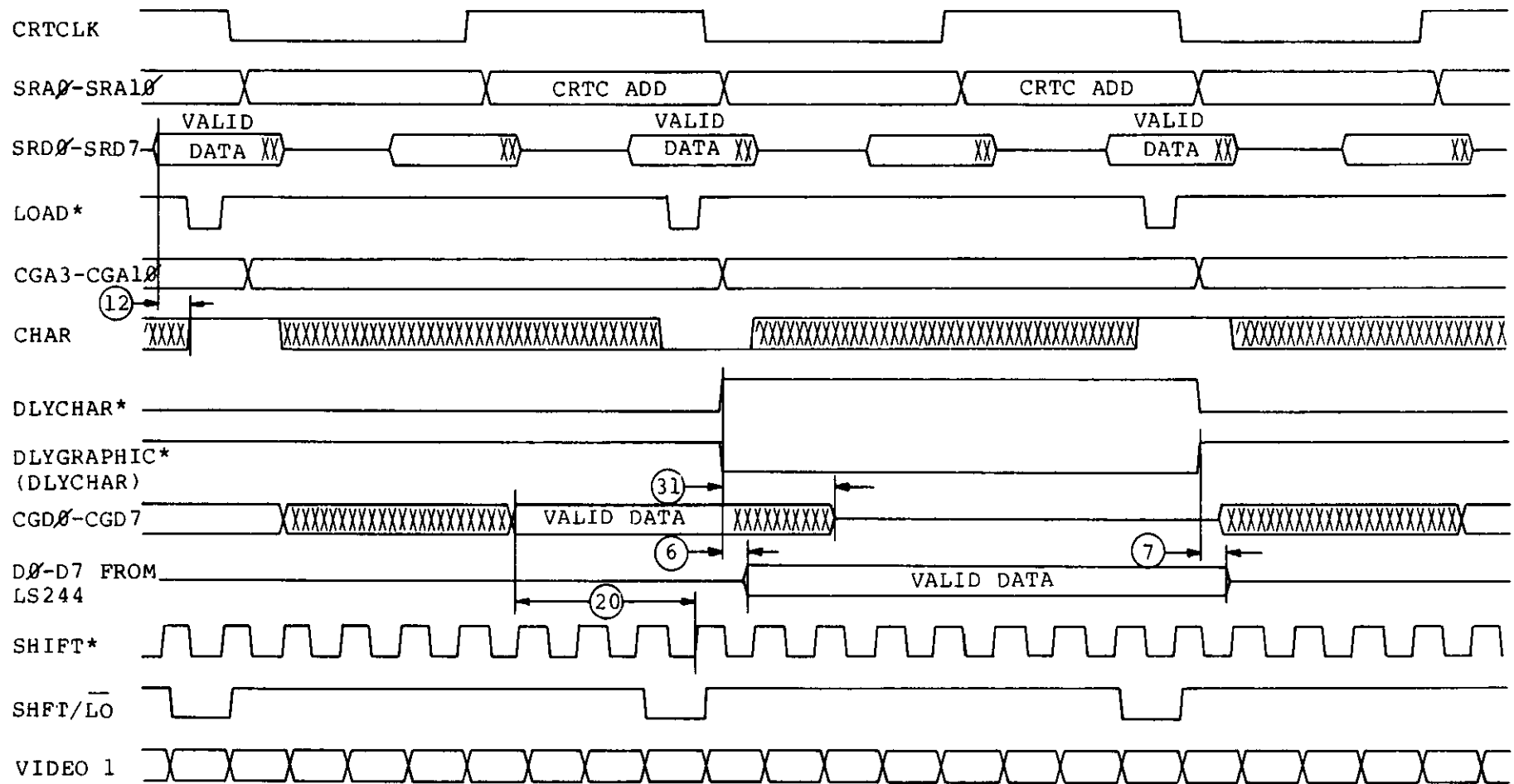
** Specs provided are for reference, timing is from external logic.



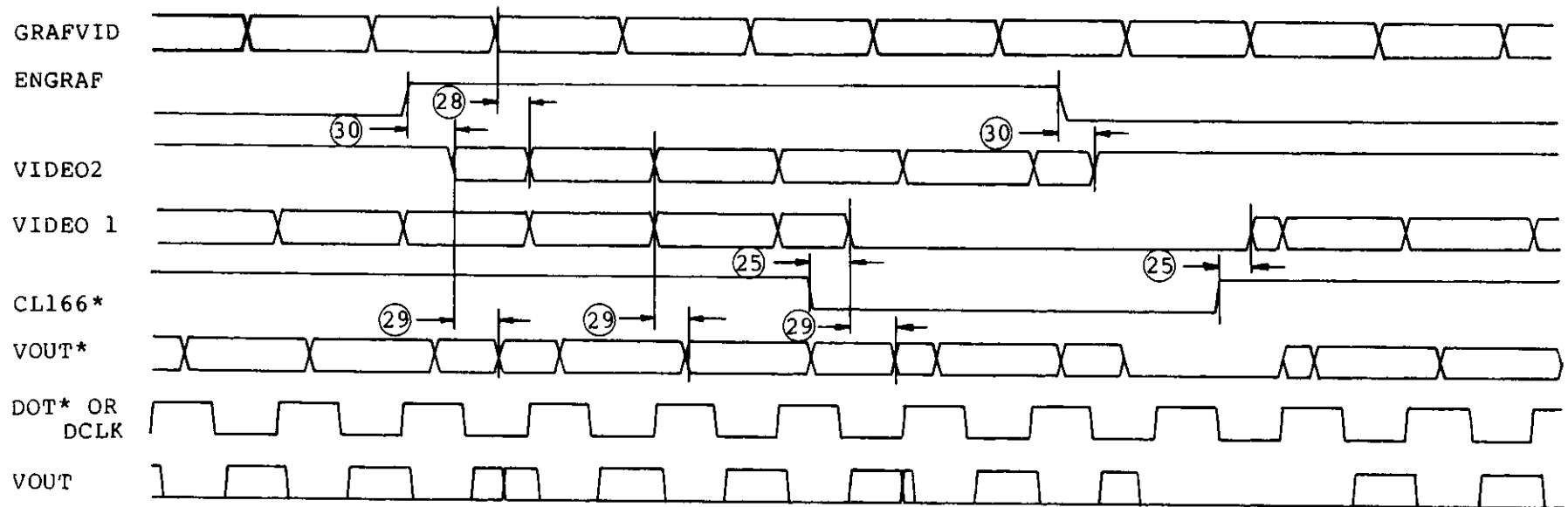
INVERSE VIDEO TIMING



BLANKING

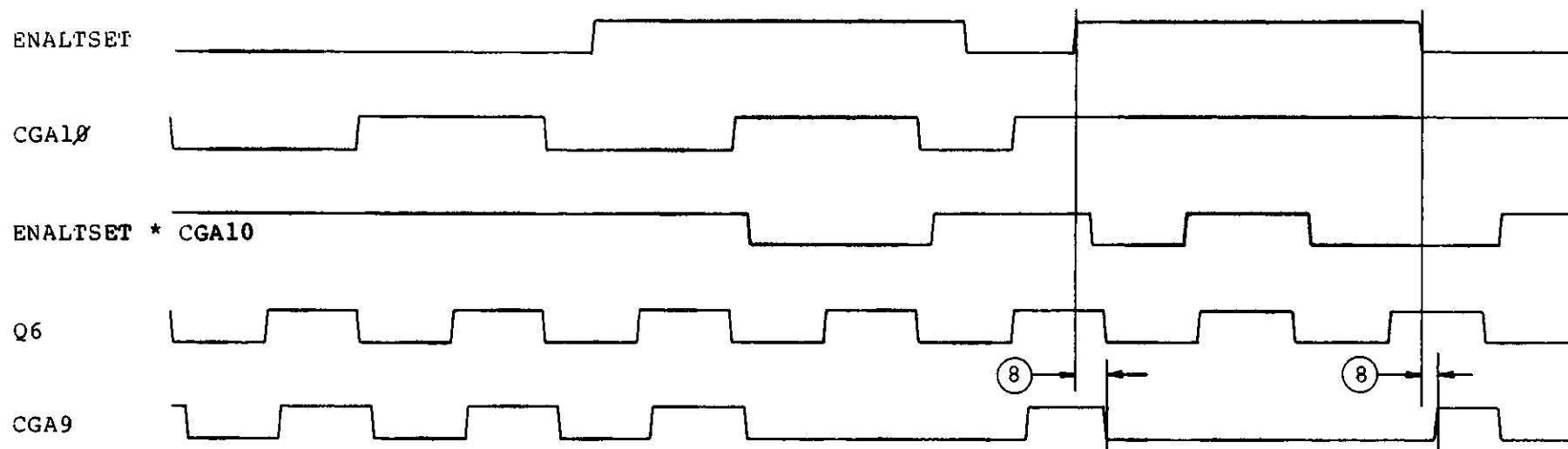


DLYCHAR* & DLYGRAPHIC* CONTROL



GRAFVID, ENGRAF, CL166*, VIDEO1

RELATIONSHIP



ENALTSET	CGA10	Q6	CGA9
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	0
1	1	1	1

ENALTSET CONTROL

DC CHARACTERISTICS (ALL PINS) 0° – 70° C

PARAMETER	MIN.	TYP.	MAX.	UNITS
Input Voltage Level (High)	2.0			V
Input Voltage Level (Low)			.8	V
Output Voltage Level (High)	2.7	3.5		V
Output Voltage Level (Low)		.35	.5	V
Input Current Level (High)			20	μa
Input Current Level (Low)			–.4	ma
Output Current Level (High)	–200			μa
Output Current Level (Low)	4			ma

<u>PIN</u>	<u>SIGNAL</u>	<u>MAX. CAPACITANCE</u>
4	CGA10	35 pf
3	CGA9	35 pf
2	CGA8	35 pf
1	CGA7	35 pf
39	CGA6	35 pf
38	CGA5	35 pf
37	CGA4	35 pf
36	CGA3	35 pf
13	DLYCHAR*	35 pf
14	DLYCHAR	35 pf
19	VOUT*	35 pf

ARRAY #: 4.4.0

CIRCUIT NAME: Floppy Disk Support

NO. OF PINS: 24

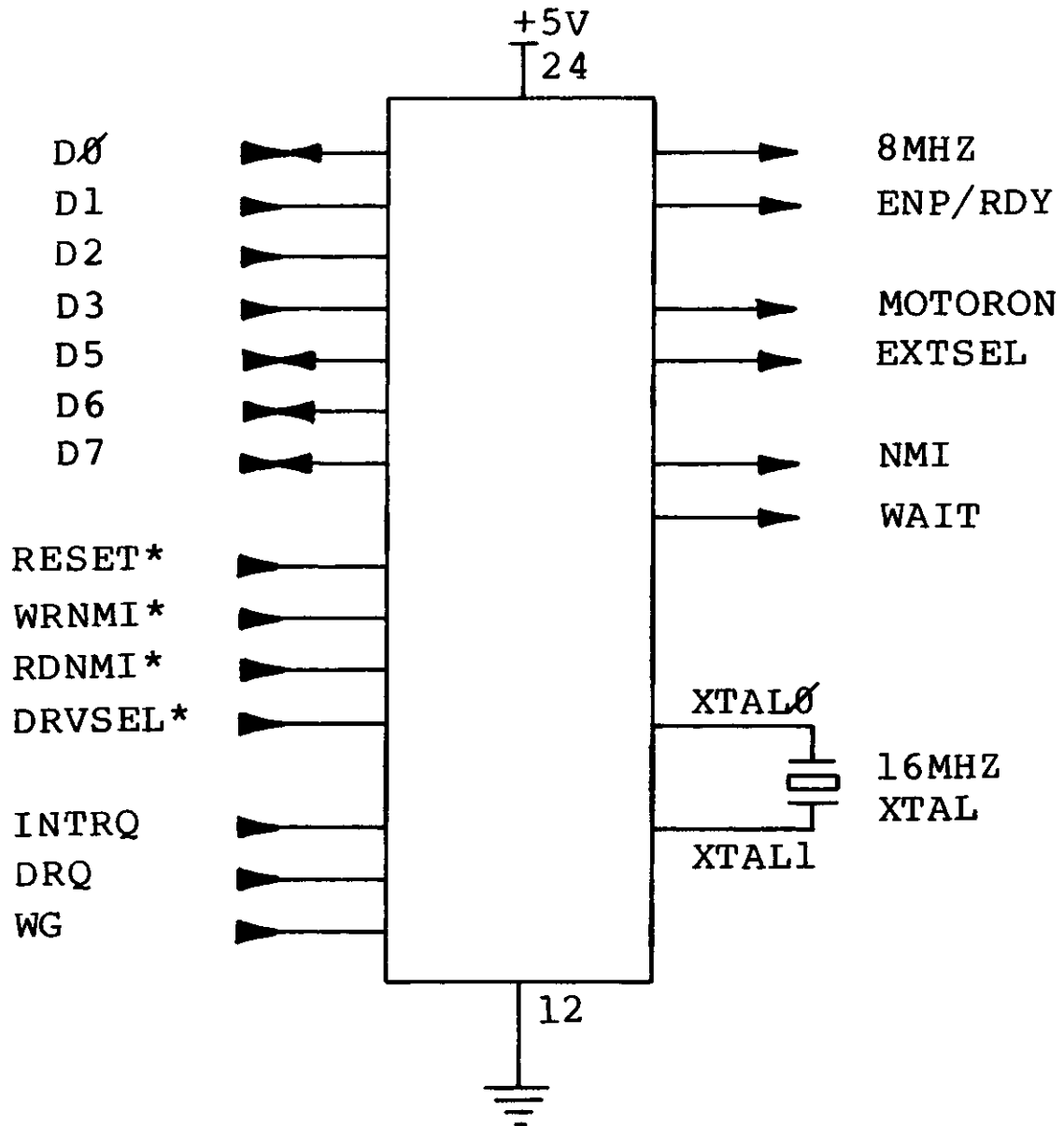
MAX. CLOCK FREQ.: 8 MHz

MAX. PROP. DELAY THROUGHPUT: 75 ns

OPER. TEMP: 0° C to 70° C

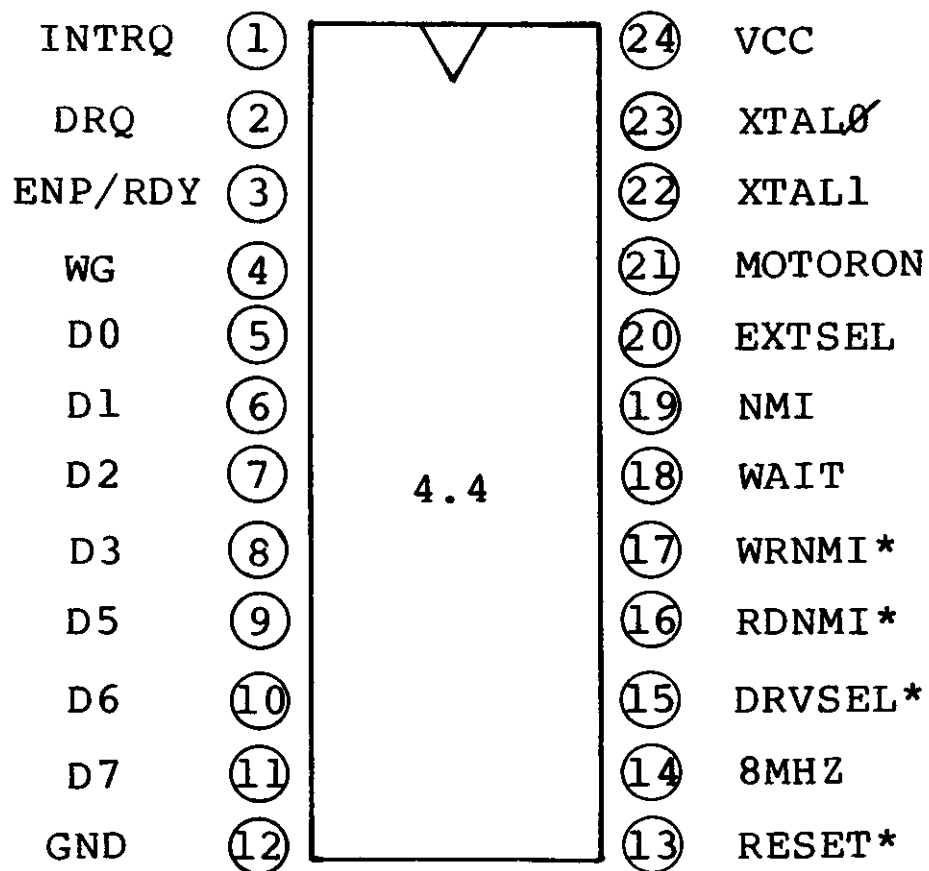
OPERATING VOLTAGE & RANGE: 5 V \pm 5%

4.4.0



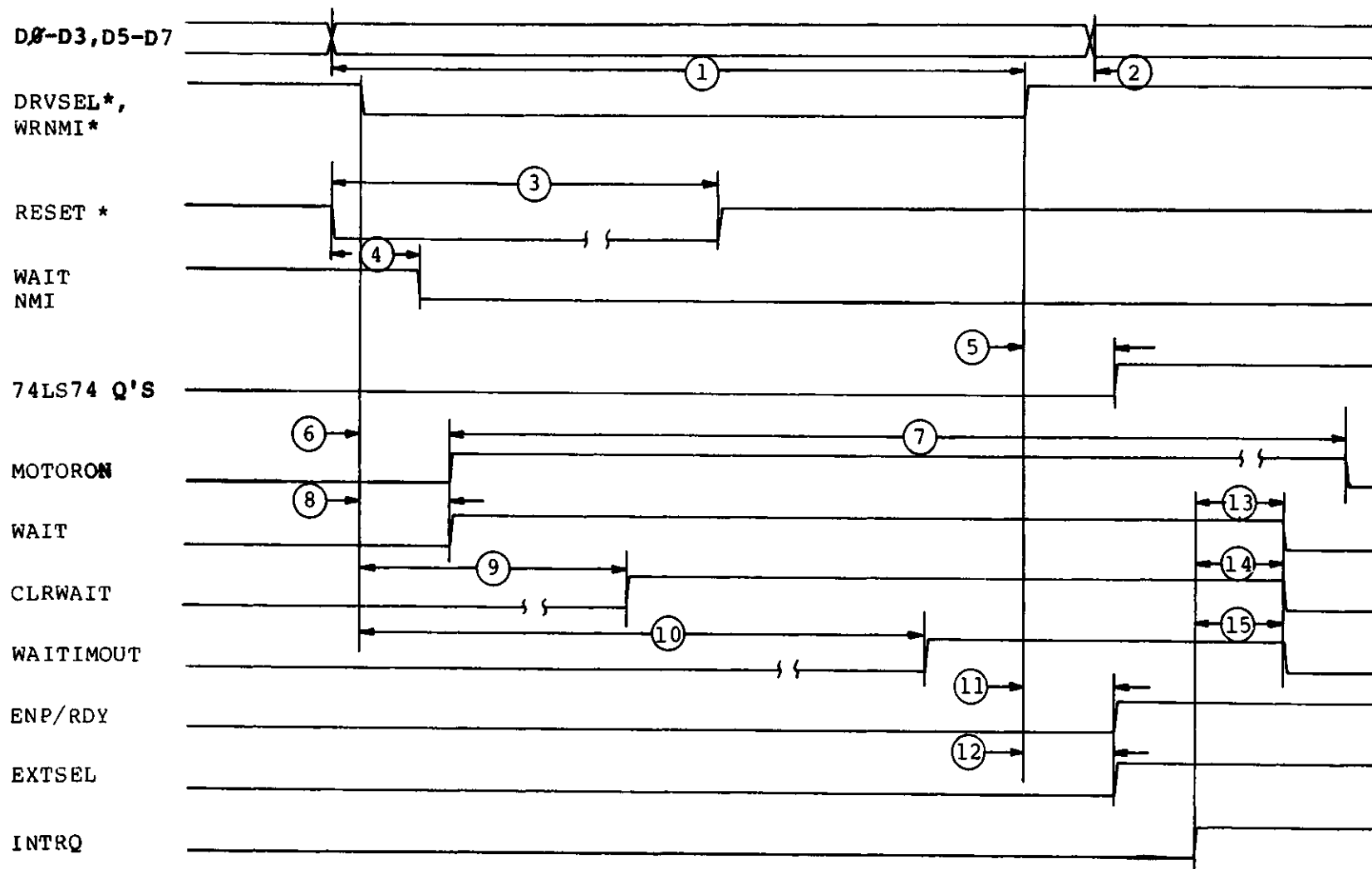
24 PIN CHIP

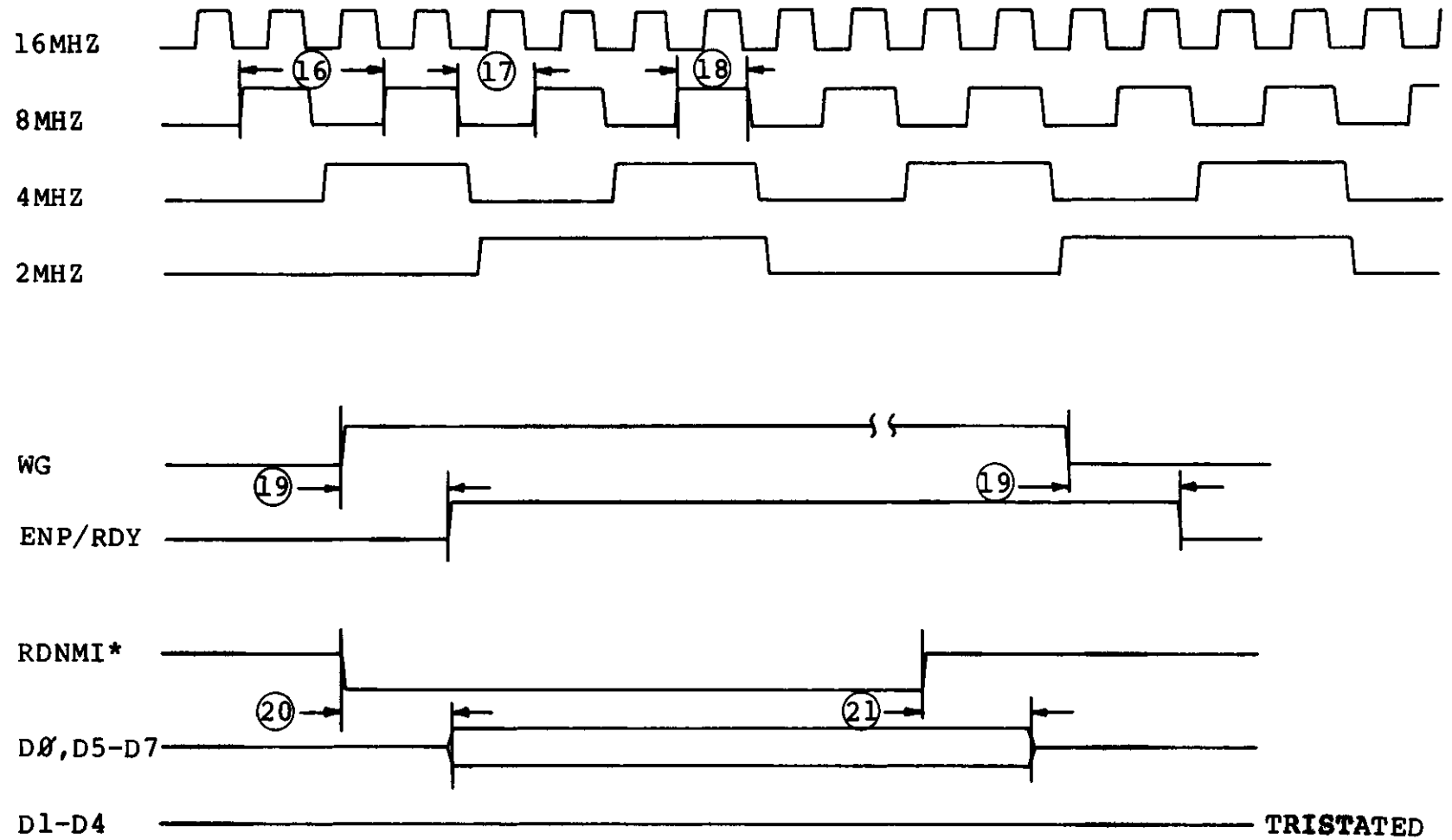
FLOPPY DISK SUPPORT



	PARAMETER	MIN	TYP	SPEC.	MAX	UNITS
1.	Data Setup Time	560				ns
2.	Data Hold Time	50				ns
3.	Reset* Pulse Width		70		100	μs
4.	Reset* ↓ to Wait or NMI ↓				75	ns
5.	WRNMI* ↑ to 74LS74 Q's Outputs ↓↑				75	ns
6.	DRVSEL* ↓ to MOTORON ↑				75	ns
*7.	MOTORON Pulse Width (Low)	3	4		5	sec.
8.	DRVSEL* ↓ to WAIT ↑				75	ns
9.	DRVSEL* ↓ to CLRWAIT ↑	500			1100	ns
10.	DRVSEL* ↓ to WAITIMOUT ↑	1024			1050	μs
11.	DRVSEL* ↑ to ENP/RDY ↑↓				75	ns
12.	DRVSEL* ↑ to EXTSEL ↑↓				75	ns
13.	INTRQ ↑ or DRQ ↑ to WAIT ↓				75	ns
14.	INTRQ ↑ or DRQ ↑ to CLRWAIT ↓				75	ns
15.	INTRQ ↑ or DRQ ↑ to WAITIMOUT ↓				75	ns
16.	8 MHZ Cycle Time		125			ns
17.	8 MHZ Pulse Width (Low)	50	62.5			ns
18.	8 MHZ Pulse Width (High)	50	62.5			ns
19.	WG ↑↓ to ENP/RDY ↑↓				75	ns
20.	RDNMI* ↓ to D0, D5-D7 Valid				75	ns
21.	RDMMI* ↑ to D0, D5-D7 Tristate 0				75	ns

* MOTORON Circuit Must Simulate a Retriggerable Monostable Multivibrator (74LS123)





CAPACITANCE LOAD

OUTPUT	CAPACITANCE MAX.
D0	80 pf
D5	80 pf
D6	80 pf
D7	80 pf
8 MHZ	15 pf
ENP/RDY	15 pf
MOTORON	15 pf
EXTSEL	15 pf
NMI	15 pf
WAIT	15 pf

DC CHARACTERISTICS 0° ~ 70° C

(ALL PINS)

PARAMETER	MIN.	TYP.	MAX.	UNITS
Input Voltage Level (High)	2.0			V
Input Voltage Level (Low)			.8	V
Output Voltage Level (High)	2.7	3.5		V
Output Voltage Level (Low)		.35	.5	V

(ALL PINS EXCEPT MOTORON & D0, D5-D7)

Input Current Level (High)			20	μA
Input Current Level (Low)			-.4	mA
Output Current Level (High)	-160			μA
Output Current Level (Low)	3.2			mA

MOTORON

Output Current Level (High)	-240			μA
Output Current Level (Low)	4.8			mA

D0, D5-D7

Input Current Level (High)			20	μA
Input Current Level (Low)			-.4	mA
Output Current Level (High)	-280			μA
Output Current Level (Low)	5.6			mA

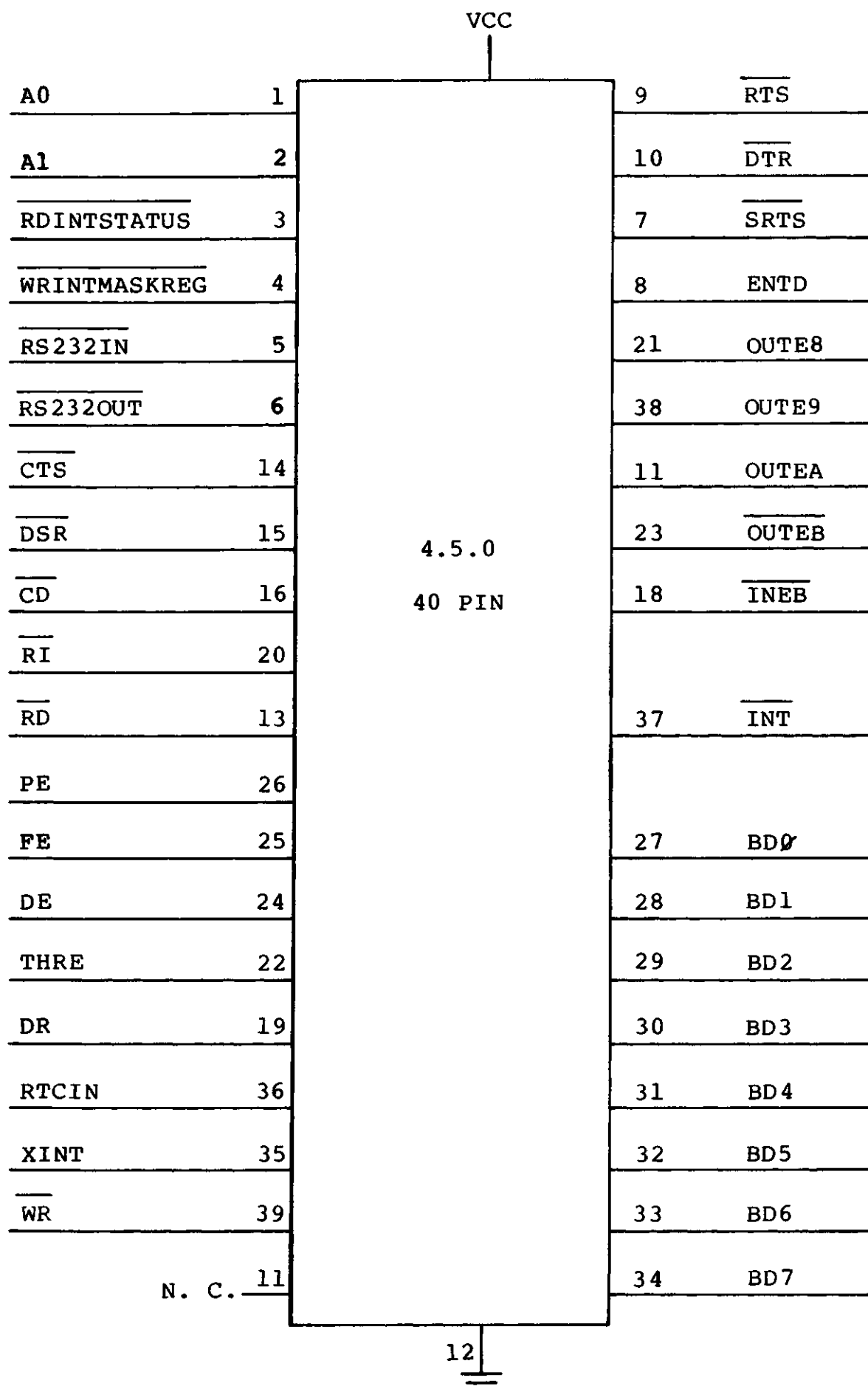
ARRAY #: 4.5.0

CIRCUIT NAME: RS232 Support

NO. OF PINS: 40

OPER. TEMP.: 0° C to 70° C

OPER. VOLTAGE: 5V \pm 5%



D. C. CHARACTERISTICS $0^{\circ} - 70^{\circ} \text{C}$

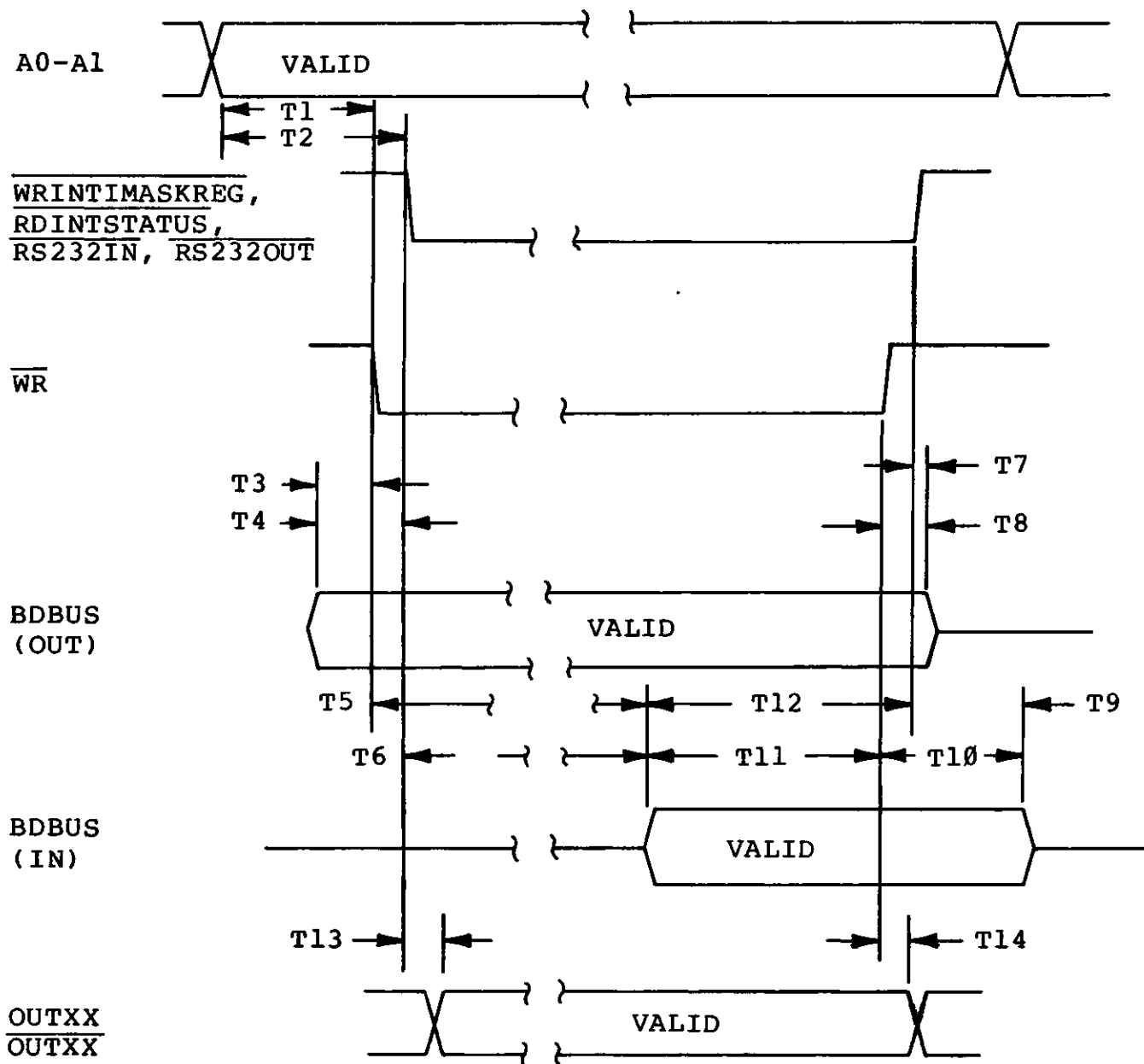
		MIN.	TYP.	MAX.	UNITS
Input Voltage (High)	V_{IH}	2.0			V
Input Voltage (Low)	V_{IL}			.8	V
Output Voltage (High)	V_{OH}	2.7	3.5		V
Output Voltage (Low)	V_{OL}		.35	.5	V
Input Current (High)	I_{IH}			20	μa
Input Current (Low)	I_{IL}			-.4	ma
Output Current (High)	I_{OH} (all except $\overline{\text{INT}}$, $\overline{\text{INEB}}$ & BD)	120			μa
	$\overline{\text{INT}}$ (O.C. or D.D.)	120			μa
	BD BUS	280			μa
	$\overline{\text{INEB}}$	120			μa
Output Current (Low)	I_{OL} (all except $\overline{\text{INT}}$, $\overline{\text{INEB}}$, & BD)	-3.2			ma
	$\overline{\text{INT}}$, (O. C. or D.D.)	-8.0			ma
	BD BUS	-5.6			ma
	$\overline{\text{INEB}}$	-4.4			ma

PROP, DELAY & TIMING	MIN.	TYP.	MAX.
Data In* to BD Bus			75
$\overline{\text{RS232 IN}} \downarrow$ to BD Bus			75
BD Bus Set Up to $\overline{\text{WR}} \uparrow$	75		
BD Bus Hold Time From $\overline{\text{WR}} \uparrow$			60
A0, A1 to $\overline{\text{INEB}}$, OUTE8, OUTE9, OUTEA, $\overline{\text{OUTEB}}$			75
$\overline{\text{RS232IN}}$, $\overline{\text{RS232OUT}} \downarrow$ to $\overline{\text{INEB}}$ OUTE8, OUTE9, OUTEA, $\overline{\text{OUTEB}}$			75
$\overline{\text{WR}} \uparrow$ to OUTE8, OUTE9, OUTEA, $\overline{\text{OUTEB}}$ (WOULD LIKE 18)			32
$\overline{\text{RS232OUT}} \downarrow$ to $\overline{\text{RTS}}$, $\overline{\text{DTR}}$, $\overline{\text{ENTD}}$, $\overline{\text{SRTS}}$			75
PE, FE, DE, THRE, DR, RTCIN, XINT to $\overline{\text{INT}} \downarrow$			75

All Delay In NSEC.

*Data in is any of the following inputs: PE, FE, DE, THRE, DR, RTCIN, XINT, $\overline{\text{CTS}}$, $\overline{\text{DSR}}$, $\overline{\text{CD}}$, $\overline{\text{RI}}$ & $\overline{\text{RD}}$.

C_{OUT} Max = 100 pf for BD Bus, $\overline{\text{INT}}$, & $\overline{\text{INEB}}$; all others C_{OUT} Max = 50 pf.



	MIN.	TYP.	MAX.
t ₁	168		
t ₂	168		
t ₃	-34		0
t ₄	-34		0
t ₅			75
t ₆			75
t ₇			34
t ₈			60
t ₉	24		250
t ₁₀	24		250
t ₁₁	75		
t ₁₂	75		
t ₁₃			75
t ₁₄			32

(Need 18)

All Timing in NSEC.

Index

Subject	Page	Subject	Page
Address decoding		CRT	
4	3	4	7, 9
4 Gate Array	26	4 Gate Array	21, 28, 36
4P	60	4P	57, 60, 85
4P Gate Array	105	4P Gate Array	103, 105, 130
Baud		Decoding, address	
4	15	4	3
4 Gate Array	51	4 Gate Array	28
4P	98	4P	60
4P Gate Array	142	4P Gate Array	105
Baud rate generator		Disk Drive	
4	15	4 Gate Array	48
4 Gate Array	51	4P	93
4P	57, 98	4P Gate Array	142
4P Gate Array	142	Drive select	
Buffering		4	17
4 Gate Array	48	4 Gate Array	47, 48
4P	95	4P	93, 95
4P Gate Array	140	4P Gate Array	142
CASIN*		DRVSEL*	
4	3, 17	4	17
4 Gate Array	54	4 Gate Array	47
CASOUT*		4P	95
4	3, 16	4P Gate Array	140
4 Gate Array	54	FDC Controller	
Cassette circuitry		4 Gate Array	47, 48
4	9	4P	93
4 Gate Array	46	4P Gate Array	138
Clock		I/O bus	
4	3	4	14
4 Gate Array	21	4 Gate Array	44
4P	57	4P	91
4P Gate Array	103	4P Gate Array	136
Compensated write data		Interrupts	
4	17	4 Gate Array	48
4 Gate Array	47	4P	95
4P	96	4P Gate Array	140
4P Gate Array	141	Keyboard	
Controller, CRT		4	7
4	7	4 Gate Array	41
4 Gate Array	21, 28, 36	4P	87
4P	57, 60, 85	4P Gate Array	132
4P Gate Array	103, 105, 130	Memory address decoding	
Controller, Floppy Disk		4	6
4 Gate Array	48	4 Gate Array	27
4P	93	4P	60
4P Gate Array	138	4P Gate Array	105
CPU Board		MODOUT	
4	3	4	16
4 Gate Array	21	4P	82
4P	57	4P Gate Array	127
4P Gate Array	103		

Index

Subject	Page	Subject	Page
NMI logic		RS-232 Board	
4 Gate Array	48	4 Gate Array	51
4P	95	4P	98
4P Gate Array	140	4P Gate Array	142
Oscillator		Sound	
4	3, 5	4	10
PAL Circuits		4 Gate Array	44
4	15	4P	91
Port Address decoding		4P Gate Array	136
4	15	Timing, CPU	
4P	81	4	3
4P Gate Array	126	4 Gate Array	21
Port bit map		4P	57
4	6, 16	4P Gate Array	103
4 Gate Array	37	Video Controller	
4P	81	4	7
4P Gate Array	126	4 Gate Array	21, 28, 36
Precompensation, write		4P	57, 60, 85
4 Gate Array	47	4P Gate Array	103, 105, 130
4P	96	Video Monitor	
4P Gate Array	141	4	7, 9
Printer status		4 Gate Array	21, 28, 36
4	9	4P	57, 60, 85
4 Gate Array	41	4P Gate Array	103, 105, 130
4P	87	Wait State	
4P Gate Array	132	4 Gate Array	47
RAM		4P	95
4	7, 8	4P Gate Array	140
4 Gate Array	36, 39	WRINTMASKREG*	
4P	71-84	4	16
4P Gate Array	116-129	4 Gate Array	48
RDINSTAUS*		4P	83
4	16	4P Gate Array	128
4 Gate Array	48	Write Precompensation	
4P	83	4 Gate Array	47
4P Gate Array	128	4P	96
RDNMISTATUS*		4P Gate Array	141
4	16	WRNMIMASKREG*	
4 Gate Array	48	4	16
4P	83	4 Gate Array	48
4P Gate Array	128	4P	83
Real Time Clock		4P Gate Array	128
4	9		
4 Gate Array	41		
4P	87		
4P Gate Array	132		
ROM			
4	7		
4 Gate Array	36		
4P	60		
4P Gate Array	105		

**MOTOROLA**

SEMICONDUCTORS

3501 ED BLUESTEIN BLVD., AUSTIN, TEXAS 78721

Advance Information

CRT CONTROLLER (CRTC)

The MC6835 is a ROM based CRT Controller which interfaces an MPU system to a raster scan CRT display. It is intended for use in MPU based controllers for CRT terminals in stand-alone or cluster configurations. The MC6835 supports two selectable mask programmed screen formats using the program select input (PROG).

The CRTC is optimized for the hardware/software balance required for maximum flexibility. All keyboard functions, reads, writes, cursor movements, scrolling, and editing are under processor control. The mask programmed registers of the CRTC are programmed to control the video format and timing.

- Cost Effective ROM Based CRTC Which Supports Two Screen Formats
- Useful in Monochrome or Color CRT Applications
- Applications Include "Glass-Teletype," Smart, Programmable, Intelligent CRT Terminals; Video Games; Information Displays
- Alphanumeric, Semigraphic, and Full Graphic Capability
- Timing May Be Generated for Almost Any Alphanumeric Screen Format, e.g., 80×24, 72×64, 132×20
- Single +5 Volt Supply
- M6800 Compatible Bus Interface
- TTL-Compatible Inputs and Outputs
- Start Address Register Provides Hardware Scroll (By Page, Line, or Character)
- Programmable Cursor Register Allows Control of Cursor Position
- Refresh (Screen) Memory May Be Multiplexed Between the CRTC and the MPU Thus Removing the Requirements for Line Buffers or External DMA Devices
- Mask Programmable Interlace or Non-Interlace Scan Modes
- 14-Bit Refresh Address Allows Up to 16K of Refresh Memory for Use in Character or Semigraphic Displays
- 5-Bit Row Address Allows up to 32 Scan-Line Character Blocks
- By Utilizing Both the Refresh Addresses and the Row Addresses, a 512K Address Space is Available for Use in Graphics Systems
- Refresh Addresses are Provided During Retrace, Allowing the CRTC to provide Row Addresses to Refresh Dynamic RAMs
- Pin Compatible with the MC6845. The MC6845 May Be Used as a Prototype Part to Emulate the MC6835.

MAXIMUM RATINGS

Rating	Symbol	Value	Unit
Supply Voltage	V_{CC}^*	-0.3 to +7.0	V
Input Voltage	V_{in}^*	-0.3 to +7.0	V
Operating Temperature Range MC6835, MC68A35, MC68B35 MC6835C, MC68A35C, MC68B35C	T_A	0 to +70 -50 to +85	°C
Storage Temperature Range	T_{stg}	-55 to +150	°C

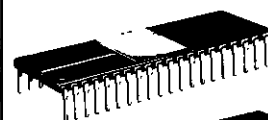
*With respect to GND (V_{SS}).

MC6835

MOS

(HIGH-DENSITY, N-CHANNEL,
SILICON-GATE DEPLETION LOAD)

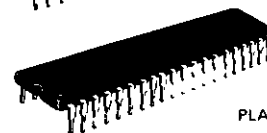
MASK PROGRAMMED CRT CONTROLLER (CRTC)



L SUFFIX
CERAMIC PACKAGE
CASE 715



S SUFFIX
CERDIP PACKAGE
CASE 734



P SUFFIX
PLASTIC PACKAGE
CASE 711

PIN ASSIGNMENT

GND	1	40	VS
RESET	2	39	HS
PROG	3	38	RA0
MA0	4	37	RA1
MA1	5	36	RA2
MA2	6	35	RA3
MA3	7	34	RA4
MA4	8	33	D0
MA5	9	32	D1
MA6	10	31	D2
MA7	11	30	D3
MA8	12	29	D4
MA9	13	28	D5
MA10	14	27	D6
MA11	15	26	D7
MA12	16	25	CS
MA13	17	24	RS
DE	18	23	E
CURSOR	19	22	W
VCC	20	21	CLK

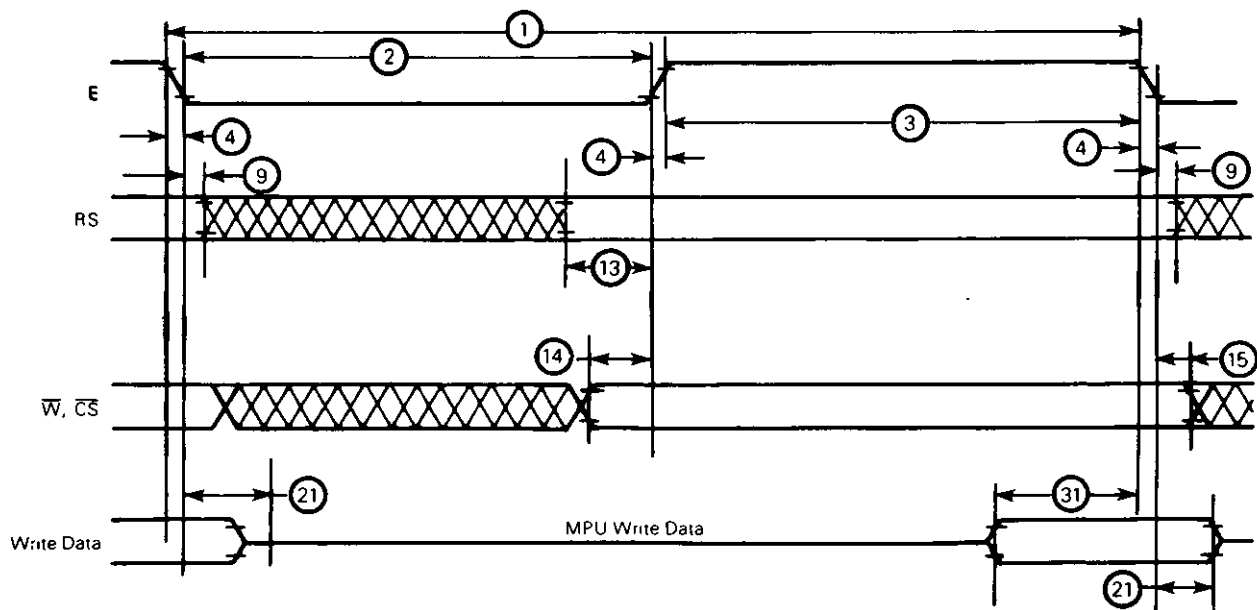
DC ELECTRICAL CHARACTERISTICS ($V_{CC} = 5.0 \text{ Vdc} \pm 5\%$, $V_{SS} = 0$, $T_A = 0 \text{ to } 70^\circ\text{C}$ unless otherwise noted) (Reference Figures 2-4)

Characteristic	Symbol	Min	Typ	Max	Unit
Input High Voltage	V_{IH}	2.0	—	V_{CC}	V
Input Low Voltage	V_{IL}	—0.3	—	0.8	V
Input Leakage Current	I_{in}	—	0.1	2.5	μA
Hi Z (Off State) Input Current ($V_{CC} = 5.25 \text{ V}$) ($V_{in} = 0.4 \text{ to } 2.4 \text{ V}$)	I_{TSI}	—10	—	10	μA
Output High Voltage ($I_{Load} = -100 \mu\text{A}$)		2.4	3.0	—	V
Output Low Voltage ($I_{Load} = 1.6 \text{ mA}$)	V_{OL}	—	0.3	0.4	V
Internal Power Dissipation (Measured at $T_A = 0^\circ\text{C}$)	P_D	—	150	300	mW
Input Capacitance	C_{in}	—	—	12.5	pF
		—	—	10	pF
Output Capacitance	C_{out}	—	—	10	pF

BUS TIMING CHARACTERISTICS (Reference Figures 2 and 3)

Ident Number	Characteristics	Symbol	MC6835		MC68A35		MC68B35		Unit
			Min	Max	Min	Max	Min	Max	
1	Cycle Time	t_{cyc}	10	10	0.67	10	0.5	10	μs
2	Pulse Width, E Low	PW_{EL}	430	—	280	—	210	—	ns
3	Pulse Width, E High	PW_{EH}	450	—	280	—	220	—	ns
4	Clock Transition Time	t_r, t_f	—	25	—	25	—	20	ns
9	Address Hold Time (RS)	t_{AH}	10	—	10	—	10	—	ns
13	RS Setup Before E	t_{AS}	80	—	60	—	40	—	ns
14	\overline{W} and \overline{CS} Setup Before E	t_{CS}	80	—	60	—	40	—	ns
15	Hold Time for \overline{W} and \overline{CS}	t_{CH}	10	—	10	—	10	—	ns
21	Write Data Hold Time Required	t_{DHW}	10	—	10	—	10	—	ns
31	Peripheral Input Data Setup	t_{DSW}	165	—	80	—	60	—	ns

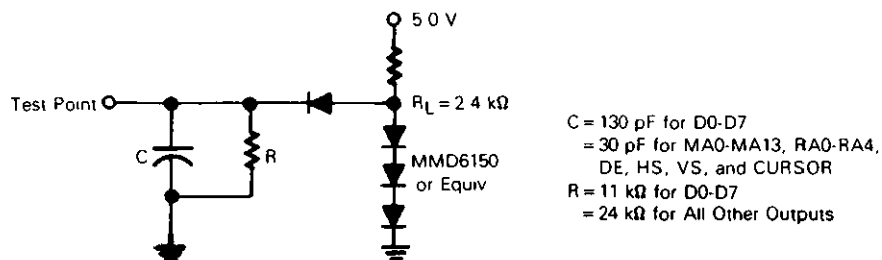
FIGURE 2 — MC6835 BUS TIMING

**NOTES**

- 1 Voltage levels shown are $V_L \leq 0.4 \text{ V}$, $V_H \geq 2.4 \text{ V}$ unless otherwise noted
- 2 Measurement points shown are 0.8 V and 2.0 V unless otherwise noted

**MOTOROLA Semiconductor Products Inc.**

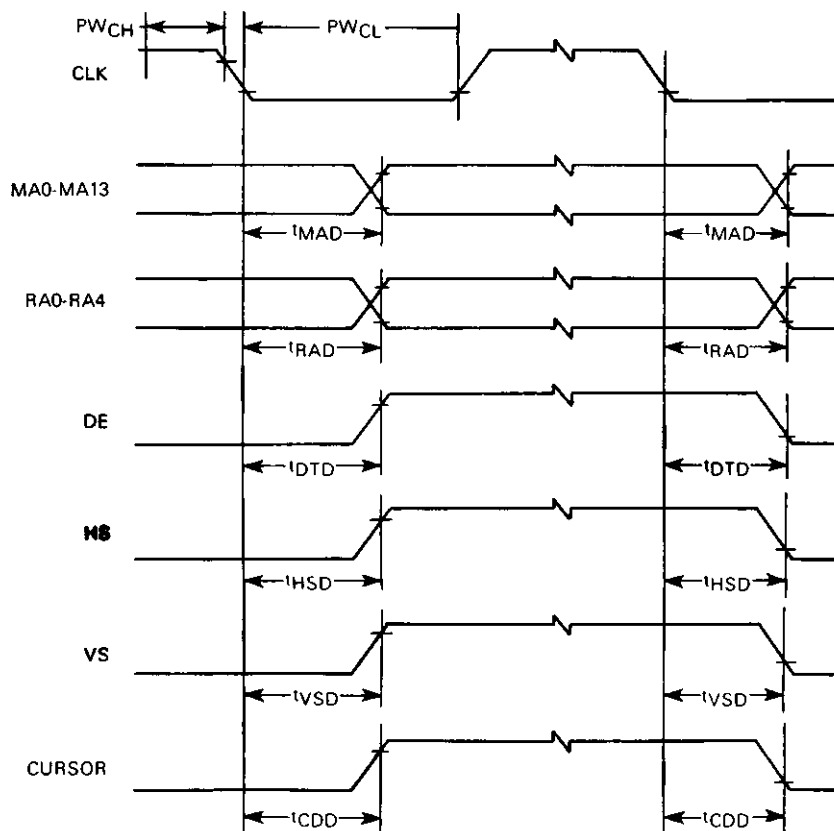
FIGURE 3 — BUS TIMING TEST LOAD



CRTC TIMING CHARACTERISTICS (See Figure 4)

Characteristics	Symbol	MC6835		MC68A35		MC68B35		Unit
		Min	Max	Min	Max	Min	Max	
Minimum Clock Pulse Width, Low	PW _{CL}	150	—	140	—	130	—	ns
Minimum Clock Pulse Width, High	PW _{CH}	150	—	140	—	130	—	ns
Clock Frequency	f_c	330	—	300	—	270	—	ns
Rise and Fall Time for Clock Input	t_r, t_f	—	20	—	20	—	20	ns
Memory Address Delay Time	t_{MAD}	—	160	—	160	—	160	ns
Raster Address Delay Time	t_{RAD}	—	160	—	160	—	160	ns
Display Timing Delay Time	t_{DTD}	—	250	—	250	—	200	ns
Horizontal Sync Delay Time	t_{HSD}	—	250	—	250	—	200	ns
Vertical Sync Delay Time	t_{VSD}	—	250	—	250	—	200	ns
Cursor Display Timing Delay Time	t_{CDD}	—	250	—	250	—	200	ns

FIGURE 4 — CRTC TIMING CHART



NOTE: Timing measurements are referenced to and from a low voltage of 0.8 volts and a high voltage of 2.0 volts unless otherwise noted.



MOTOROLA Semiconductor Products Inc.

CRTC INTERFACE SYSTEM DESCRIPTION

The MC6835 CRT Controller generates the signals necessary to interface a digital system to a raster scan CRT display. In this type of display, an electron beam starts in the upper left hand corner, moves quickly across the screen and returns. This action is called a horizontal scan. After each horizontal scan the beam is incrementally moved down in the vertical direction until it has reached the bottom. At this point one frame has been displayed, as the beam has made many horizontal scans and one vertical scan.

Two types of raster scanning are used in CRTs, interlace and non-interlace, shown in Figures 5 and 6. Non-interlacing scanning consists of one field per frame. The scan lines in Figure 5 are shown as solid lines and the retrace patterns are indicated by the dotted lines. Increasing the number of frames per second will decrease the flicker. Ordinarily, either a 50 or 60 frame per second refresh rate is used to minimize beating between the frequency of the CRT horizontal oscillator and the power line frequency. This prevents the displayed data from weaving or swimming.

Interlace scanning is used in broadcast TV and on data monitors where high density or high resolution data must be displayed. Two fields, or vertical scans are made down the screen for each single picture or frame. The first field (Even

field) starts in the upper left hand corner, the second (Odd field) in the upper center. Both fields overlap as shown in Figure 6, thus interlacing the two fields into a single frame.

In order to display the characters on the CRT screen the frames must be continually repeated. The data to be displayed is stored in the Refresh (Screen) memory by the MPU controlling the data processing system. The data is usually written in ASCII code, so it cannot be directly displayed as characters. A Character Generator ROM is typically used to convert the ASCII codes into the "dot" pattern for every character.

The most common method of generating characters is to create a matrix of "x" dots (columns) wide and "y" dots (rows) high. Each character is created by selectively filling in the dots. As "x" and "y" get larger a more detailed character may be created. Two common dot matrices are 5×7 and 7×9 . Many variations of these standards will allow Chinese, Japanese, or Arabic letters instead of English. Since characters require some space between them, a character block larger than the character is typically used as shown in Figure 7. The figure also shows the corresponding timing and levels for a video signal that would generate the characters.

FIGURE 5 — RASTER SCAN SYSTEM (NON-INTERLACE)

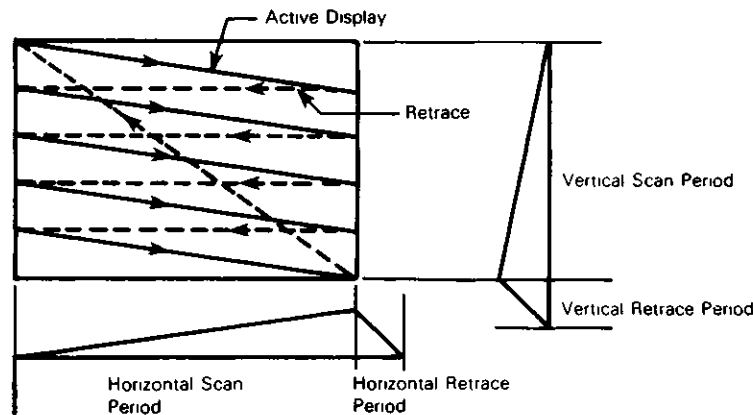


FIGURE 6 — RASTER SCAN SYSTEM (INTERLACE)

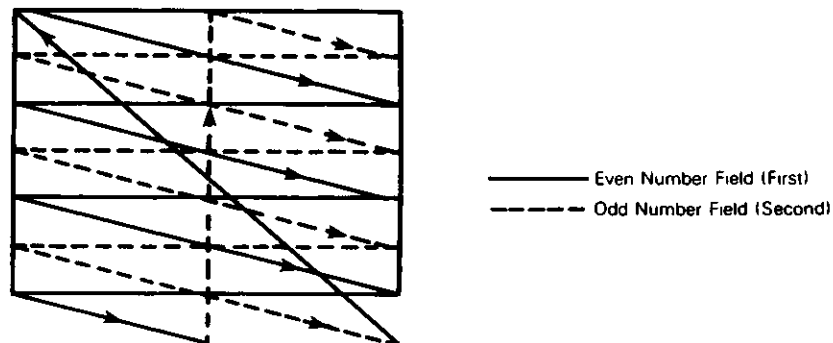
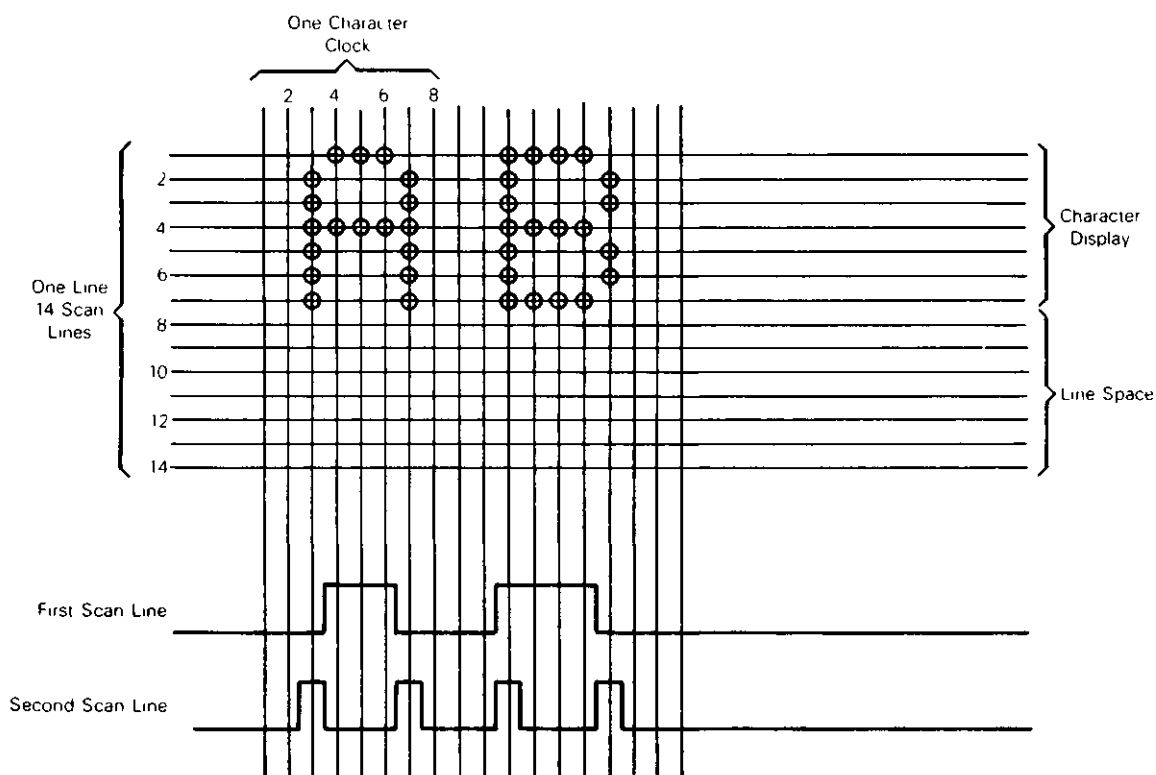

MOTOROLA Semiconductor Products Inc.

FIGURE 7 — CHARACTER DISPLAY ON THE SCREEN AND VIDEO SIGNAL



Referring to Figure 1, the MC6835 CRT controller generates the Refresh addresses (MA0-MA13), row addresses (RA0-RA4), and the video timing (vertical sync — VS, horizontal sync — HS and display enable — DE). Other functions include an internal cursor register which generates a Cursor output when its contents compare to the current Refresh address. A select input, PROG, allows selection of one of two mask programmed video formats (e.g., for 50 Hz and 60 Hz compatibility).

All timing in the CRTC is derived from the CLK input. In alphanumeric terminals, this signal is the character rate. The video rate or "dot" clock is externally divided by high speed logic (TTL) to generate the CLK signal. The high speed logic must also generate the timing and control signals necessary for the Shift Register, Latch and MUX Control shown in Figure 1.

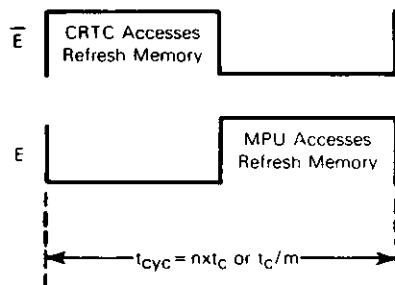
The processor communicates with the CRTC through an 8-bit data bus by writing into the five user programmable registers of the MC6835.

The Refresh memory address is multiplexed between the processor and the CRTC. Data appears on a secondary bus separate from the processor's bus. The secondary data bus concept in no way precludes using the Refresh RAM for other purposes. It looks like any other RAM to the processor. A number of approaches are possible for solving contentions for the Refresh memory.

- 1 Processor always gets priority (Generally, "hash" occurs as MPU and CRTC clocks are not synchronized.)

- 2 Processor gets priority access anytime, but can be synchronized by an interrupt to perform accesses only during horizontal and vertical retrace times.
- 3 Synchronize the processor with memory wait cycles (states).
- 4 Synchronize the processor to the character rate as shown in Figure 8. The M6800 processor family works very well in this configuration as constant cycle lengths are present. This method provides no overhead for the processor as there is never a contention for a memory access. All accesses are transparent.

FIGURE 8 — TRANSPARENT REFRESH MEMORY CONFIGURATION TIMING USING M6800 FAMILY MPU



Where m, n are integers, t_c is character period



MOTOROLA Semiconductor Products Inc.

PIN DESCRIPTION

PROCESSOR INTERFACE

The CRTC interfaces to a processor bus on the data bus (D0-D7) using \overline{CS} , RS, E, and \overline{W} for control signals

Data Bus (D0-D7) — The data lines (D0-D7) comprise the write only data bus

Enable (E) — The Enable signal is a high-impedance TTL/MOS-compatible input which enables the data bus input/output buffers and clocks data to the CRTC. This signal is usually derived from the processor clock. The high to low transition is the active edge

Chip Select (\overline{CS}) — The \overline{CS} line is an active-low high-impedance TTL/MOS-compatible input which selects the CRTC write to the internal register file. This signal should only be active when there is a valid stable address being decoded from the processor

Register Select (RS) — The RS line is a high-impedance TTL/MOS-compatible input which selects either the Address Register (RS="0") or one of the Data Registers (RS="1") of the internal register file when \overline{CS} is low

Write (\overline{W}) — The \overline{W} line is a high-impedance TTL/MOS-compatible input which determines whether the internal register file gets written. A write is defined as a low level

CRTC CONTROL

The CRTC provides horizontal sync (HS), vertical sync (VS), and display enable (DE) signals

NOTE — Care should be exercised when interfacing to CRT monitors as many monitors claiming to be "TTL compatible," have transistor input circuits which require the CRTC or TTL devices buffering signals from the CRTC/video circuits to exceed the maximum rated drive currents

Vertical Sync (VS) and Horizontal Sync (HS) — These TTL-compatible outputs are active-high signals which drive the monitor directly or are fed to the video processing circuitry to generate a composite video signal. The VS signal determines the vertical position of the displayed text while the HS signal determines the horizontal position of the displayed text

Display Enable (DE) — This TTL-compatible output is an active-high signal which indicates the CRTC is providing addressing in the active Display Area

REFRESH MEMORY/CHARACTER GENERATOR ADDRESSING

The CRTC provides Memory Addresses (MA0-MA13) to scan the Refresh RAM. Row Addresses (RA0-RA4) are also provided for use with character generator ROMs. In a graphics system both the Memory Addresses and the Row Addresses would be used to scan the Refresh RAM. Both

the Memory Addresses and the Row Addresses continue to run during vertical retrace thus allowing the CRTC to provide the refresh addresses required to refresh dynamic RAMs

Refresh Memory Addresses (MA0-MA13) — These 14 outputs are used to refresh the CRT screen with pages of data located within a 16K block of refresh memory. These outputs are capable of driving one standard TTL load and 30 pF

Row Addresses (RA0-RA4) — These five outputs from the internal Row Address counter are used to address the Character Generator ROM. These outputs are capable of driving one standard TTL load and 30 pF

OTHER PINS

Cursor — This TTL-compatible output indicates a valid Cursor address to external video processing logic. It is an active-high signal

Clock (CLK) — The CLK is a TTL/MOS-compatible input used to synchronize all CRT functions except for the processor interface. An external dot counter is used to derive this signal which is usually the character rate in an alphanumeric CRT. The active transition is high-to-low

Program Select (PROG) — This TTL-compatible input allows selection of one of two sets of mask programmed video formats. Set zero is selected when PROG is low and set one is selected when PROG is high

VCC, GND — These inputs supply +5 Vdc $\pm 5\%$ to the CRTC

RESET — The \overline{RESET} input is used to reset the CRTC. Functionality of \overline{RESET} differs from that of other M6800 parts. \overline{RESET} must remain low for at least one cycle of the character clock (CLK). A low level on the \overline{RESET} input forces the CRTC into the following state

- All counters in the CRTC are cleared and the device stops the display operation
- All the outputs are driven low, except the MA0-MA13 outputs which are driven to the current value in the Start Address Register
- The control registers of the CRTC are not affected and remain unchanged
- The CRTC resumes the display operation immediately after the release of \overline{RESET}

CRTC DESCRIPTION

The CRTC consists of mask-programmable horizontal and vertical timing generators, software-programmable linear address register, mask-programmable cursor logic and control circuitry for interfacing to a M6800 family microprocessor bus

All CRTC timing is derived from CLK, usually the output of an external dot rate counter. Coincidence (CO) circuits continuously compare counter contents to the contents of the



MOTOROLA Semiconductor Products Inc.

TABLE 1 — INTERNAL REGISTER ASSIGNMENT

CS	RS	Address Register						Register #	Register File	Program Unit	Read	Write	Number of Bits							
		4	3	2	1	0							7	6	5	4	3	2	1	0
1	X	X	X	X	X	X	X	X	—	—	—	—								
0	0	X	X	X	X	X	X	AR	Address Register	—	No	Yes								
Note 3								R0	Horizontal Total	Char	No	No								
								R1	Horizontal Displayed	Char	No	No								
								R2	H Sync Position	Char	No	No								
								R3	Sync Width	—	No	No	V	V	V	V	H	H	H	H
								R4	Vertical Total	Char Row	No	No								
								R5	V Total Adjust	Scan Line	No	No								
								R6	Vertical Displayed	Char Row	No	No								
								R7	V Sync Position	Char Row	No	No								
								R8	Interlace Mode and Skew	Note 1	No	No	C	C	D	D			I	I
								R9	Max Scan Line Address	Scan Line	No	No								
								R10	Cursor Start	Scan Line	No	No		B	P					(Note 2)
								R11	Cursor End	Scan Line	No	No								
0	1	0	1	1	0	0		R12	Start Address (H)	—	No	Yes	0	0						
0	1	0	1	1	0	1		R13	Start Address (L)	—	No	Yes								
0	1	0	1	1	1	0		R14	Cursor (H)	—	No	Yes	0	0						
0	1	0	1	1	1	1		R15	Cursor (L)	—	No	Yes								

NOTES

- 1 The Interlace Control is shown in Table 2 while Skew Control is shown in Table 3
- 2 Bit 5 of the Cursor Start Raster Register is used to blink period control, and Bit 6 is used to select blink or non-blink
- 3 R0-R11 are mask-programmable and are not accessible via the data bus

mask programmable register file, R0-R11 For horizontal timing generation, comparisons result in

- 1 Horizontal sync pulse (HS) of a frequency, position and width determined by the register contents
- 2 Horizontal Display signal of a frequency, position and duration determined by the register contents

The horizontal counter produces H clock which drives the Scan Line Counter and Vertical Control. The contents of the Raster Counter are continuously compared to the Max Scan Line Address Register. A coincidence resets the Raster Counter and clocks the Vertical Counter.

Comparisons of Vertical Counter contents and Vertical Registers result in

- 1 Vertical sync pulse (VS) of a frequency, position and width determined by the register contents
- 2 Vertical Display signal of a frequency, position, and duration determined by the register contents

The Vertical Control Logic has other functions

- 1 Generate row selects, RA0-RA4, from the Raster Count for the corresponding interlace or non-interlace modes
- 2 Extend the number of scan lines in the vertical total by the amount programmed in the Vertical Total Adjust Register

The cursor logic determines the size and blink rate of the

cursor as indicated by the register contents

The Linear Address Generator is driven by CLK and locates the relative positions of characters in memory and their positions on the screen. Fourteen outputs, MA0-MA13, are available for addressing up to four pages of 4K characters, eight pages of 2K characters, etc.

Five additional write-only registers define the Start Address and cursor position. Using the Start Address Register, hardware scrolling through 16K characters is possible. The Linear Address Generator repeats the same sequence of addresses for each scan line of a character row. The Start Address Register and the Cursor Position Register are programmed by the processor through the data bus, D0-D7 and the control signals — \bar{W} , \bar{CS} , RS, and E. Refer to Figure 9.

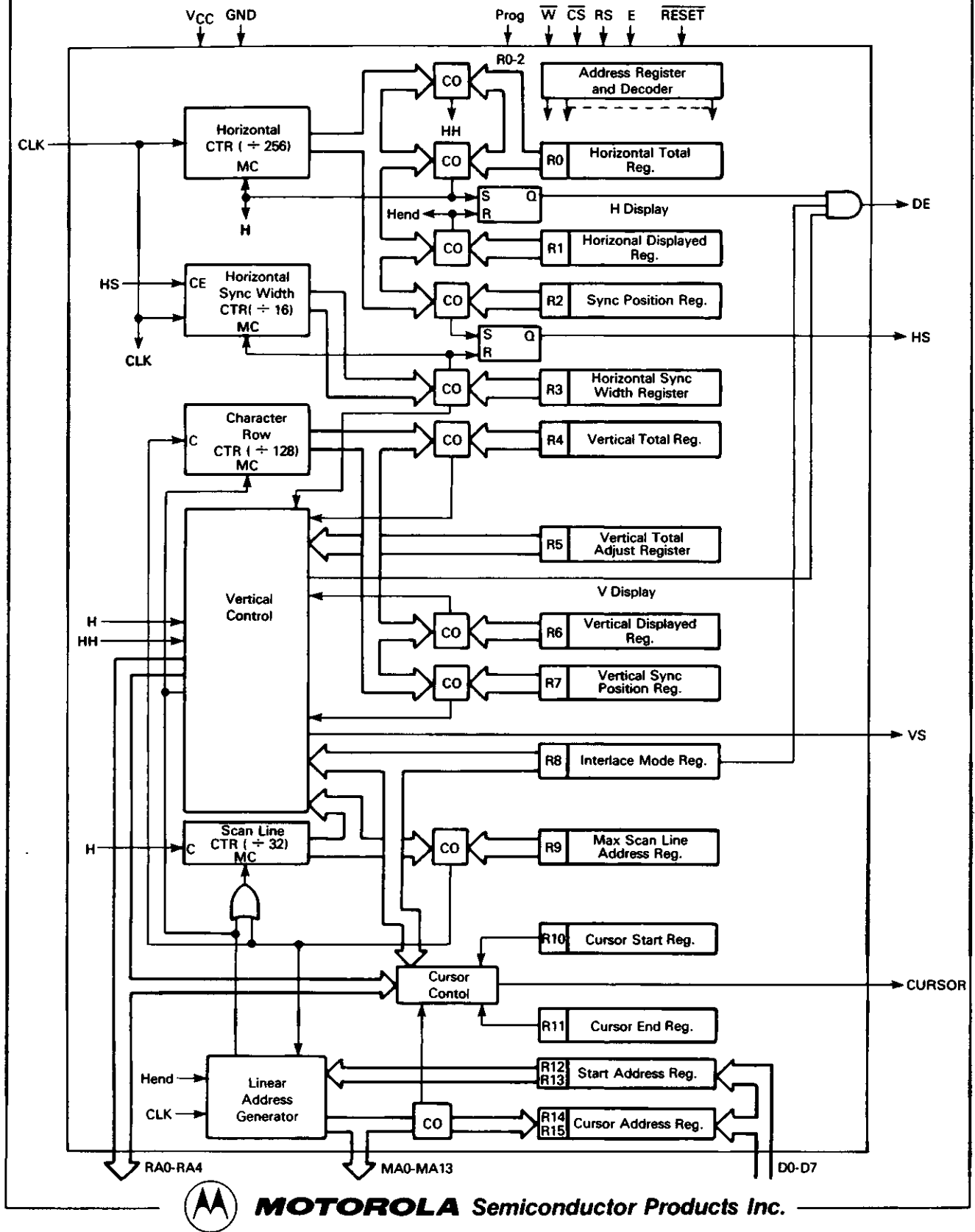
REGISTER FILE DESCRIPTION

The MC6835 has 17 control registers of which 12 are mask programmable. The remaining five registers — Address register, Start Address register pair, and Cursor Position register pair — are write-only registers programmed by the MPU. These registers control horizontal timing, vertical timing, interlace operation, row address operation and define the cursor, cursor address, and start address. The register addresses and sizes are shown in Table 1.



MOTOROLA Semiconductor Products Inc.

FIGURE 9 — CRTC BLOCK DIAGRAM



MASK PROGRAMMABLE REGISTERS R0-R11

The twelve mask programmable registers determine the display format generated by the MC6835. The PROG input is used to select one of two sets of register values.

Figure 10 shows the visible display area of a typical CRT monitor giving the point of reference for horizontal registers as the left most displayed character position. Horizontal registers are programmed in character clock time units with respect to the reference as shown in Figure 11. The point of reference for the vertical registers is the top character position displayed. Vertical registers are programmed in character row times or scan line times as shown in Figure 12.

Horizontal Total Register (R0) — This 8-bit register determines the horizontal sync (HS) frequency by defining the HS period in character times. It is the total of the displayed characters plus the non-displayed character times (retrace) minus one.

Horizontal Displayed Register (R1) — This 8-bit register determines the number of displayed characters per line. Any 8-bit number may be programmed as long as the contents of R0 are greater than the contents of R1.

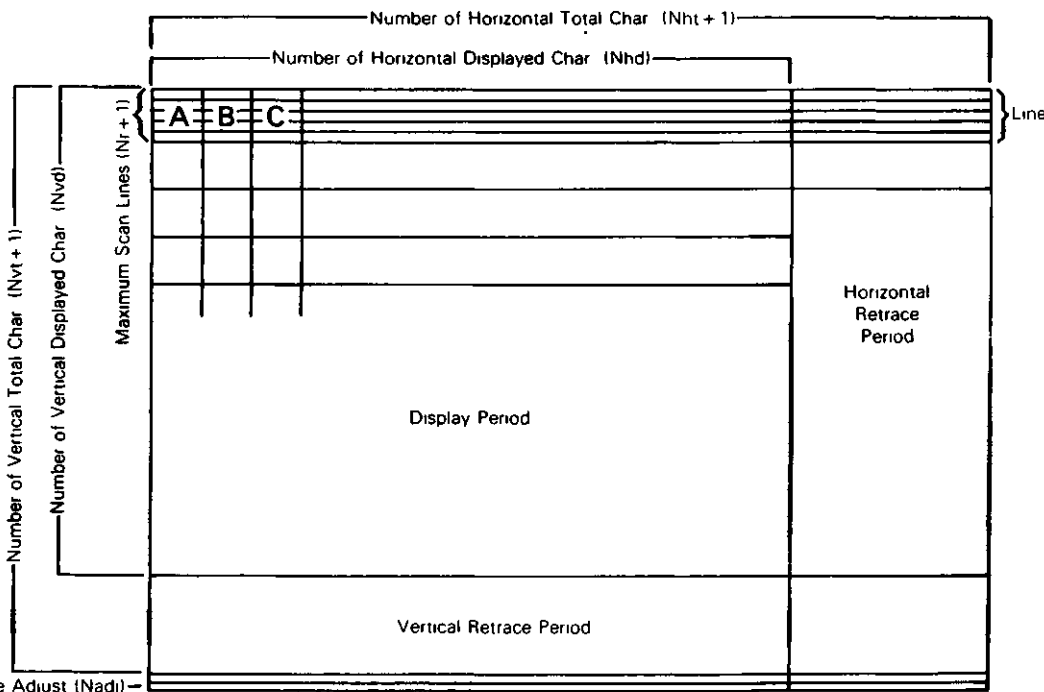
Horizontal Sync Position Register (R2) — This 8-bit register controls the HS position. The horizontal sync position defines the horizontal sync delay (Front Porch) and the horizontal scan delay (Back Porch). When the programmed value of this register is increased, the display on the CRT screen is shifted to the left. When the programmed value is

decreased the display is shifted to the right. Any 8-bit number may be programmed as long as the sum of the contents of R1, R2, and the lower four bits of R3 are less than the contents of R0.

Sync Width Register (R3) — This 8-bit register determines the width of the vertical sync (VS) pulse and the horizontal sync (HS) pulse. Programming the upper four bits for 1-to-15 will select VS pulse widths from 1-to-15 scan-line times. Programming the upper four bits as zeros will select a VS pulse width of 16 scan line times. The HS pulse width may be programmed from 1-to-15 character clock periods thus allowing compatibility with the HS pulse width specifications of many different monitors. If zeros are written into the lower four bits of this register, then no HS is provided.

Horizontal Timing Summary (Figure 11) — The difference between R0 and R1 is the horizontal blanking interval. This interval in the horizontal scan period allows the beam to return (retrace) to the left side of the screen. The retrace time is determined by the monitor's horizontal scan components. Retrace time is less than the horizontal blanking interval. A good rule of thumb is to make the horizontal blanking about 20% of the total horizontal scanning period for a CRT. In inexpensive TV receivers, the beam overscans the display screen so that aging of parts does not result in underscanning. Because of this, the retrace time should be about 1/3 the horizontal scanning period. The horizontal sync delay, HS pulse width and horizontal scan delay are typically programmed with 1:2:2 ratio.

FIGURE 10 — ILLUSTRATION OF THE CRT SCREEN FORMAT

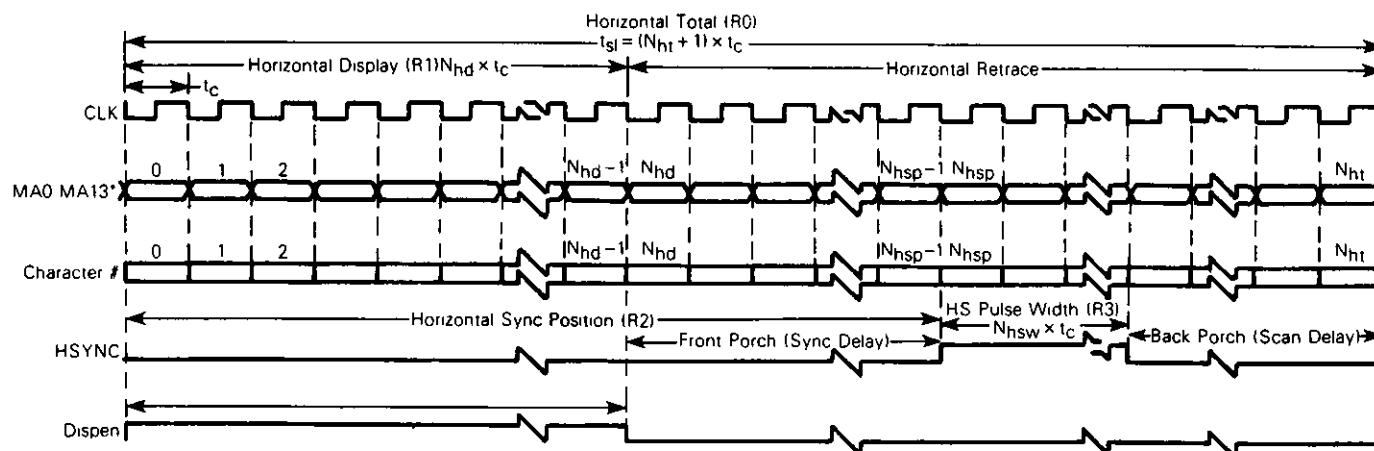


NOTE 1 Timing values are described in Table 8



MOTOROLA Semiconductor Products Inc.

FIGURE 11 — CRTC HORIZONTAL TIMING



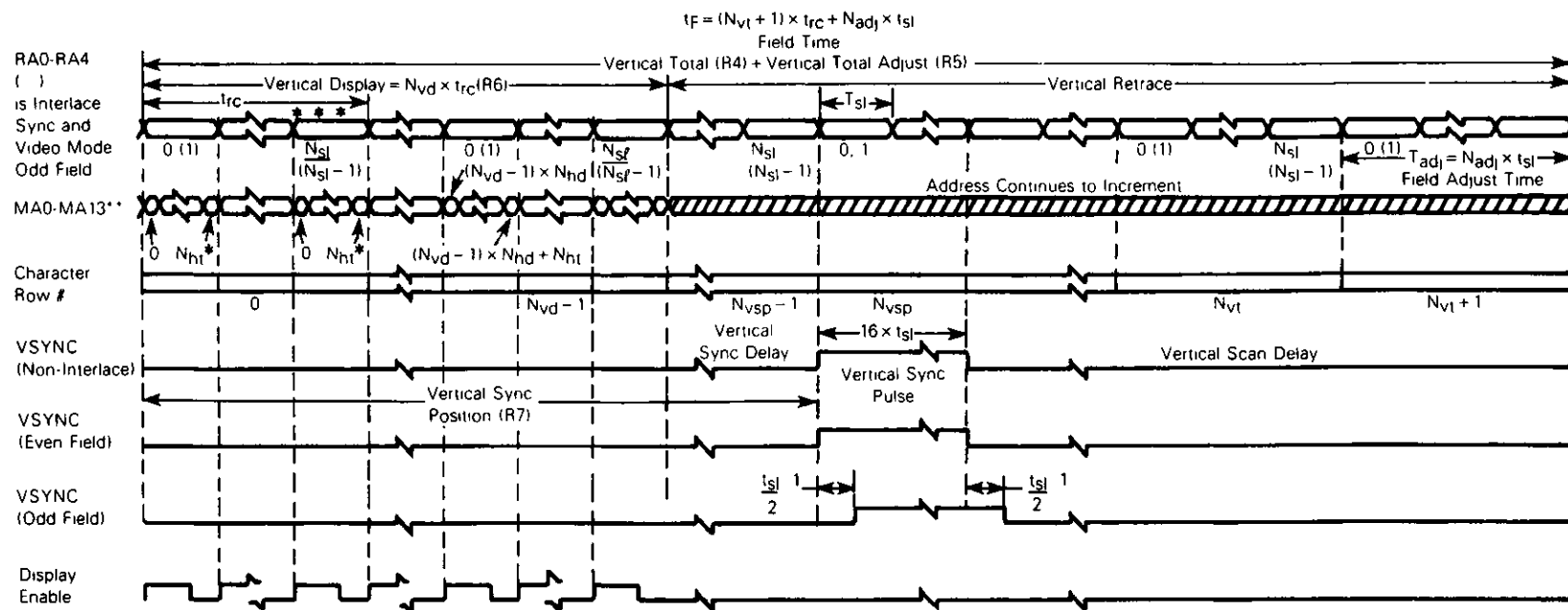
* Timing is shown for first displayed scan row only. See Chart in Figure 15 for other rows. The initial MA is determined by the contents of Start Address Register, R12/R13. Timing is shown for R12/R13=0.

NOTE 1: Timing values are described in Table 5.





FIGURE 12 — CRTC VERTICAL TIMING



- * N_{ht} must be an odd number for both interlace modes.
- **Initial MA is determined by R12/R13 (Start Address Register), which is zero in this timing example.
- *** N_{sl} must be an odd number for Interlace Sync and Video Mode

NOTES

1. Refer to Figure 6 — The Odd Field is offset $\frac{1}{2}$ horizontal scan time.
2. Timing values are described in Table 5.

TABLE 4 — CURSOR AND DE SKEW CONTROL

Value	Skew
00	No Character Skew
01	One Character Skew
10	Two Character Skew
11	Not Available

Maximum Scan Line Address Register (R9) — This 5-bit register determines the number of scan lines per character row including the spacing thus controlling operation of the Row Address counter. The programmed value is a maximum address and is one less than the number of scan lines.

Cursor Start Register (R10) and Cursor End Register (R11)

These registers allow a cursor of up to 32 scan lines in height to be placed on any scan line of the character block as shown in Figure 14. R10 is a 7-bit register used to define the start scan line and blink rate for the cursor. Bits 5 and 6 of the Cursor Start Address Register control the cursor operation as shown in Table 4. Non-display, display and two blink modes (16 times or 32 times the field period) are available. R11 is a 5-bit register which defines the last scan line of the cursor.

When an external blink feature on characters is required, it may be necessary to perform cursor blink externally so that both blink rates are synchronized. Note that an invert/non-invert cursor is easily implemented by programming the CRTC for a blinking cursor and externally inverting the video signal with an exclusive-OR gate.

PROGRAMMABLE REGISTERS

The four programmable registers allow the MPU to posi-

tion the cursor anywhere on the screen and allow the start address to be modified.

The Address Register is a five-bit write-only register used as an "indirect" or "pointer" register. Its contents are the address of one of the other 18 registers. When both RS and \overline{CS} are low, the Address Register is selected. When \overline{CS} is low and RS is high, the register pointed to by the Address Register is selected.

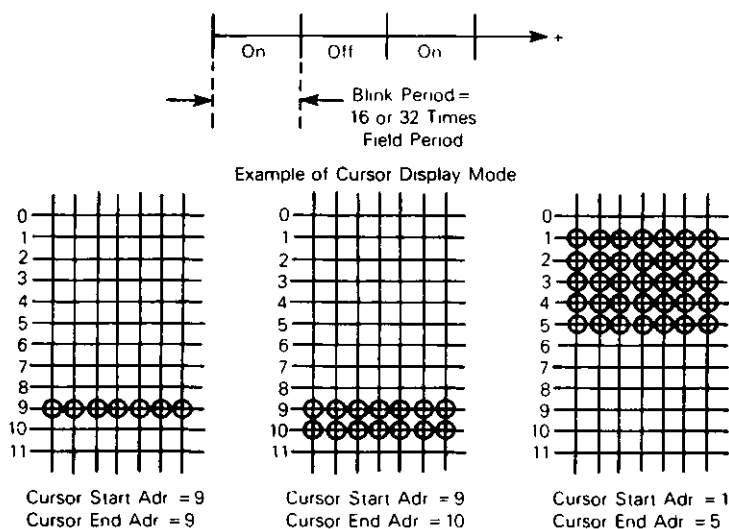
Start Address Register (R12-H, R13-L) — This 14-bit write-only register pair controls the first address output by the CRTC after vertical blanking. It consists of an 8-bit low order (MA0-MA7) register and a 6-bit high order (MA8-MA13) register. The start address register determines which portion of the refresh RAM is displayed on the CRT screen. Hardware scrolling by character, line or page may be accomplished by modifying the contents of this register.

Cursor Register (R14-H, R15-L) — This 14-bit write-only register pair is programmed to position the cursor anywhere in the refresh RAM area thus allowing hardware paging and scrolling through memory without loss of the original cursor position. It consists of an 8-bit low order (MA0-MA7) register and a 6-bit high order (MA8-MA13) register.

CRTC INITIALIZATION

Registers R12-R15 must be initialized after the system is powered up. The processor will normally load the CRTC register file from a firmware table. Figure 15 shows an M6800 program which could be used to program the CRT Controller.

FIGURE 14 — CURSOR CONTROL



MOTOROLA Semiconductor Products Inc.

ADDITIONAL CRTC APPLICATIONS

The foremost system function which may be performed by the CRTC controller is the refreshing of dynamic RAM. This is quite simple as the refresh addresses continually run.

Both the VS and the HS outputs may be used as a real time clock. Once programmed, the CRTC will provide a stable reference frequency.

SELECTING MASK PROGRAMMED REGISTER VALUES

A prototype system may be developed using the MC6845 CRTC. This will allow register values to be modified as re-

quired to meet system specifications. The worksheet of Table 5 is extremely useful in computing proper register values for the MC6835. The program shown in Figure 15 may be expanded to properly load the calculated register values in the MC6845. Once the two sets of register values have been developed, fill out the ROM program worksheet of Figure 18.

To order a custom programmed MC6835, contact your local field service office, local sales person or your local Motorola representative. A manufacturing mask will be developed for the data entered in Figure 18.

FIGURE 15 — M6800 PROGRAM FOR CRTC INITIALIZATION

```

PAGE 001 CRTCINIT.SA:1 MC6835 CRTC initialization program

00001          NAM      MC6835
00002          TTL      CRTC initialization program
00003          OPT      G,S,LLE=85 print FCB'x', FDB's & XREF table
00004          *****
00005          * Assign CRTC address
00006          *
00007          9000 A CRTCAD EQU $9000 Address Register
00008          9001 A CRTCRG EQU CRTCAD+1 Data Register
00009          *****
00010          * Initialization Program
00011          *
00012A 0000          ORG      0      a place to start
00013A 0000 C6 0C      A      LDAB    $C      initialize pointer
00014A 0002 CE 1020    A      LDX     38RTTAB table pointer
00015A 0005 F7 9000    A CRTCL STAB    CRTCAD load address register
00016A 0008 A6 00      A      LDAA    0,X    get register value from table
00017A 000A B7 9001    A      STAA    CRTCRG program register
00018A 000D 08          INX          increment counter
00019A 000E 5C          INCB
00020A 000F D1 10      A      CMPB    $10    finished?
00021A 0011 26 F2 0005 BNE     CRTCL    no: take branch
00022A 0013 3F          SWI          yes: call monitor
00023          *****
00024          * CRTC register initialization table
00025          *
00026A 1020          ORG      $1020 start of table
00027A 1020 0080 A CRTTAB FDB    $0080 R12, R13 - Start Address
00028A 1022 0080 A      FDB    $0080 R14, R15 - Cursor Address
00029          END
TOTAL ERRORS 00000--00000

```

CRTCL 0005 CRTCAD 9000 CRTCRG 9001 CRTTAB 1020



MOTOROLA Semiconductor Products Inc.



TABLE 5 - CRTC FORMAT WORKSHEET

Display Format Worksheet				CRTC Registers		
					Decimal	Hex
1	Displayed Characters per Row	_____	Char			
2	Displayed Character Rows per Screen	_____	Rows	R0	Horizontal Total (Line 15 - 1)	_____
3	Character Matrix			R1	Horizontal Displayed (Line 1)	_____
	a Columns	_____	Columns	R2	Horizontal Sync Position (Line 1 + Line 12)	_____
	b Rows	_____	Rows	R3	Horizontal Sync Width (Line 13)	_____
4	Character Block			R4	Vertical Total (Line 9 - 1)	_____
	a Columns	_____	Columns	R5	Vertical Adjust (Line 9 Lines)	_____
	b Rows	_____	Rows	R6	Vertical Displayed (Line 2)	_____
5	Frame Refresh Rate	_____	Hz	R7	Vertical Sync Position (Line 2 + Line 10)	_____
6	Horizontal Oscillator Frequency	_____	Hz	R8	Interlace (00 Normal, 01 Interlace, 03 Interlace, and Video)	_____
7	Active Scan Lines (Line 2 x Line 4b)	_____	Lines	R9	Max Scan Line Add (Line 4b - 1)	_____
8	Total Scan Lines (Line 6 - Line 5)	_____	Lines	R10	Cursor Start	_____
9	Total Rows Per Screen (Line 8 - Line 4b)	_____ Rows	and _____ Lines	R11	Cursor End	_____
10	Vertical Sync Delay (Char Rows)	_____	Rows	R12, R13	Start Address (H and L)	_____
11	Vertical Sync Width (Scan Lines (16))	_____	Lines	R14, R15	Cursor (H and L)	_____
12	Horizontal Sync Delay (Character Times)	_____	Char Times			
13	Horizontal Sync Width (Character Times)	_____	Char Times			
14	Horizontal Scan Delay (Character Times)	_____	Char Times			
15	Total Character Times (Line 1 + 12 + 13 + 14)	_____	Char Times			
16	Character Rate (Line 6 x 15)	_____	Hz			
17	Dot Clock Rate (Line 4a x 16)	_____	Hz			



TABLE 6 — WORKSHEET FOR 80x24 FORMAT

Display Format Worksheet

1	Displayed Characters per Row	<u>80</u>	Char
2	Displayed Character Rows per Screen	<u>24</u>	Rows
3	Character Matrix		
	a Columns	<u>7</u>	Columns
	b Rows	<u>9</u>	Rows
4	Character Block		
	a Columns	<u>9</u>	Columns
	b Rows	<u>11</u>	Rows
5	Frame Refresh Rate	<u>60</u>	Hz
6	Horizontal Oscillator Frequency	<u>18 600</u>	Hz
7	Active Scan Lines (Line 2 x Line 4b)	<u>264</u>	Lines
8	Total Scan Lines (Line 6 - Line 5)	<u>310</u>	Lines
9	Total Rows Per Screen (Line 8 - Line 4b)	<u>28</u>	Rows and <u>2</u> Lines
10	Vertical Sync Delay (Char Rows)	<u> </u>	Rows <u> </u>
11	Vertical Sync Width (Scan Lines (16))	<u>16</u>	Lines
12	Horizontal Sync Delay (Character Times)	<u>6</u>	Char Times
13	Horizontal Sync Width (Character Times)	<u>9</u>	Char Times
14	Horizontal Scan Delay (Character Times)	<u>7</u>	Char Times
15	Total Character Times (Line 1 + 12 + 13 + 14)	<u>102</u>	Char Times
16	Character Rate (Line 6 times 15)	<u>1 8972 M</u>	MHz
17	Dot Clock Rate (Line 4a times 16)	<u>17 075 M</u>	MHz

CRTC Registers

	Decimal	Hex
R0 Horizontal Total (Line 15 minus 1)	<u>101</u>	<u>65</u>
R1 Horizontal Displayed (Line 1)	<u>80</u>	<u>50</u>
R2 Horizontal Sync Position (Line 1 + Line 12)	<u>86</u>	<u>56</u>
R3 Horizontal Sync Width (Line 13)	<u>9</u>	<u>9</u>
R4 Vertical Total (Line 9 minus 1)	<u>27</u>	<u>18</u>
R5 Vertical Adjust (Line 9 Lines)	<u>2</u>	<u>0A</u>
R6 Vertical Displayed (Line 2)	<u>24</u>	<u>18</u>
R7 Vertical Sync Position (Line 2 + Line 10)	<u>24</u>	<u>18</u>
R8 Interlace (00 Normal 01 Interlace, 03 Interlace and Video)	<u> </u>	<u>0</u>
R9 Max Scan Line Add (Line 4b minus 1)	<u>10</u>	<u>8</u>
R10 Cursor Start	<u>0</u>	<u>0</u>
R11 Cursor End	<u>11</u>	<u>8</u>
R12 R13 Start Address (H and L)	<u>128</u>	<u>00</u>
		<u>80</u>
R14 R15 Cursor (H and L)	<u>128</u>	<u>00</u>
		<u>80</u>

OPERATION OF THE CRTC

Timing of the CRT Interface Signals — Timing charts of CRT interface signals are illustrated in this section with the aid of programmed example of the CRTC. When values listed in Table 7 are programmed into CRTC control registers, the device provides the outputs as shown in the Timing Diagrams (Figures 11, 12, 16, and 17). The screen

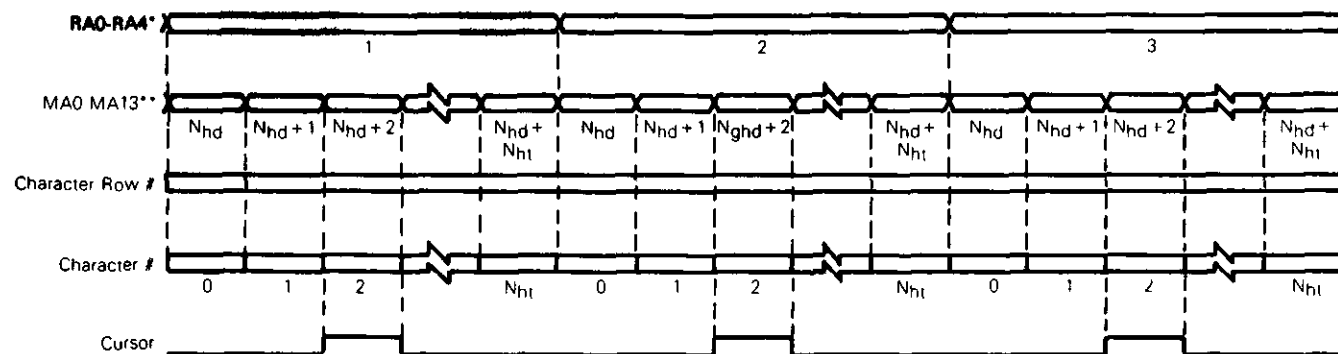
format of this example is shown in Figure 10. Figure 17 is an illustration of the relation between Refresh Memory Address (MA0-MA13), Raster Address (RA0-RA4) and the position on the screen. In this example, the start address is assumed to be "0".

TABLE 7 — VALUES PROGRAMMED INTO CRTC REGISTERS

Register Number	Register Name	Value	Programmed Value
R0	H Total	$N_{ht} + 1$	N_{ht}
R1	H Displayed	N_{hd}	N_{hd}
R2	H Sync Position	N_{hsp}	N_{hsp}
R3	H Sync Width	N_{hsw}	N_{hsw}
R4	V Total	$N_{vt} + 1$	N_{vt}
R5	V Scan Line Adjust	N_{adj}	N_{adj}
R6	V Displayed	N_{vd}	N_{vd}
R7	V Sync Position	N_{vsp}	N_{vsp}
R8	Interlace Mode		
R9	Max Scan Line Address	N_{sl}	N_{sl}
R10	Cursor Start		
R11	Cursor End		
R12	Start Address (H)	0	
R13	Start Address (L)	0	
R14	Cursor (H)		
R15	Cursor (L)		



FIGURE 16 — CURSOR TIMING



*Timing is shown for non interlace and interlace sync modes

Example shown has cursor programmed as

Cursor Register = $N_{hd} + 2$

Cursor Start = 1

Cursor End = 3

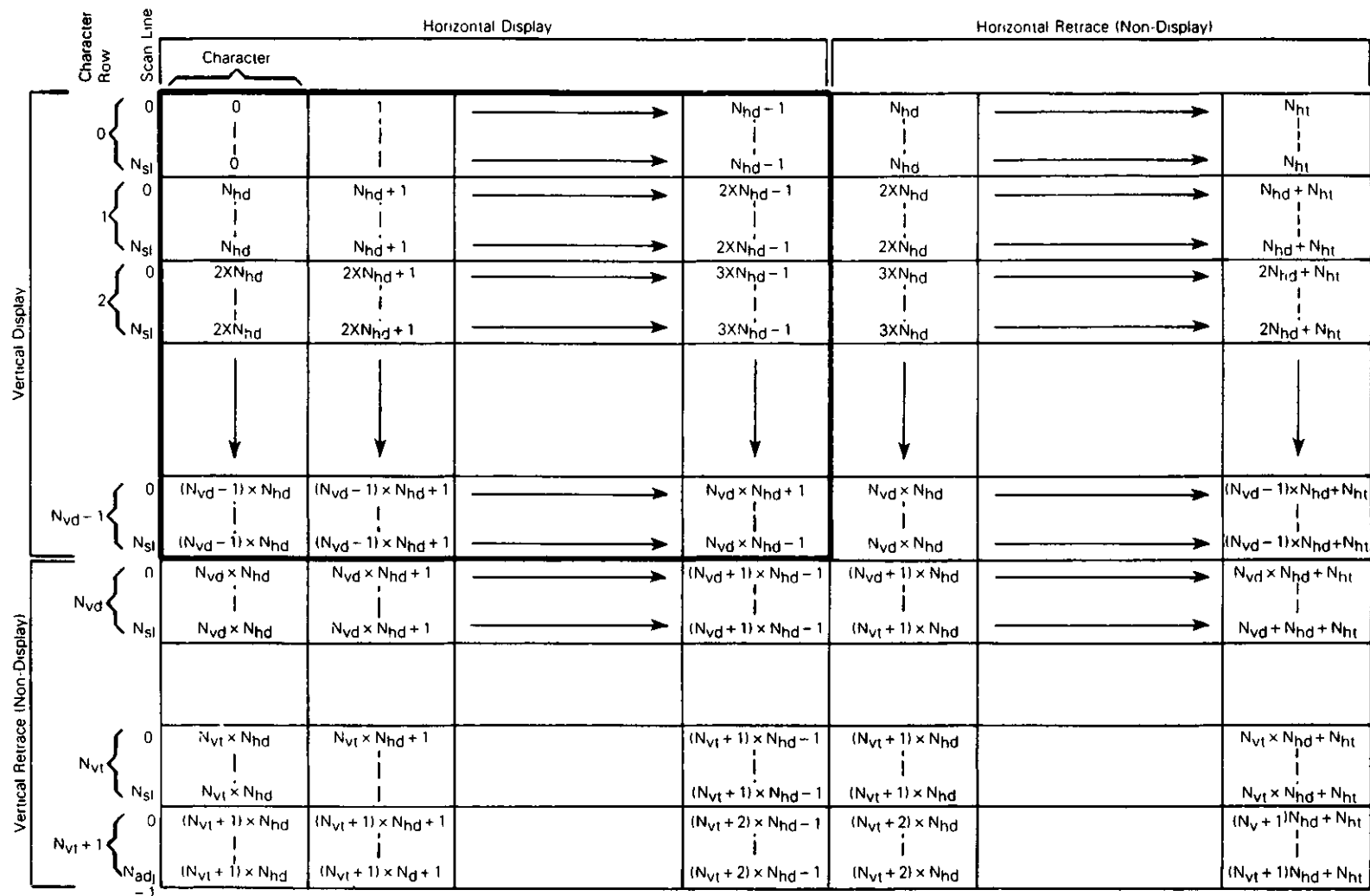
**The initial MA is determined by the contents of Start Address Register, R12/R13. Timing is shown for R12/R13 = 0

NOTE 1 Timing values are described in Table 8



MOTOROLA Semiconductor Products Inc.

FIGURE 17 -- REFRESH MEMORY ADDRESSING (MA0-MA13) STATE CHART



MOTOROLA Semiconductor Products Inc.

FIGURE 18 — ROM PROGRAM WORKSHEET

The value in each register of the MC6845 should be entered without any modifications. Motorola will take care of translating into the appropriate format.

☐ All numbers are in decimal. ☐ All numbers are in hex.

	ROM Program Zero (PROG=0)	ROM Program One (PROG=1)
R0	_____	_____
R1	_____	_____
R2	_____	_____
R3	_____	_____
R4	_____	_____
R5	_____	_____
R6	_____	_____
R7	_____	_____
R8	_____	_____
R9	_____	_____
R10	_____	_____
R11	_____	_____

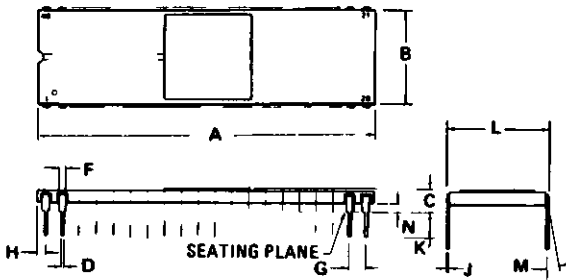
ORDERING INFORMATION

Package Type	Frequency (MHz)	Temperature	Order Number
Ceramic L Suffix	1.0	0°C to 70°C	MC6835L
	1.0	-50°C to 85°C	MC6835CL
	1.5	0°C to 70°C	MC68A35L
	1.5	-50°C to 85°C	MC68A35CL
	2.0	0°C to 70°C	MC68B35L
	2.0	-50°C to 85°C	MC68B35CL
Cerdip S Suffix	1.0	0°C to 70°C	MC6835S
	1.0	-50°C to 85°C	MC6835CS
	1.5	0°C to 70°C	MC68A35S
	1.5	-50°C to 85°C	MC68A35CS
	2.0	0°C to 70°C	MC68B35S
	2.0	-50°C to 85°C	MC68B35CS
Plastic P Suffix	1.0	0°C to 70°C	MC6835P
	1.0	-50°C to 85°C	MC6835CP
	1.5	0°C to 70°C	MC68A35P
	1.5	-50°C to 85°C	MC68A35CP
	2.0	0°C to 70°C	MC68B35P
	2.0	-50°C to 85°C	MC68B35CP



MOTOROLA Semiconductor Products Inc.

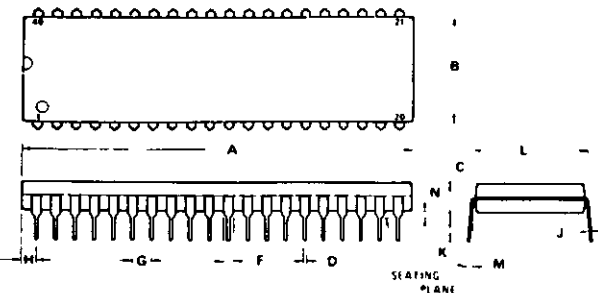
PACKAGE DIMENSIONS

L SUFFIX
 CERAMIC PACKAGE
 CASE 715-04


DIM	MILLIMETERS		INCHES	
	MIN	MAX	MIN	MAX
A	50.75	51.31	1.980	2.020
B	14.94	15.34	0.588	0.604
C	3.05	4.06	0.120	0.160
D	0.38	0.53	0.015	0.021
E	0.76	1.40	0.030	0.055
F	2.54 BSC		0.100 BSC	
G	0.76	1.78	0.030	0.070
H	0.20	0.33	0.008	0.013
J	2.54	4.19	0.100	0.165
K	14.99	15.49	0.590	0.610
M		10°		10°
N	1.02	1.52	0.040	0.060

NOTES

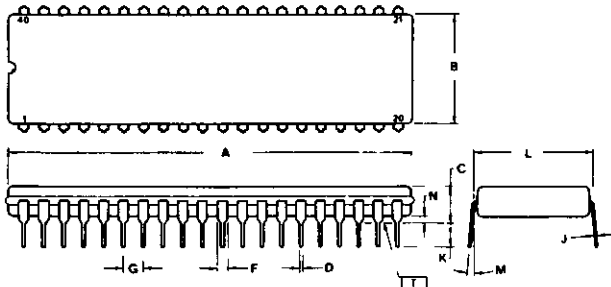
- LEADS TRUE POSITIONED WITHIN 0.25 mm (0.010) DIA (AT SEATING PLANE) AT MAX MAT L CONDITION
- DIMENSION L TO CENTER OF LEADS WHEN FORMED PARALLEL

P SUFFIX
 PLASTIC PACKAGE
 CASE 711-03


DIM	MILLIMETERS		INCHES	
	MIN	MAX	MIN	MAX
A	51.69	52.45	2.035	2.065
B	13.72	14.22	0.540	0.560
C	3.94	5.08	0.155	0.200
D	0.36	0.56	0.014	0.022
E	1.02	1.52	0.040	0.060
F	2.54 BSC		0.100 BSC	
G	1.65	2.16	0.065	0.085
H	0.20	0.38	0.008	0.015
J	2.92	3.43	0.115	0.135
K	15.24 BSC		0.600 BSC	
M	0°	15°	0°	15°
N	0.51	1.02	0.020	0.040

NOTES

- POSITIONAL TOLERANCE OF LEADS (D) SHALL BE WITHIN 0.25 mm (0.010) AT MAXIMUM MATERIAL CONDITION IN RELATION TO SEATING PLANE AND EACH OTHER
- DIMENSION L TO CENTER OF LEADS WHEN FORMED PARALLEL
- DIMENSION B DOES NOT INCLUDE MOLD FLASH

S SUFFIX
 CERP PACKAGE
 CASE 734-03


DIM	MILLIMETERS		INCHES	
	MIN	MAX	MIN	MAX
A	51.31	53.24	2.020	2.096
B	12.70	15.49	0.500	0.610
C	4.06	5.84	0.160	0.230
D	0.38	0.56	0.015	0.022
E	1.27	1.65	0.050	0.065
F	2.54 BSC		0.100 BSC	
G	0.20	0.30	0.008	0.012
H	3.18	4.06	0.125	0.160
J	15.24 BSC		0.600 BSC	
M	5°	15°	5°	15°
N	0.51	1.27	0.020	0.050

NOTES

- DIMENSION A IS DATUM
- POSITIONAL TOLERANCE FOR LEADS
 $\varnothing 0.25 (0.010) \text{ } \textcircled{A} \text{ } T \text{ } \textcircled{A}$
- IS SEATING PLANE
- DIMENSION L TO CENTER OF LEADS WHEN FORMED PARALLEL
- DIMENSION A AND B INCLUDES MENISCUS

Motorola reserves the right to make changes to any products herein to improve reliability function or design. Motorola does not assume any liability arising out of the application or use of any product or circuit described herein, neither does it convey any license under its patent rights nor the rights of others.


MOTOROLA Semiconductor Products Inc.

3501 ED BLUESTEIN BLVD AUSTIN TEXAS 78721 • A SUBSIDIARY OF MOTOROLA INC

BR1941(5016) Dual Baud Rate Clock

BR1941(5016)

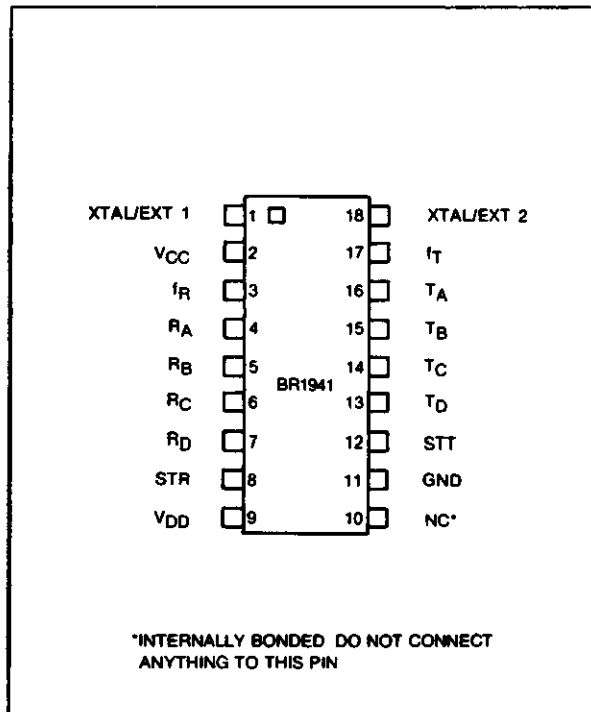
FEATURES

- 16 SELECTABLE BAUD RATE CLOCK FREQUENCIES
- SELECTABLE 1X, 16X OR 32X CLOCK OUTPUTS FOR FULL DUPLEX OPERATIONS
- OPERATES WITH CRYSTAL OSCILLATOR OR EXTERNALLY GENERATED FREQUENCY INPUT
- ROM MASKABLE FOR NON-STANDARD FREQUENCY SELECTIONS
- INTERFACES EASILY WITH MICROCOMPUTERS
- OUTPUTS A 50% DUTY CYCLE CLOCK WITH 0.01% ACCURACY
- 6 DIFFERENT FREQUENCY/DIVISOR PAIRS AVAILABLE
- TTL, MOS COMPATIBILITY
- PIN COMPATIBLE WITH COM5016

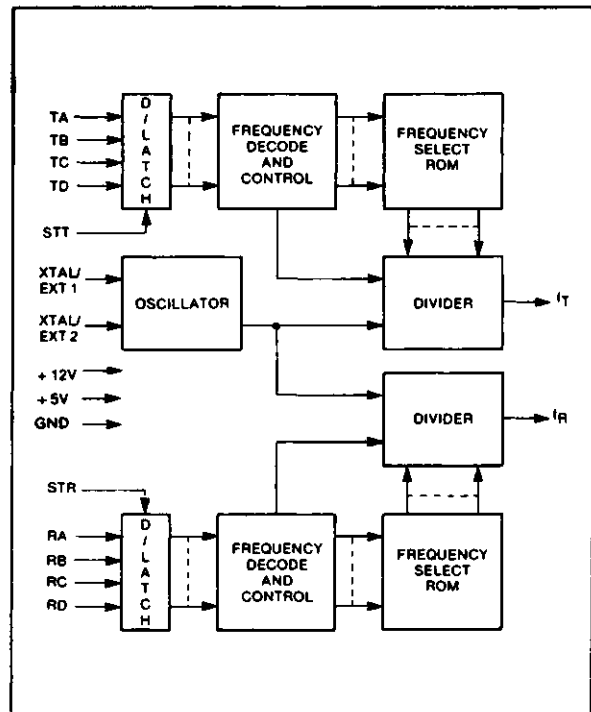
GENERAL DESCRIPTION

The BR1941 is a combination Baud Rate Clock Generator and Programmable Divider. It is manufactured in N-channel MOS using silicon gate technology. This device is capable of generating 16 externally selected clock rates whose frequency is determined by either a single crystal or an externally generated input clock. The BR1941 is a programmable counter capable of generating a division from 2 to $(2^{15} - 1)$.

The BR1941 is available programmed with the most used frequencies in data communication. Each frequency is selectable by strobing or hard wiring each of the two sets of four Rate Select inputs. Other frequencies/division rates can be generated by reprogramming the internal ROM coding through a MOS mask change. Additionally, further clock division may be accomplished through cascading of devices. The frequency output is fed into the XTAL/EXT input on a subsequent device.



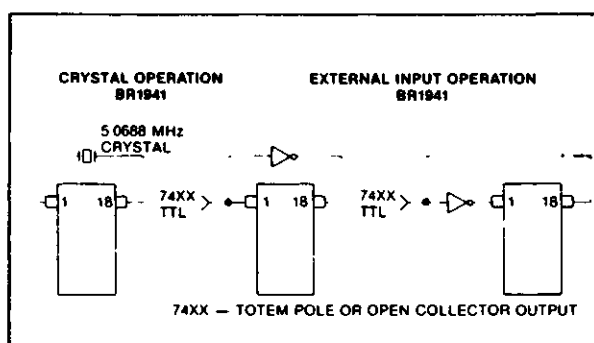
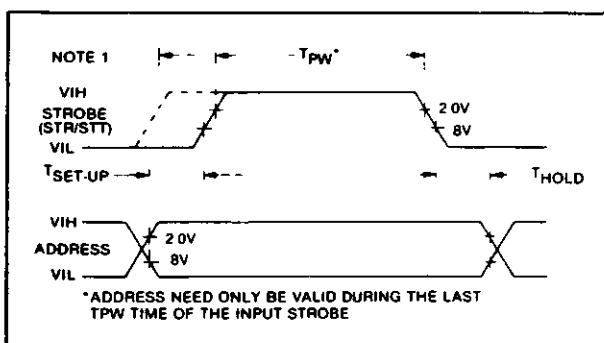
PIN CONNECTIONS



BR1941 BLOCK DIAGRAM

PIN DESCRIPTION

PIN NUMBER	SYMBOL	NAME	FUNCTION
1	XTAL/EXT 1	Crystal or External Input 1	This input receives one pin of the crystal package or one polarity of the external input.
2	VCC	Power Supply	+ 5 volt Supply
3	f_R	Receiver Output Frequency	This output runs at a frequency selected by the Receiver Address inputs.
4-7	R_A, R_B, R_C, R_D	Receiver Address	The logic level on these inputs as shown in Tables 1 through 6, selects the receiver output frequency, f_R .
8	STR	Strobe-Receiver Address	A high-level input strobe loads the receiver address (R_A, R_B, R_C, R_D) into the receiver address register. This input may be strobed or hard wired to +5V.
9	VDD	Power Supply	+ 12 volt Supply
10	NC	No Connection	Internally bonded. Do not connect anything to this pin.
11	GND	Ground	Ground
12	STT	Strobe-Transmitter Address	A high-level input strobe loads the transmitter address (T_A, T_B, T_C, T_D) into the transmitter address register. This input may be strobed or hard wired to +5V.
13-16	T_D, T_C, T_B, T_A	Transmitter Address	The logic level on these inputs, as shown in Tables 1 through 6, selects the transmitter output frequency, f_T .
17	f_T	Transmitter Output Frequency	This output runs at a frequency selected by the Transmitter Address inputs.
18	XTAL/EXT 2	Crystal or External Input 2	This input receives the other pin of the crystal package or the other polarity of the external input.



ABSOLUTE MAXIMUM RATINGS

Positive Voltage on any Pin, with respect to ground	+ 20.0V
Negative Voltage on any Pin, with respect to ground	- 0.3V
Storage Temperature	(plastic package) - 55°C to + 125°C (cerdip package and ceramic package) - 65°C to + 150°C
Lead Temperature (Soldering, 10 sec.)	+ 325°C

*Stresses above those listed may cause permanent damage to the device. This is a stress rating only and Functional Operation of the device at these or at any other condition above those indicated in the operational sections of this specification are not implied.

ELECTRICAL CHARACTERISTICS(T_A = 0°C to +70°C, V_{CC} = +5V ± 5%, V_{DD} = +12V ± 5%, unless otherwise noted)

PARAMETER	MIN	TYP	MAX	UNIT	COMMENTS	
DC CHARACTERISTICS						
INPUT VOLTAGE LEVELS						
Low-level, V_{IL}	$V_{CC} - 1.5$		0.8	V	See Note 1	
High-level, V_{IH}			V_{CC}	V		
OUTPUT VOLTAGE LEVELS						
Low-level, V_{OL}	$V_{CC} - 1.5$	4.0	0.4	V	$I_{OL} = 3.2\text{ mA}$ $I_{OH} = 100\mu\text{A}$	
High-level, V_{OH}			V			
INPUT CURRENT						
Low-level, I_{IL}			0.3	mA	$V_{IN} = \text{GND}$, excluding XTAL inputs	
INPUT CAPACITANCE						
All Inputs, C_{IN}		5	10	pf	$V_{IN} = \text{GND}$, excluding XTAL inputs	
INPUT RESISTANCE						
Crystal Input, R_{XTAL}	1.1			K Ω	Resistance to ground for Pin 1 and Pin 18	
POWER SUPPLY CURRENT						
I_{CC}		20	60	mA	$T_A = +25^\circ\text{C}$	
I_{DD}		20	70	mA		
AC CHARACTERISTICS						
CLOCK FREQUENCY						
PULSE WIDTH (T_{PW})						
Clock					50% duty cycle $\pm 10\%$. See Note 2	
Receiver strobe	150		DC	ns		
Transmitter strobe	150		DC	ns		
INPUT SET-UP TIME (T_{SET-UP})						
Address	50			ns	See Note 3	
OUTPUT HOLD TIME (T_{HOLD})						
Address	50			ns		

NOTE 1: BR1941 — XTAL/EXT inputs are either TTL compatible or crystal compatible. See crystal specification in Applications Information section.

All inputs except XTAL/EXT have internal pull-up resistors.

NOTE 2: Refer to frequency option tables for maximum input frequency on XTAL/EXT pins.

Typical Clock Pulse width is 1/2xCL.

NOTE 3: Input set-up time can be decreased to ≥ 0 ns by increasing the minimum strobe width by 50 ns to a total of 200 ns.

OPERATION**Standard Frequencies**

Choose a Transmitter and Receiver frequency from the table below. Program the corresponding address into TA-TD and RA-RD respectively using strobe pulses or by hard wiring the strobe and address inputs.

Non-Standard Frequencies

To accomplish non-standard frequencies do one of the following:

1. Choose a crystal that when divided by the BR1941 generates the desired frequency.
2. Cascade devices by using the frequency outputs as an

input to the XTAL/EXT inputs of the subsequent
BR1941.

3. Consult the factory for possible changes via ROM mask
reprogramming.

FREQUENCY OPTIONS

TABLE 1. CRYSTAL FREQUENCY = 5.0688 MHZ

Transmit/Receive Address				Baud Rate (16X Clock)	Theoretical Freq. (kHz)	Actual Freq. (kHz)	Percent Error	Duty Cycle %	Divisor
D	C	B	A						
0	0	0	0	50	0.8	0.8	—	50/50	6336
0	0	0	1	75	1.2	1.2	—	50/50	4224
0	0	1	0	110	1.76	1.76	—	50/50	2880
0	0	1	1	134.5	2.152	2.1523	0.016	50/50	2355
0	1	0	0	150	2.4	2.4	—	50/50	2112
0	1	0	1	300	4.8	4.8	—	50/50	1056
0	1	1	0	600	9.6	9.6	—	50/50	528
0	1	1	1	1200	19.2	19.2	—	50/50	264
1	0	0	0	1800	28.8	28.8	—	50/50	176
1	0	0	1	2000	32.0	32.081	0.253	50/50	158
1	0	1	0	2400	38.4	38.4	—	50/50	132
1	0	1	1	3600	57.6	57.6	—	50/50	88
1	1	0	0	4800	76.8	76.8	—	50/50	66
1	1	0	1	7200	115.2	115.2	—	50/50	44
1	1	1	0	9600	153.6	153.6	—	48/52	33
1	1	1	1	19,200	307.2	316.8	3.125	50/50	16

BR1941-00

TABLE 2. CLOCK FREQUENCY = 2.76480 MHZ

Transmit/Receive Address				Baud Rate (16X Clock)	Theoretical Freq. (kHz)	Actual Freq. (kHz)	Percent Error	Duty Cycle %	Divisor
D	C	B	A						
0	0	0	0	50	0.8	0.8	—	50/50	3456
0	0	0	1	75	1.2	1.2	—	50/50	2304
0	0	1	0	110	1.76	1.76	-0.006	50/50	1571
0	0	1	1	134.5	2.152	2.152	-0.019	50/50	1285
0	1	0	0	150	2.4	2.4	—	50/50	1152
0	1	0	1	200	3.2	3.2	—	50/50	864
0	1	1	0	300	4.8	4.8	—	50/50	576
0	1	1	1	600	9.6	9.6	—	50/50	288
1	0	0	0	1200	19.2	19.2	—	50/50	144
1	0	0	1	1800	28.8	28.8	—	50/50	96
1	0	1	0	2000	32.0	32.15	+0.465	50/50	86
1	0	1	1	2400	38.4	38.4	—	50/50	72
1	1	0	0	3600	57.6	57.6	—	50/50	48
1	1	0	1	4800	76.8	76.8	—	50/50	36
1	1	1	0	9600	153.6	153.6	—	50/50	18
1	1	1	1	19,200	307.2	307.2	—	50/50	9

BR1941-02

TABLE 3. CRYSTAL FREQUENCY = 6.018305 MHZ

Transmit/Receive Address				Baud Rate (16X Clock)	Theoretical Freq. (kHz)	Actual Freq. (kHz)	Percent Error	Duty Cycle %	Divisor
D	C	B	A						
0	0	0	0	50	0.8	.7999	0	50/50	7523*
0	0	0	1	75	1.2	1.2000	0	50/50	5015*
0	0	1	0	110	1.76	1.7597	0	50/50	3420
0	0	1	1	134.5	2.152	2.1517	0	50/50	2797*
0	1	0	0	150	2.4	2.3996	0	50/50	2508
0	1	0	1	200	3.2	3.1995	0	50/50	1881*
0	1	1	0	300	4.8	4.7993	0	50/50	1254
0	1	1	1	600	9.6	9.5986	0	50/50	627*
1	0	0	0	1200	19.2	19.2279	+0.14	50/50	31.3*
1	0	0	1	1800	28.8	28.7959	0	50/50	209*
1	0	1	0	2000	32.0	32.0125	0	50/50	188
1	0	1	1	2400	38.4	38.3334	-0.17	50/50	157*
1	1	0	0	3600	57.6	57.8687	+0.46	50/50	104
1	1	0	1	4800	76.8	77.1583	+0.46	50/50	78
1	1	1	0	9600	153.6	154.3166	+0.46	50/50	39*
1	1	1	1	19,200	307.2	300.9175	-2.04	50/50	20

BR1941-03

TABLE 4. CLOCK FREQUENCY = 5.52960 MHZ

Transmit/Receive Address				Baud Rate	Theoretical	Actual	Percent	Duty Cycle	Divisor
D	C	B	A	(16X Clock)	Freq. (kHz)	Freq. (kHz)	Error	%	
0	0	0	0	50	1.6	1.6	—	50/50	3456
0	0	0	1	75	2.4	2.4	—	50/50	2304
0	0	1	0	110	3.52	3.52	-0.006	50/50	1571
0	0	1	1	134.5	4.304	4.303	-0.019	50/50	1285
0	1	0	0	150	4.8	4.8	—	50/50	1152
0	1	0	1	200	6.4	6.4	—	50/50	864
0	1	1	0	300	9.6	9.6	—	50/50	576
0	1	1	1	600	19.2	19.2	—	50/50	288
1	0	0	0	1200	38.4	38.4	—	50/50	144
1	0	0	1	1800	57.6	57.6	—	50/50	96
1	0	1	0	2000	64.0	64.3	+0.465	50/50	86
1	0	1	1	2400	76.8	76.8	—	50/50	72
1	1	0	0	3600	115.2	115.2	—	50/50	48
1	1	0	1	4800	153.6	153.6	—	50/50	36
1	1	1	0	9600	307.2	307.2	—	50/50	18
1	1	1	1	19,200	614.4	614.4	—	50/50	9

BR1941-04

TABLE 5. CRYSTAL FREQUENCY = 4.9152 MHZ

Transmit/Receive Address				Baud Rate	Theoretical	Actual	Percent	Duty Cycle	Divisor
D	C	B	A	(32X Clock)	Freq. (kHz)	Freq. (kHz)	Error	%	
0	0	0	0	50	0.8	0.8	—	50/50	6144
0	0	0	1	75	1.2	1.2	—	50/50	4096
0	0	1	0	110	1.76	1.7598	-0.01	•	2793
0	0	1	1	134.5	2.152	2.152	—	50/50	2284
0	1	0	0	150	2.4	2.4	—	50/50	2048
0	1	0	1	300	4.8	4.8	—	50/50	1024
0	1	1	0	600	9.6	9.6	—	50/50	512
0	1	1	1	1200	19.2	19.2	—	50/50	256
1	0	0	0	1800	28.8	28.7438	-0.19	•	171
1	0	0	1	2000	32.0	31.9168	-0.26	50/50	154
1	0	1	0	2400	38.4	38.4	—	50/50	128
1	0	1	1	3600	57.6	57.8258	0.39	•	85
1	1	0	0	4800	76.8	76.8	—	50/50	64
1	1	0	1	7200	115.2	114.306	-0.77	•	43
1	1	1	0	9600	153.6	153.6	—	50/50	32
1	1	1	1	19,200	307.2	307.2	—	50/50	16

BR1941-05

TABLE 6. CRYSTAL FREQUENCY = 5.0688 MHZ

Transmit/Receive Address				Baud Rate	Theoretical	Actual	Percent	Duty Cycle	Divisor
D	C	B	A	(32X Clock)	Freq. (kHz)	Freq. (kHz)	Error	%	
0	0	0	0	50	1.6	1.6	—	50/50	3168
0	0	0	1	75	2.4	2.4	—	50/50	2112
0	0	1	0	110	3.52	3.52	—	50/50	1440
0	0	1	1	134.5	4.304	4.303	.026	50/50	1178
0	1	0	0	150	4.8	4.8	—	50/50	1056
0	1	0	1	200	6.4	6.4	—	50/50	792
0	1	1	0	300	9.6	9.6	—	50/50	528
0	1	1	1	600	19.2	19.2	—	50/50	264
1	0	0	0	1200	38.4	38.4	—	50/50	132
1	0	0	1	1800	57.6	57.6	—	50/50	88
1	0	1	0	2400	76.8	76.8	—	50/50	66
1	0	1	1	3600	115.2	115.2	—	50/50	44
1	1	0	0	4800	153.6	153.6	—	•	33
1	1	0	1	7200	230.4	230.4	—	50/50	22
1	1	1	0	9600	307.2	298.16	2.941	•	17
1	1	1	1	19,200	614.4	633.6	3.125	50/50	8

*When the duty cycle is not exactly 50% it is 50% \pm 10%

BR1941-06

CRYSTAL SPECIFICATIONS

User must specify termination (pin, wire, other)
Frequency — See Tables 1-6.
Temperature range 0°C to +70°C
Series resistance $\leq 50\Omega$
Series resonant
Overall tolerance $\pm .01\%$

CRYSTAL MANUFACTURERS (Partial List)

American Time Products Div.
Frequency Control Products, Inc.
61-20 Woodside Ave.
Woodside, New York 11377
(212) 458-5811

Bliley Electric Co.
2545 Grandview Blvd.
Erie, Pennsylvania 16508
(814) 838-3571

M-tron Ind. Inc.
P.O. Box 630
Yankton, South Dakota 57078
(605) 665-9321

Erie Frequency Control
453 Lincoln St.
Calisle, Pennsylvania 17013
(714) 249-2232

APPLICATIONS INFORMATION

OPERATION WITH A CRYSTAL

The BR1941 Baud Rate Generator may be driven by either a crystal or TTL level clock. When using a crystal, the waveform that appears at pins 1 (XTAL/EXT 1) and 18 (XTAL/EXT 2) does not conform to the normal TTL limits of $V_{IL} \leq 0.8V$ and $V_{IH} \geq 2.0V$. Figure 1 illustrates a typical crystal waveform when connected to a BR1941.

Since the D.C. level of the waveform causes the least positive point to typically be greater than 0.8V, the BR1941 is designed to look for an edge, as opposed to a TTL level. The XTAL/EXT logic triggers on a rising edge of typically 1V in magnitude. This allows the use of a crystal without any additional components.

OPERATIONS WITH TTL LEVEL CLOCK

With clock frequencies in the area of 5 MHz, significant overshoot and undershoot ("ringing") can appear at pins 1 and/or 18. The BR1941, may, at times, be triggered on a rising edge of an overshoot or undershoot waveform, causing the device to effectively "double-trigger." This phenomenon may result as a twice expected baud rate, or as an apparent device failure. Figure 2 shows a typical waveform that exhibits the "ringing" problem.

The design methods required to minimize ringing include the following:

1. Minimize the P.C. trace length. At 5 MHz, each inch of trace can add significantly to overshoot and undershoot.
2. Match impedances at both ends of the trace. For example, a series resistor near the BR1941 may be helpful.
3. A uniform impedance is important. This can be accomplished through the use of:

- a. parallel ground lines
- b. evenly spaced ground lines crossing the trace on the opposite side of PC board
- c. an inner plane of ground, e.g., as in a four layered PC board.

In the event that ringing exists on an already finished board, several techniques can be used to reduce it. These are:

1. Add a series resistor to match impedance as shown in Figure 3.
2. Add pull-up/pull-down resistor to match impedance, as shown in Figure 4.
3. Add a high speed diode to clamp undershoot, as shown in Figure 5.

The method that is easiest to implement in many systems is method 1, the series resistor. The series resistor will cause the D.C. level to shift up, but that does not cause a problem since the BR1941 is triggered by an edge, as opposed to a TTL level.

The BR1941 Baud Rate Generator can save both board space and cost in a communications system. By choosing either a crystal or a TTL level clock, the user can minimize the logic required to provide baud rate clocks in a given design.

POWER LINE SPIKES

Voltage transients on the AC power line may appear on the DC power output. If this possibility exists, it is suggested that one by-pass capacitor is used between +5V and GND and another between +12V and GND.

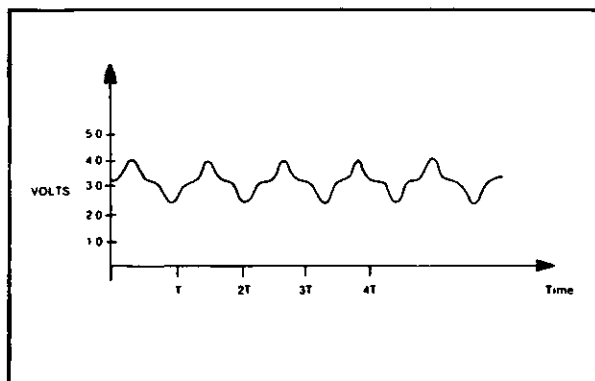


Figure 1 TYPICAL CRYSTAL WAVEFORM

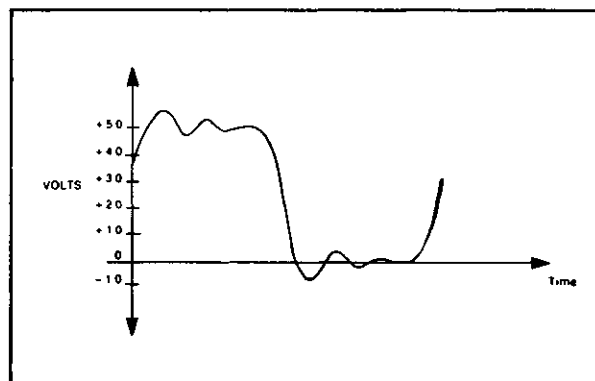


Figure 2 TYPICAL "RINGING" WAVEFORM

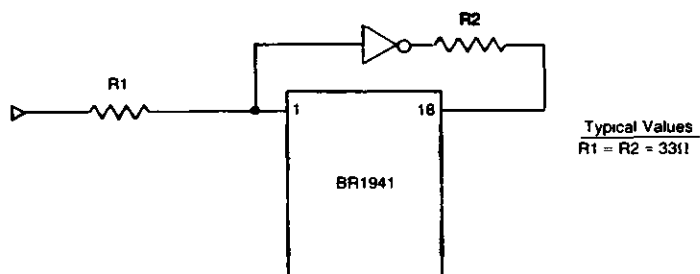


Figure 3 SERIES RESISTOR TO MATCH IMPEDANCE

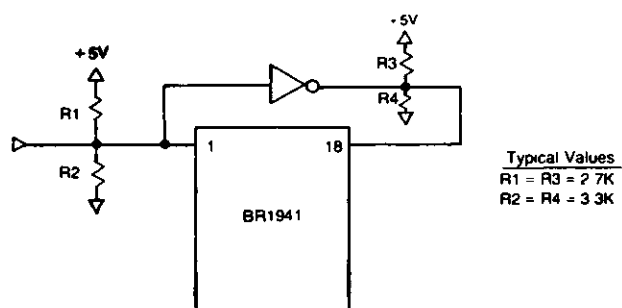


Figure 4 PULL-UP/PULL-DOWN RESISTORS TO MATCH IMPEDANCE

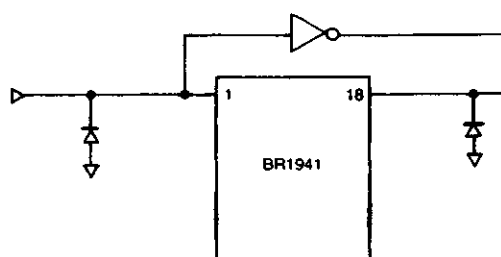


Figure 5 HIGH-SPEED DIODE TO CLAMP UNDERSHOOT

See page 725 for ordering information.

Information furnished by Western Digital Corporation is believed to be accurate and reliable. However, no responsibility is assumed by Western Digital Corporation for its use, nor for any infringements of patents or other rights of third parties which may result from its use. No license is granted by implication or otherwise under any patent or patent rights of Western Digital Corporation. Western Digital Corporation reserves the right to change specifications at anytime without notice.

FEATURES

- ## GENERAL DESCRIPTION

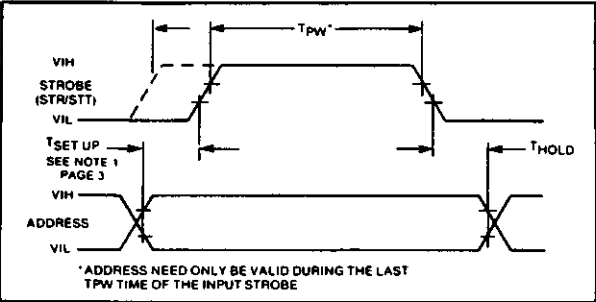
The WD1943/45 is available programmed with the most used frequencies in data communication. Each frequency is selectable by strobing or hard wiring each of the two sets of four Rate Select inputs. Other frequencies/division rates can be generated by reprogramming the internal ROM coding through a MOS mask change. Additionally, further clock division may be accomplished through cascading of devices. The frequency output is fed into the XTAL/EXT input on a subsequent device.

XTAL/EXT 1	□ 1 □	18 □	XTAL/EXT 2
+ 5V	□ 2	17 □	f _T
f _R	□ 3	16 □	T _A
R _A	□ 4	15 □	T _B
R _B	□ 5	14 □	T _C
	WD1943 OR		
R _C	□ 6	13 □	T _D
	WD1945		
R _D	□ 7	12 □	STT
STR	□ 8	11 □	GND
NC	□ 9	10 □	NC (1943) f/4 (1945)

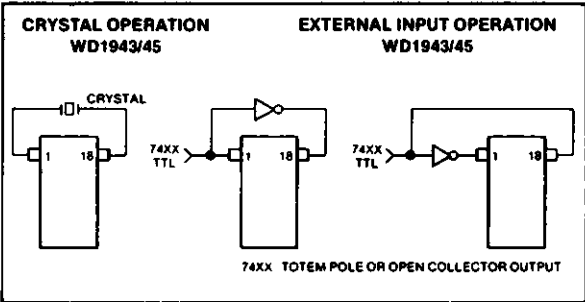
397

PIN DESCRIPTION

PIN NUMBER	SYMBOL	NAME	FUNCTION
1	XTAL/EXT 1	Crystal or External Input 1	This input receives one pin of the crystal package or one polarity of the external input.
2	VCC	Power Supply	+ 5 volt Supply
3	f _R	Receiver Output Frequency	This output runs at a frequency selected by the Receiver Address inputs.
4-7	R _A , R _B , R _C , R _D	Receiver Address	The logic level on these inputs as shown in Table 1 thru 6, selects the receiver output frequency, f _R .
8	STR	Strobe-Receiver Address	A high-level input strobe loads the receiver address (R _A , R _B , R _C , R _D) into the receiver address register. This input may be strobed or hard wired to +5V.
9	NC	No Connection	No Internal Connection
10	NC (1943) f/4 (1945)	No Connection freq/4 Output	No Internal Connection XTAL1 input freq divided by four.
11	GND	Ground	Ground
12	STT	Strobe-Transmitter Address	A high-level input strobe loads the transmitter address (T _A , T _B , T _C , T _D) into the transmitter address register. This input may be strobed or hard wired to +5V.
13-16	T _D , T _C , T _B , T _A	Transmitter Address	The logic level on these inputs, as shown in Table 1 thru 6, selects the transmitter output frequency, f _T .
17	f _T	Transmitter Output Frequency	This output runs at a frequency selected by the Transmitter Address inputs.
18	XTAL/EXT 2	Crystal or External Input 2	This input receives the other pin of the crystal package or the other polarity of the external input.



CONTROL TIMING



CRYSTAL/CLOCK OPTIONS

ABSOLUTE MAXIMUM RATINGS

Positive Voltage on any Pin, with respect to ground	+ 7.0V
Negative Voltage on any Pin, with respect to ground	- 0.3V
Storage Temperature	(plastic package) - 55°C to + 125°C (Cerdip package and Ceramic package) - 65°C to + 150°C
Lead Temperature (Soldering, 10 sec.)	+ 325°C

*Stresses above those listed may cause permanent damage to the device. This is a stress rating only and Functional Operation of the device at these or at any other condition above those indicated in the operational sections of this specification are not implied.

ELECTRICAL CHARACTERISTICS ($T_A = 0^\circ\text{C}$ to $+70^\circ\text{C}$, $V_{CC} = +5\text{V} \pm 5\%$ standard.)

PARAMETER	MIN	TYP	MAX	UNIT	COMMENTS
DC CHARACTERISTICS					
INPUT VOLTAGE LEVELS					
Low-level, V_{IL}	2.0		0.8	V	See Note 1
High-level, V_{IH}			V_{CC}	V	
OUTPUT VOLTAGE LEVELS					
Low-level, V_{OL}	$V_{CC}-1.5$	4.0	0.4	V	$I_{OL} = 3.2\text{ mA}$
High-level, V_{OH}			V	$I_{OH} = 100\mu\text{A}$	
INPUT CURRENT					
High-level, I_{IH}			- 10	μA	$V_{IN} = V_{CC}$ STR (8) and STT (12)
Low-level, I_{IL}			10	μA	$V_{IN} = \text{GND}$ Only
			300	μA	$V_{IN} = \text{GND}$ (All inputs except XTAL, STR and STT)
			10	μA	$V_{IN} = \text{GND}$ STR, STT
INPUT CAPACITANCE					
All Inputs, C_{IN}		5	10	pf	$V_{IN} = \text{GND}$, excluding XTAL inputs
EXT. INPUT LOAD					
		4	5		Series 7400 unit loads
INPUT RESISTANCE					
Crystal Input, R_{XTAL}	1.1			K Ω	Resistance to ground for Pin 1 and Pin 18
POWER SUPPLY CURRENT					
I_{CC}		40	80	mA	
AC CHARACTERISTICS					
$T_A = +25^\circ\text{C}$					
CLOCK FREQUENCY					
					See Note 2
PULSE WIDTH (T_{PW})					
Clock					50% Duty Cycle $\pm 10\%$. See Note 2
Receiver strobe	150		DC	ns	See Note 3
Transmitter strobe	150		DC	ns	See Note 3
INPUT SET-UP TIME (T_{SET-UP})					
Address	50			ns	See Note 3
OUTPUT HOLD TIME (T_{HOLD})					
Address	50			ns	
STROBE TO NEW FREQUENCY DELAY					
			6	CLK	

NOTE 1: XTAL/EXT inputs are either TTL compatible or crystal compatible. See crystal specification in Applications Information section.

All inputs except XTAL, STR and STT have internal pull-up resistors.

NOTE 2: Refer to frequency option tables for maximum input frequency on XTAL/EXT pins.

Typical clock pulse width is $1/2 \times CL$.

NOTE 3: Input set-up time can be decreased to >0 ns by increasing the minimum strobe width (50 ns) to a total of 200 ns. T_{A-D} and R_{A-D} have internal pull-up resistors.

OPERATION**Standard Frequencies**

Choose a Transmitter and Receiver frequency from the table below. Program the corresponding address into T_{A-TD} and R_{A-RD} respectively using strobe pulses or by hard wiring the strobe and address inputs.

Non-Standard Frequencies

To accomplish non-standard frequencies do one of the following:

1. Choose a crystal that when divided by the WD1943 generates the desired frequency.
2. Cascade devices by using the frequency outputs as an input to the XTAL/EXT inputs of the subsequent WD1943/45.
3. Consult the factory for possible changes via ROM mask reprogramming.

FREQUENCY OPTIONS

TABLE 1. CRYSTAL FREQUENCY = 5.0688 MHZ

Transmit/Receive Address				Baud Rate (16X Clock)	Theoretical Freq. (kHz)	Actual Freq. (kHz)	Percent Error	Duty Cycle %	Divisor
D	C	B	A						
0	0	0	0	50	0.8	0.8	—	50/50	6336
0	0	0	1	75	1.2	1.2	—	50/50	4224
0	0	1	0	110	1.76	1.76	—	50/50	2880
0	0	1	1	134.5	2.152	2.1523	0.016	50/50	2355
0	1	0	0	150	2.4	2.4	—	50/50	2112
0	1	0	1	300	4.8	4.8	—	50/50	1056
0	1	1	0	600	9.6	9.6	—	50/50	528
0	1	1	1	1200	19.2	19.2	—	50/50	264
1	0	0	0	1800	28.8	28.8	—	50/50	176
1	0	0	1	2000	32.0	32.081	0.253	50/50	158
1	0	1	0	2400	38.4	38.4	—	50/50	132
1	0	1	1	3600	57.6	57.6	—	50/50	88
1	1	0	0	4800	76.8	76.8	—	50/50	66
1	1	0	1	7200	115.2	115.2	—	50/50	44
1	1	1	0	9600	153.6	153.6	—	48/52	33
1	1	1	1	19,200	307.2	316.8	3.125	50/50	16

WD1943-00 or WD1945-00

TABLE 2. CLOCK FREQUENCY = 2.76480 MHZ

Transmit/Receive Address				Baud Rate (16X Clock)	Theoretical Freq. (kHz)	Actual Freq. (kHz)	Percent Error	Duty Cycle %	Divisor
D	C	B	A						
0	0	0	0	50	0.8	0.8	—	50/50	3456
0	0	0	1	75	1.2	1.2	—	50/50	2304
0	0	1	0	110	1.76	1.76	-0.006	50/50	1571
0	0	1	1	134.5	2.152	2.152	-0.019	50/50	1285
0	1	0	0	150	2.4	2.4	—	50/50	1152
0	1	0	1	200	3.2	3.2	—	50/50	864
0	1	1	0	300	4.8	4.8	—	50/50	576
0	1	1	1	600	9.6	9.6	—	50/50	288
1	0	0	0	1200	19.2	19.2	—	50/50	144
1	0	0	1	1800	28.8	28.8	—	50/50	96
1	0	1	0	2000	32.0	32.15	+0.465	50/50	86
1	0	1	1	2400	38.4	38.4	—	50/50	72
1	1	0	0	3600	57.6	57.6	—	50/50	48
1	1	0	1	4800	76.8	76.8	—	50/50	36
1	1	1	0	9600	153.6	153.6	—	50/50	18
1	1	1	1	19,200	307.2	307.2	—	50/50	9

WD1943-02 or WD1945-02

TABLE 3. CRYSTAL FREQUENCY = 6.018305 MHZ

Transmit/Receive Address				Baud Rate (16X Clock)	Theoretical Freq. (kHz)	Actual Freq. (kHz)	Percent Error	Duty Cycle %	Divisor
D	C	B	A						
0	0	0	0	50	0.8	.7999	0	50/50	7523*
0	0	0	1	75	1.2	1.2000	0	50/50	5015*
0	0	1	0	110	1.76	1.7597	0	50/50	3420
0	0	1	1	134.5	2.152	2.1517	0	50/50	2797*
0	1	0	0	150	2.4	2.3996	0	50/50	2508
0	1	0	1	200	3.2	3.1995	0	50/50	1881*
0	1	1	0	300	4.8	4.7993	0	50/50	1254
0	1	1	1	600	9.6	9.5986	0	50/50	627*
1	0	0	0	1200	19.2	19.2279	+0.14	50/50	31.3*
1	0	0	1	1800	28.8	28.7959	0	50/50	209*
1	0	1	0	2000	32.0	32.0125	0	50/50	188
1	0	1	1	2400	38.4	38.3334	-0.17	50/50	157*
1	1	0	0	3600	57.6	57.8687	+0.46	50/50	104
1	1	0	1	4800	76.8	77.1583	+0.46	50/50	78
1	1	1	0	9800	153.6	154.3166	+0.46	50/50	39*
1	1	1	1	19,200	307.2	300.9175	-2.04	50/50	20

WD1943-03 or WD1945-03

TABLE 4. CLOCK FREQUENCY = 5.52960 MHZ

Transmit/Receive Address				Baud Rate (32X Clock)	Theoretical Freq. (kHz)	Actual Freq. (kHz)	Percent Error	Duty Cycle %	Divisor
D	C	B	A						
0	0	0	0	50	1.6	1.6	—	50/50	3456
0	0	0	1	75	2.4	2.4	—	50/50	2304
0	0	1	0	110	3.52	3.52	-0.006	50/50	1571
0	0	1	1	134.5	4.304	4.303	-0.019	50/50	1285
0	1	0	0	150	4.8	4.8	—	50/50	1152
0	1	0	1	200	6.4	6.4	—	50/50	864
0	1	1	0	300	9.6	9.6	—	50/50	576
0	1	1	1	600	19.2	19.2	—	50/50	288
1	0	0	0	1200	38.4	38.4	—	50/50	144
1	0	0	1	1800	57.6	57.6	—	50/50	96
1	0	1	0	2000	64.0	64.3	+0.465	50/50	86
1	0	1	1	2400	76.8	76.8	—	50/50	72
1	1	0	0	3600	115.2	115.2	—	50/50	48
1	1	0	1	4800	153.6	153.6	—	50/50	36
1	1	1	0	9600	307.2	307.2	—	50/50	18
1	1	1	1	19,200	614.4	614.4	—	50/50	9

WD1943-04 or WD1945-04

TABLE 5. CRYSTAL FREQUENCY = 4.9152 MHZ

Transmit/Receive Address				Baud Rate (16X Clock)	Theoretical Freq. (kHz)	Actual Freq. (kHz)	Percent Error	Duty Cycle %	Divisor
D	C	B	A						
0	0	0	0	50	0.8	0.8	—	50/50	6144
0	0	0	1	75	1.2	1.2	—	50/50	4096
0	0	1	0	110	1.76	1.7598	-0.01	*	2793
0	0	1	1	134.5	2.152	2.152	—	50/50	2284
0	1	0	0	150	2.4	2.4	—	50/50	2048
0	1	0	1	300	4.8	4.8	—	50/50	1024
0	1	1	0	600	9.6	9.6	—	50/50	512
0	1	1	1	1200	19.2	19.2	—	50/50	256
1	0	0	0	1800	28.8	28.7438	-0.19	*	171
1	0	0	1	2000	32.0	31.9168	-0.26	50/50	154
1	0	1	0	2400	38.4	38.4	—	50/50	128
1	0	1	1	3600	57.6	57.8258	0.39	*	85
1	1	0	0	4800	76.8	76.8	—	50/50	64
1	1	0	1	7200	115.2	114.306	-0.77	*	43
1	1	1	0	9600	153.6	153.6	—	50/50	32
1	1	1	1	19,200	307.2	307.2	—	50/50	16

WD1943-05 or WD1945-05

TABLE 6. CRYSTAL FREQUENCY = 5.0688 MHZ

Transmit/Receive Address				Baud Rate (32X Clock)	Theoretical Freq. (kHz)	Actual Freq. (kHz)	Percent Error	Duty Cycle %	Divisor
D	C	B	A						
0	0	0	0	50	1.6	1.6	—	50/50	3168
0	0	0	1	75	2.4	2.4	—	50/50	2112
0	0	1	0	110	3.52	3.52	—	50/50	1440
0	0	1	1	134.5	4.304	4.303	.026	50/50	1178
0	1	0	0	150	4.8	4.8	—	50/50	1056
0	1	0	1	200	6.4	6.4	—	50/50	792
0	1	1	0	300	9.6	9.6	—	50/50	528
0	1	1	1	600	19.2	19.2	—	50/50	264
1	0	0	0	1200	38.4	38.4	—	50/50	132
1	0	0	1	1800	57.6	57.6	—	50/50	88
1	0	1	0	2400	76.8	76.8	—	50/50	66
1	0	1	1	3600	115.2	115.2	—	50/50	44
1	1	0	0	4800	153.6	153.6	—	*	33
1	1	0	1	7200	230.4	230.4	—	50/50	22
1	1	1	0	9600	307.2	298.16	2.941	*	17
1	1	1	1	19,200	614.4	633.6	3.125	50/50	8

*When the duty cycle is not exactly 50% it is 50% ± 10%

WD1943-06 or WD1945-06

APPLICATIONS INFORMATION

OPERATION WITH A CRYSTAL

The WD1943/45 Baud Rate Generator may be driven by either a crystal or TTL level clock. When using a crystal, the waveform that appears at pins 1 (XTAL/EXT 1) and 18 (XTAL/EXT 2) does not conform to the normal TTL limits of $V_{IL} \leq 0.8V$ and $V_{IH} \geq 2.0V$. Figure 1 illustrates a typical crystal waveform when connected to a WD1943/45.

Since the D.C. level of the waveform causes the least positive point to typically be greater than 0.8V, the WD1943/45 is designed to look for an edge, as opposed to a TTL level. The XTAL/EXT logic triggers on a rising edge of typically 1V in magnitude. This allows the use of a crystal without any additional components.

OPERATIONS WITH TTL LEVEL CLOCK

With clock frequencies in the area of 5 MHz, significant overshoot and undershoot ("ringing") can appear at pins 1 and/or 18. The clock oscillator may, at times be triggered on a rising edge of an overshoot or undershoot waveform, causing the device to effectively "double-trigger." This phenomenon may result as a twice expected baud rate, or as an apparent device failure. Figure 2 shows a typical waveform that exhibits the "ringing" problem.

The design methods required to minimize ringing include the following:

1. Minimize the P.C. trace length. At 5 MHz, each inch of trace can add significantly to overshoot and undershoot.
2. Match impedances at both ends of the trace. For example, a series resistor near the device may be helpful.
3. A uniform impedance is important. This can be accomplished through the use of:
 - a. parallel ground lines
 - b. evenly spaced ground lines crossing the trace on the opposite side of PC board
 - c. an inner plane of ground, e.g., as in a four layered PC board.

In the event that ringing exists on an already finished board, several techniques can be used to reduce it. These are:

1. Add a series resistor to match impedance as shown in Figure 3.
2. Add pull-up/pull-down resistor to match impedance, as shown in Figure 4.
3. Add a high speed diode to clamp undershoot, as shown in Figure 5.

The method that is easiest to implement in many systems is method 1, the series resistor. The series resistor will cause the D.C. level to shift up, but that does not cause a problem since the OSC is triggered by an edge, as opposed to a TTL level.

The 1943/45 Baud Rate Generator can save both board space and cost in a communications system. By choosing either a crystal or a TTL level clock, the user can minimize the logic required to provide baud rate clocks in a given design.

POWER LINE SPIKES

Voltage transients on the AC power line may appear on the DC power output. If this possibility exists, it is suggested that a by-pass capacitor is used between +5V and GND.

CRYSTAL SPECIFICATIONS

User must specify termination (pin, wire, other)

Frequency — See Tables 1-6.

Temperature range 0°C to +70°C

Series resistance $\leq 50\Omega$

Series resonant

Overall tolerance $\pm 0.01\%$

CRYSTAL MANUFACTURERS (Partial List)

American Time Products Div.
Frequency Control Products, Inc.
61-20 Woodside Ave.
Woodside, New York 11377
(213) 458-5811

Bliley Electric Co.
2545 Grandview Blvd.
Erie, Pennsylvania 16508
(814) 838-3571

M-tron Ind. Inc.
P.O. Box 630
Yankton, South Dakota 57078
(605) 665-9321

Erie Frequency Control
453 Lincoln St.
Calisle, Pennsylvania 17013
(714) 249-2232

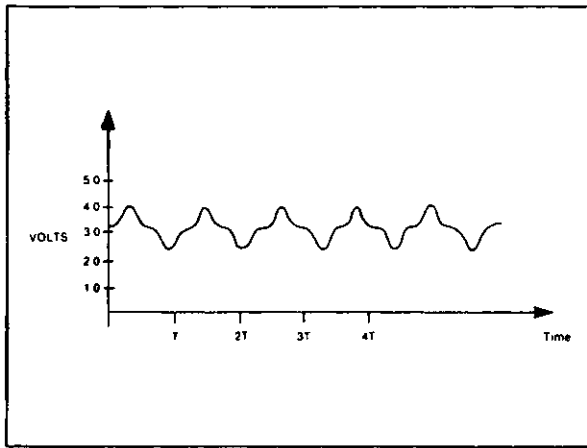


Figure 1. TYPICAL CRYSTAL WAVEFORM

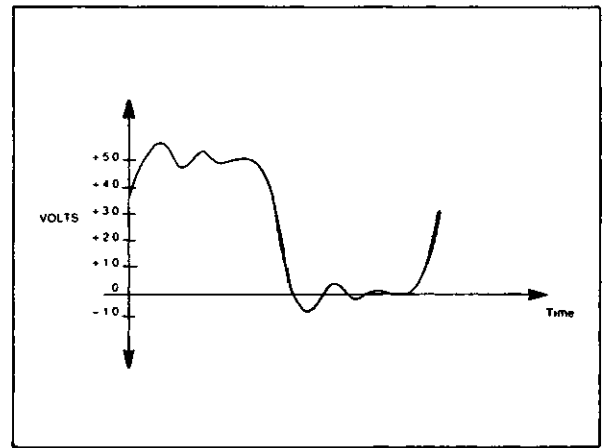


Figure 2. TYPICAL "RINGING" WAVEFORM from TTL INPUT

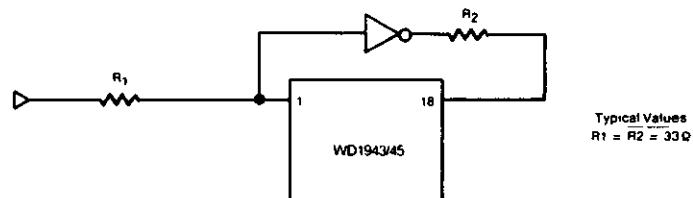


Figure 3. SERIES RESISTOR TO MATCH IMPEDANCE

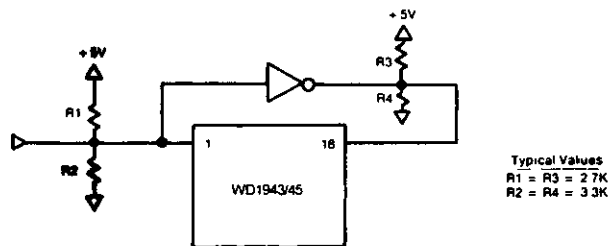


Figure 4. PULL-UP/PULL-DOWN RESISTORS TO MATCH IMPEDANCE

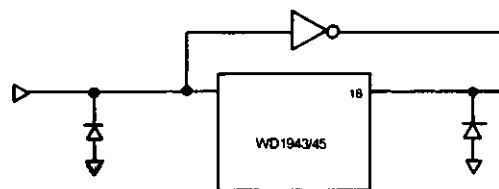


Figure 5. HIGH-SPEED DIODE TO CLAMP UNDERSHOOT

See page 725 for ordering information.

Information furnished by Western Digital Corporation is believed to be accurate and reliable. However, no responsibility is assumed by Western Digital Corporation for its use, nor for any infringements of patents or other rights of third parties which may result from its use. No license is granted by implication or otherwise under any patent or patent rights of Western Digital Corporation. Western Digital Corporation reserves the right to change specifications at anytime without notice.

WESTERN DIGITAL

C O R P O R A T I O N

FD179X-02

Floppy Disk Formatter/Controller Family

FEATURES

- TWO VFO CONTROL SIGNALS — RG & VFOE
- SOFT SECTOR FORMAT COMPATIBILITY
- AUTOMATIC TRACK SEEK WITH VERIFICATION
- ACCOMMODATES SINGLE AND DOUBLE DENSITY FORMATS
 - IBM 3740 Single Density (FM)
 - IBM System 34 Double Density (MFM)
 - Non IBM Format for Increased Capacity
- READ MODE
 - Single/Multiple Sector Read with Automatic Search or Entire Track Read
- WRITE MODE
 - Single/Multiple Sector Write with Automatic Sector Search
 - Entire Track Write for Diskette Formatting
- SYSTEM COMPATIBILITY
 - Double Buffering of Data 8 Bit Bi-Directional Bus for Data, Control and Status
 - DMA or Programmed Data Transfers
 - All Inputs and Outputs are TTL Compatible
 - On-Chip Track and Sector Registers/Comprehensive Status Information

PROGRAMMABLE CONTROLS

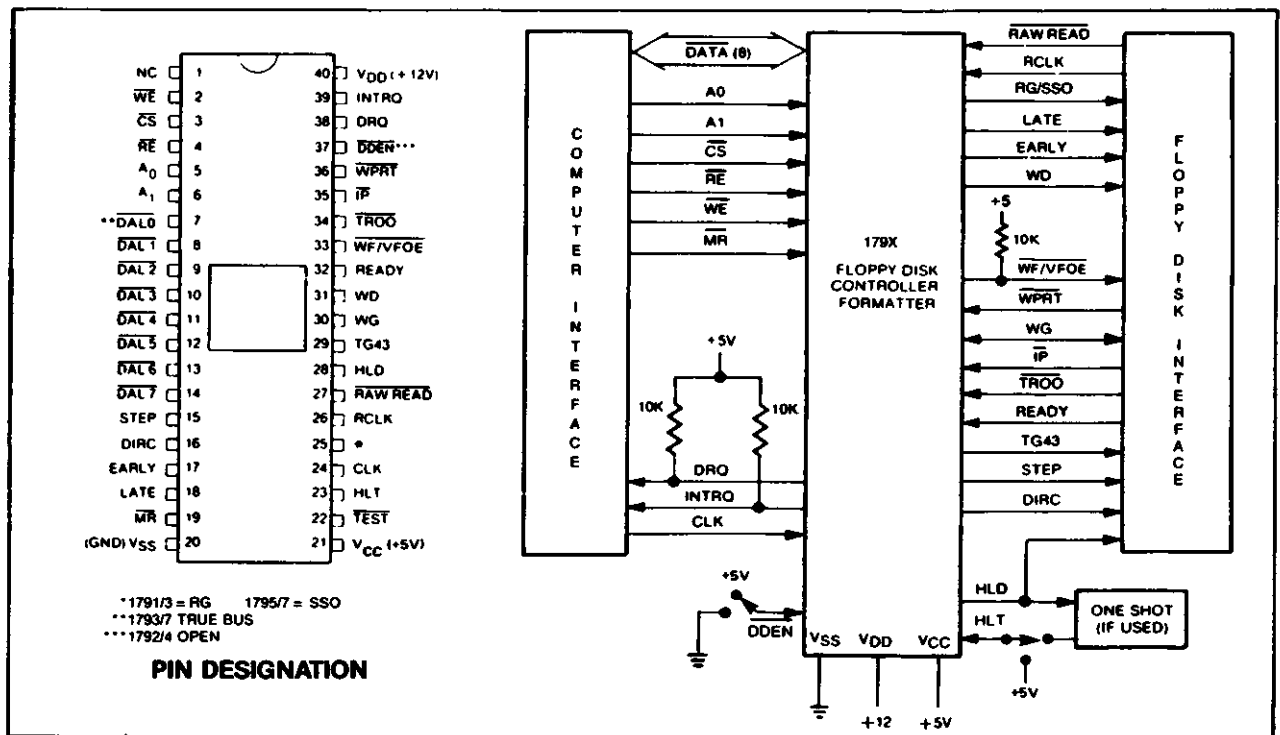
- Selectable Track to Track Stepping Time
- Side Select Compare
- INTERFACES TO WD1691 DATA SEPARATOR
- WINDOW EXTENSION
- INCORPORATES ENCODING/DECODING AND ADDRESS MARK CIRCUITRY
- FD1792/4 IS SINGLE DENSITY ONLY
- FD1795/7 HAS A SIDE SELECT OUTPUT

179X-02 FAMILY CHARACTERISTICS

FEATURES	1791	1792	1793	1794	1795	1797
Single Density (FM)	X	X	X	X	X	X
Double Density (MFM)	X		X		X	X
True Data Bus			X	X		X
Inverted Data Bus	X	X			X	
Write Precomp	X	X	X	X	X	X
Side Selection Output					X	X

APPLICATIONS

8" FLOPPY AND 5 1/4" MINI FLOPPY CONTROLLER
SINGLE OR DOUBLE DENSITY
CONTROLLER/FORMATTER



FD179X SYSTEM BLOCK DIAGRAM

PIN OUTS

PIN NUMBER	PIN NAME	SYMBOL	FUNCTION																									
1	NO CONNECTION	NC	Pin 1 is internally connected to a back bias generator and must be left open by the user.																									
19	MASTER RESET	\overline{MR}	A logic low (50 microseconds min.) on this input resets the device and loads HEX 03 into the command register. The Not Ready (Status Bit 7) is reset during \overline{MR} ACTIVE. When \overline{MR} is brought to a logic high a RESTORE Command is executed, regardless of the state of the Ready signal from the drive. Also, HEX 01 is loaded into sector register.																									
20	POWER SUPPLIES	Vss	Ground																									
21		Vcc	+5V \pm 5%																									
40		Vdd	+12V \pm 5%																									
COMPUTER INTERFACE:																												
2	WRITE ENABLE	\overline{WE}	A logic low on this input gates data on the DAL into the selected register when \overline{CS} is low.																									
3	CHIP SELECT	\overline{CS}	A logic low on this input selects the chip and enables computer communication with the device.																									
4	READ ENABLE	\overline{RE}	A logic low on this input controls the placement of data from a selected register on the DAL when \overline{CS} is low.																									
5,6	REGISTER SELECT LINES	A0, A1	These inputs select the register to receive/transfer data on the DAL lines under \overline{RE} and \overline{WE} control: <table><tr><td>\overline{CS}</td><td>A1</td><td>A0</td><td>\overline{RE}</td><td>\overline{WE}</td></tr><tr><td>0</td><td>0</td><td>0</td><td>Status Reg</td><td>Command Reg</td></tr><tr><td>0</td><td>0</td><td>1</td><td>Track Reg</td><td>Track Reg</td></tr><tr><td>0</td><td>1</td><td>0</td><td>Sector Reg</td><td>Sector Reg</td></tr><tr><td>0</td><td>1</td><td>1</td><td>Data Reg</td><td>Data Reg</td></tr></table>	\overline{CS}	A1	A0	\overline{RE}	\overline{WE}	0	0	0	Status Reg	Command Reg	0	0	1	Track Reg	Track Reg	0	1	0	Sector Reg	Sector Reg	0	1	1	Data Reg	Data Reg
\overline{CS}	A1	A0	\overline{RE}	\overline{WE}																								
0	0	0	Status Reg	Command Reg																								
0	0	1	Track Reg	Track Reg																								
0	1	0	Sector Reg	Sector Reg																								
0	1	1	Data Reg	Data Reg																								
7-14	DATA ACCESS LINES	DAL0-DAL7	Eight bit Bidirectional bus used for transfer of data, control, and status. This bus is receiver enabled by \overline{WE} or transmitter enabled by \overline{RE} . Each line will drive 1 standard TTL load.																									
24	CLOCK	CLK	This input requires a free-running 50% duty cycle square wave clock for internal timing reference, 2 MHz \pm 1% for 8" drives, 1 MHz \pm 1% for mini-floppies.																									
38	DATA REQUEST	DRQ	This open drain output indicates that the DR contains assembled data in Read operations, or the DR is empty in Write operations. This signal is reset when serviced by the computer through reading or loading the DR in Read or Write operations, respectively. Use 10K pull-up resistor to +5.																									
39	INTERRUPT REQUEST	INTRQ	This open drain output is set at the completion of any command and is reset when the STATUS register is read or the command register is written to. Use 10K pull-up resistor to +5.																									
FLOPPY DISK INTERFACE:																												
15	STEP	STEP	The step output contains a pulse for each step.																									
16	DIRECTION	DIRC	Direction Output is active high when stepping in, active low when stepping out.																									
17	EARLY	EARLY	Indicates that the WRITE DATA pulse occurring while Early is active (high) should be shifted early for write precompensation.																									
18	LATE	LATE	Indicates that the write data pulse occurring while Late is active (high) should be shifted late for write precompensation.																									

PIN NUMBER	PIN NAME	SYMBOL	FUNCTION
22	TEST	TEST	This input is used for testing purposes only and should be tied to +5V or left open by the user unless interfacing to voice coil actuated steppers.
23	HEAD LOAD TIMING	HLT	When a logic high is found on the HLT input the head is assumed to be engaged. It is typically derived from a 1 shot triggered by HLD.
25	READ GATE (1791, 1792, 1793, 1794)	RG	This output is used for synchronization of external data separators. The output goes high after two Bytes of zeros in single density, or 4 Bytes of either zeros or ones in double density operation.
25	SIDE SELECT OUTPUT (1795, 1797)	SSO	The logic level of the Side Select Output is directly controlled by the 'S' flag in Type II or III commands. When U = 1, SSO is set to a logic 1. When U = 0, SSO is set to a logic 0. The SSO is compared with the side information in the Sector I.D. Field. If they do not compare Status Bit 4 (RNF) is set. The Side Select Output is only updated at the beginning of a Type II or III command. It is forced to a logic 0 upon a MASTER RESET condition.
26	READ CLOCK	RCLK	A nominal square-wave clock signal derived from the data stream must be provided to this input. Phasing (i.e. RCLK transitions) relative to RAW READ is important but polarity (RCLK high or low) is not.
27	RAW READ	RAW READ	The data input signal directly from the drive. This input shall be a negative pulse for each recorded flux transition.
28	HEAD LOAD	HLD	The HLD output controls the loading of the Read-Write head against the media.
29	TRACK GREATER THAN 43	TG43	This output informs the drive that the Read/Write head is positioned between tracks 44-76. This output is valid only during Read and Write Commands.
30	WRITE GATE	WG	This output is made valid before writing is to be performed on the diskette.
31	WRITE DATA	WD	A 200 ns (MFM) or 500 ns (FM) output pulse per flux transition. WD contains the unique Address marks as well as data and clock in both FM and MFM formats.
32	READY	READY	This input indicates disk readiness and is sampled for a logic high before Read or Write commands are performed. If Ready is low the Read or Write operation is not performed and an interrupt is generated. Type I operations are performed regardless of the state of Ready. The Ready input appears in inverted format as Status Register bit 7.
33	WRITE FAULT VFO ENABLE	WF/VFOE	This is a bi-directional signal used to signify writing faults at the drive, and to enable the external PLO data separator. When WG = 1, Pin 33 functions as a WF input. If WF = 0, any write command will immediately be terminated. When WG = 0, Pin 33 functions as a VFOE output. VFOE will go low during a read operation after the head has loaded and settled (HLT = 1). On the 1795/7, it will remain low until the last bit of the second CRC byte in the ID field. VFOE will then go high until 8 bytes (MFM) or 4 bytes (FM) before the Address Mark. It will then go active until the last bit of the second CRC byte of the Data Field. On the 1791/3, VFOE will remain low until the end of the Data Field. This pin has an internal 100K Ohm pull-up resistor.
34	TRACK 00	TR00	This input informs the FD179X that the Read/Write head is positioned over Track 00.

PIN NUMBER	PIN NAME	SYMBOL	FUNCTION
35	INDEX PULSE	\overline{IP}	This input informs the FD179X when the index hole is encountered on the diskette.
36	WRITE PROTECT	\overline{WPRT}	This input is sampled whenever a Write Command is received. A logic low terminates the command and sets the Write Protect Status bit.
37	DOUBLE DENSITY	\overline{DDEN}	This input pin selects either single or double density operation. When $\overline{DDEN} = 0$, double density is selected. When $\overline{DDEN} = 1$, single density is selected. This line must be left open on the 1792/4.

GENERAL DESCRIPTION

The FD179X are N-Channel Silicon Gate MOS LSI devices which perform the functions of a Floppy Disk Formatter/Controller in a single chip implementation. The FD179X, which can be considered the end result of both the FD1771 and FD1781 designs, is IBM 3740 compatible in single density mode (FM) and System 34 compatible in Double Density Mode (MFM). The FD179X contains all the features of its predecessor the FD1771, plus the added features necessary to read/write and format a double density diskette. These include address mark detection, FM and MFM encode and decode logic, window extension, and write precompensation. In order to maintain compatibility, the FD1771, FD1781, and FD179X designs were made as close as possible with the computer interface, instruction set, and I/O registers being identical. Also, head load control is identical. In each case, the actual pin assignments vary by only a few pins from any one to another.

The processor interface consists of an 8-bit bi-directional bus for data, status, and control word transfers. The FD179X is set up to operate on a multiplexed bus with other bus-oriented devices.

The FD179X is TTL compatible on all inputs and outputs. The outputs will drive ONE TTL load or three LS loads. The 1793 is identical to the 1791 except the DAL lines are TRUE for systems that utilize true data busses.

The 1795/7 has a side select output for controlling double sided drives, and the 1792 and 1794 are "Single Density Only" versions of the 1791 and 1793 respectively. On these devices, \overline{DDEN} must be left open.

ORGANIZATION

The Floppy Disk Formatter block diagram is illustrated on page 5. The primary sections include the parallel processor interface and the Floppy Disk interface.

Data Shift Register — This 8-bit register assembles serial data from the Read Data input ($\overline{RAW READ}$) during Read operations and transfers serial data to the Write Data output during Write operations.

Data Register — This 8-bit register is used as a holding register during Disk Read and Write operations. In Disk Read operations the assembled data byte is transferred in parallel to the Data Register from the Data Shift Register. In Disk Write operations information is transferred in parallel from the Data Register to the Data Shift Register.

When executing the Seek command the Data Register holds the address of the desired Track position. This register is loaded from the DAL and gated onto the DAL under processor control.

Track Register — This 8-bit register holds the track number of the current Read/Write head position. It is incremented by one every time the head is stepped in (towards track 76) and decremented by one when the head is stepped out (towards track 00). The contents of the register are compared with the recorded track number in the ID field during disk Read, Write, and Verify operations. The Track Register can be loaded from or transferred to the DAL. This Register should not be loaded when the device is busy.

Sector Register (SR) — This 8-bit register holds the address of the desired sector position. The contents of the register are compared with the recorded sector number in the ID field during disk Read or Write operations. The Sector Register contents can be loaded from or transferred to the DAL. This register should not be loaded when the device is busy.

Command Register (CR) — This 8-bit register holds the command presently being executed. This register should not be loaded when the device is busy unless the new command is a force interrupt. The command register can be loaded from the DAL, but not read onto the DAL.

Status Register (STR) — This 8-bit register holds device Status information. The meaning of the Status bits is a function of the type of command previously executed. This register can be read onto the DAL, but not loaded from the DAL.

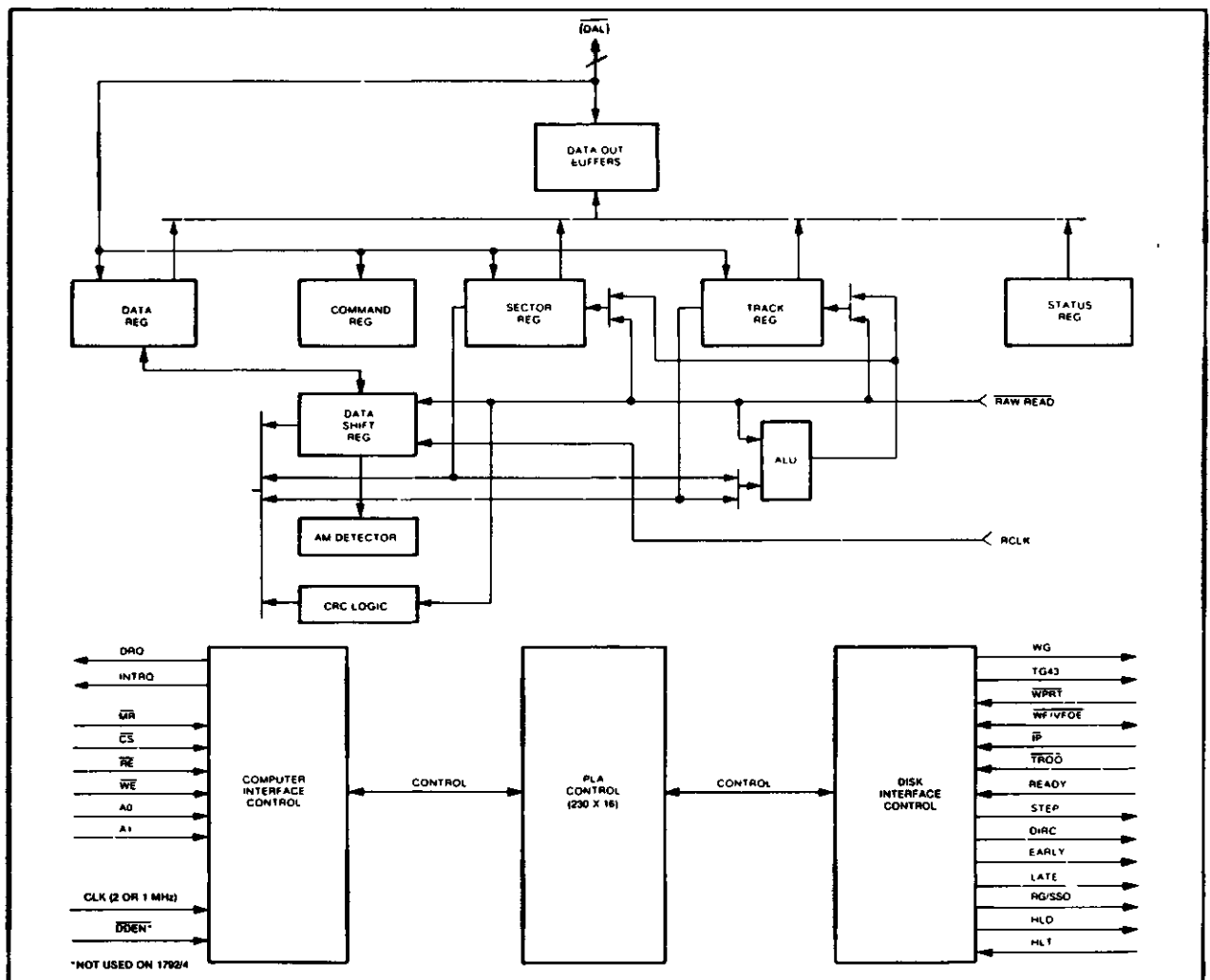
CRC Logic — This logic is used to check or to generate the 16-bit Cyclic Redundancy Check (CRC). The polynomial is: $G(x) = x^{16} + x^{12} + x^5 + 1$.

The CRC includes all information starting with the address mark and up to the CRC characters. The CRC register is preset to ones prior to data being shifted through the circuit.

Arithmetic/Logic Unit (ALU) — The ALU is a serial comparator, incrementer, and decrementer and is used for register modification and comparisons with the disk recorded ID field.

Timing and Control — All computer and Floppy Disk Interface controls are generated through this logic. The internal device timing is generated from an external crystal clock.

The FD179X has two different modes of operation according to the state of \overline{DDEN} . When $\overline{DDEN} = 0$ double density (MFM) is assumed. When $\overline{DDEN} = 1$, single



FD179X BLOCK DIAGRAM

density (FM) is assumed. 1792 & 1794 are single density only.

AM Detector — The address mark detector detects ID, data and index address marks during read and write operations.

PROCESSOR INTERFACE

The interface to the processor is accomplished through the eight Data Access Lines (\overline{DAL}) and associated control signals. The \overline{DAL} are used to transfer Data, Status, and Control words out of, or into the FD179X. The \overline{DAL} are three state buffers that are enabled as output drivers when Chip Select (\overline{CS}) and Read Enable (\overline{RE}) are active (low logic state) or act as input receivers when \overline{CS} and Write Enable (\overline{WE}) are active.

When transfer of data with the Floppy Disk Controller is required by the host processor, the device address is decoded and \overline{CS} is made low. The address bits A1 and A0, combined with the signals \overline{RE} during a Read operation or \overline{WE} during a Write operation are interpreted as selecting the following registers:

A1 - A0	READ (\overline{RE})	WRITE (\overline{WE})
0 0	Status Register	Command Register
0 1	Track Register	Track Register
1 0	Sector Register	Sector Register
1 1	Data Register	Data Register

During Direct Memory Access (DMA) types of data transfers between the Data Register of the FD179X and the processor, the Data Request (DRQ) output is used in Data Transfer control. This signal also appears as status bit 1 during Read and Write operations.

On Disk Read operations the Data Request is activated (set high) when an assembled serial input byte is transferred in parallel to the Data Register. This bit is cleared when the Data Register is read by the processor. If the Data Register is read after one or more characters are lost, by having new data transferred into the register prior to processor readout, the Lost Data bit is set in the Status Register. The Read operation continues until the end of sector is reached.

On Disk Write operations the data Request is activated when the Data Register transfers its contents to the Data

Shift Register, and requires a new data byte. It is reset when the Data Register is loaded with new data by the processor. If new data is not loaded at the time the next serial byte is required by the Floppy Disk, a byte of zeroes is written on the diskette and the Lost Data bit is set in the Status Register.

At the completion of every command an INTRQ is generated. INTRQ is reset by either reading the status register or by loading the command register with a new command. In addition, INTRQ is generated if a Force Interrupt command condition is met.

The 179X has two modes of operation according to the state of $\overline{\text{DDEN}}$ (Pin 37). When $\overline{\text{DDEN}} = 1$, single density is selected. In either case, the CLK input (Pin 24) is at 2 MHz. However, when interfacing with the mini-floppy, the CLK input is set at 1 MHz for both single density and double density.

GENERAL DISK READ OPERATIONS

Sector lengths of 128, 256, 512 or 1024 are obtainable in either FM or MFM formats. For FM, $\overline{\text{DDEN}}$ should be placed to logical "1." For MFM formats, $\overline{\text{DDEN}}$ should be placed to a logical "0." Sector lengths are determined at format time by the fourth byte in the "ID" field.

Sector Length Table*	
Sector Length Field (hex)	Number of Bytes in Sector (decimal)
00	128
01	256
02	512
03	1024

*1795/97 may vary — see command summary.

The number of sectors per track as far as the FD179X is concerned can be from 1 to 255 sectors. The number of tracks as far as the FD179X is concerned is from 0 to 255 tracks. For IBM 3740 compatibility, sector lengths are 128 bytes with 26 sectors per track. For System 34 compatibility (MFM), sector lengths are 256 bytes/sector with 26 sectors/track; or lengths of 1024 bytes/sector with 8 sectors/track. (See Sector Length Table)

For read operations in 8" double density the FD179X requires RAW READ Data (Pin 27) signal which is a 200 ns pulse per flux transition and a Read clock (RCLK) signal to indicate flux transition spacings. The RCLK (Pin 26) signal is provided by some drives but if not it may be derived externally by Phase lock loops, one shots, or counter techniques. In addition, a Read Gate Signal is provided as an output (Pin 25) on 1791/92/93/94 which can be used to inform phase lock loops when to acquire synchronization. When reading from the media in FM, RG is made true when 2 bytes of zeroes are detected. The FD179X must find an address mark within the next 10 bytes; otherwise RG is reset and the search for 2 bytes of zeroes begins all over again. If an address mark is found within 10 bytes, RG remains true as long as the FD179X is deriving any useful information from the data stream. Similarly for MFM, RG is made active when 4 bytes of "00" or "FF" are detected. The FD179X must find an address mark within the next 16 bytes, otherwise RG is reset and search resumes.

During read operations ($\text{WG} = 0$), the $\overline{\text{VFOE}}$ (Pin 33) is provided for phase lock loop synchronization. $\overline{\text{VFOE}}$ will go active low when:

- Both HLT and HLD are True
- Settling Time, if programmed, has expired
- The 179X is inspecting data off the disk

If $\overline{\text{WF/VFOE}}$ is not used, leave open or tie to a 10K resistor to +5.

GENERAL DISK WRITE OPERATION

When writing is to take place on the diskette the Write Gate (WG) output is activated, allowing current to flow into the Read/Write head. As a precaution to erroneous writing the first data byte must be loaded into the Data Register in response to a Data Request from the FD179X before the Write Gate signal can be activated.

Writing is inhibited when the Write Protect input is a logic low, in which case any Write command is immediately terminated, an interrupt is generated and the Write Protect status bit is set. The Write Fault input, when activated, signifies a writing fault condition detected in disk drive electronics such as failure to detect write current flow when the Write Gate is activated. On detection of this fault the FD179X terminates the current command, and sets the Write Fault bit (bit 5) in the Status Word. The Write Fault input should be made inactive when the Write Gate output becomes inactive.

For write operations, the FD179X provides Write Gate (Pin 30) and Write Data (Pin 31) outputs. Write data consists of a series of 500 ns pulses in FM ($\overline{\text{DDEN}} = 1$) and 200 ns pulses in MFM ($\overline{\text{DDEN}} = 0$). Write Data provides the unique address marks in both formats.

Also during write, two additional signals are provided for write precompensation. These are EARLY (Pin 17) and LATE (Pin 18). EARLY is active true when the WD pulse appearing on (Pin 30) is to be written EARLY. LATE is active true when the WD pulse is to be written LATE. If both EARLY and LATE are low when the WD pulse is present, the WD pulse is to be written at nominal. Since write precompensation values vary from disk manufacturer to disk manufacturer, the actual value is determined by several one shots or delay lines which are located external to the FD179X. The write precompensation signals EARLY and LATE are valid for the duration of WD in both FM and MFM formats.

READY

Whenever a Read or Write command (Type II or III) is received the FD179X samples the Ready input. If this input is logic low the command is not executed and an interrupt is generated. All Type I commands are performed regardless of the state of the Ready input. Also, whenever a Type II or III command is received, the TG43 signal output is updated.

COMMAND DESCRIPTION

The FD179X will accept eleven commands. Command words should only be loaded in the Command Register when the Busy status bit is off (Status bit 0). The one exception is the Force Interrupt command. Whenever a command is being executed, the Busy status bit is set. When a command is completed, an interrupt is generated and the Busy status bit is reset. The Status Register indicates whether the completed command encountered an error or was fault free. For ease of discussion, commands are divided into four types. Commands and types are summarized in Table 1.

TABLE 1. COMMAND SUMMARY

A. Commands for Models: 1791, 1792, 1793, 1794

B. Commands for Models: 1795, 1797

Type	Command	Bits								Bits							
		7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0
I	Restore	0	0	0	0	h	V	r ₁	r ₀	0	0	0	0	h	V	r ₁	r ₀
I	Seek	0	0	0	1	h	V	r ₁	r ₀	0	0	0	1	h	V	r ₁	r ₀
I	Step	0	0	1	T	h	V	r ₁	r ₀	0	0	1	T	h	V	r ₁	r ₀
I	Step-in	0	1	0	T	h	V	r ₁	r ₀	0	1	0	T	h	V	r ₁	r ₀
I	Step-out	0	1	1	T	h	V	r ₁	r ₀	0	1	1	T	h	V	r ₁	r ₀
II	Read Sector	1	0	0	m	S	E	C	0	1	0	0	m	L	E	U	0
II	Write Sector	1	0	1	m	S	E	C	a ₀	1	0	1	m	L	E	U	a ₀
III	Read Address	1	1	0	0	0	E	0	0	1	1	0	0	0	E	U	0
III	Read Track	1	1	1	0	0	E	0	0	1	1	1	0	0	E	U	0
III	Write Track	1	1	1	1	0	E	0	0	1	1	1	1	0	E	U	0
IV	Force Interrupt	1	1	0	1	l ₃	l ₂	l ₁	l ₀	1	1	0	1	l ₃	l ₂	l ₁	l ₀

FLAG SUMMARY

TABLE 2. FLAG SUMMARY

Command Type	Bit No(s)	Description																					
I	0, 1	r ₁ r ₀ = Stepping Motor Rate See Table 3 for Rate Summary																					
I	2	V = Track Number Verify Flag	V = 0, No verify V = 1, Verify on destination track																				
I	3	h = Head Load Flag	h = 1, Load head at beginning h = 0, Unload head at beginning																				
I	4	T = Track Update Flag	T = 0, No update T = 1, Update track register																				
II	0	a ₀ = Data Address Mark	a ₀ = 0, FB (DAM) a ₀ = 1, F8 (deleted DAM)																				
II	1	C = Side Compare Flag	C = 0, Disable side compare C = 1, Enable side compare																				
II & III	1	U = Update SSO	U = 0, Update SSO to 0 U = 1, Update SSO to 1																				
II & III	2	E = 15 MS Delay	E = 0, No 15 MS delay E = 1, 15 MS delay																				
II	3	S = Side Compare Flag	S = 0, Compare for side 0 S = 1, Compare for side 1																				
II	3	L = Sector Length Flag	<table><tr><th colspan="5">LSB's Sector Length in ID Field</th></tr><tr><th></th><th>00</th><th>01</th><th>10</th><th>11</th></tr><tr><td>L = 0</td><td>256</td><td>512</td><td>1024</td><td>128</td></tr><tr><td>L = 1</td><td>128</td><td>256</td><td>512</td><td>1024</td></tr></table>	LSB's Sector Length in ID Field						00	01	10	11	L = 0	256	512	1024	128	L = 1	128	256	512	1024
LSB's Sector Length in ID Field																							
	00	01	10	11																			
L = 0	256	512	1024	128																			
L = 1	128	256	512	1024																			
II	4	m = Multiple Record Flag	m = 0, Single record m = 1, Multiple records																				
IV	0-3	l _x = Interrupt Condition Flags l ₀ = 1 Not Ready To Ready Transition l ₁ = 1 Ready To Not Ready Transition l ₂ = 1 Index Pulse l ₃ = 1 Immediate Interrupt, Requires A Reset l ₃ -l ₀ = 0 Terminate With No Interrupt (INTRQ)																					

*NOTE: See Type IV Command Description for further information.

TYPE I COMMANDS

The Type I Commands include the Restore, Seek, Step, Step-In, and Step-Out commands. Each of the Type I Commands contains a rate field (R0 R1), which determines the stepping motor rate as defined in Table 3.

A 2 μ s (MFM) or 4 μ s (FM) pulse is provided as an output to the drive. For every step pulse issued, the drive moves one track location in a direction determined by the direction output. The chip will step the drive in the same direction it last stepped unless the command changes the direction.

The Direction signal is active high when stepping in and low when stepping out. The Direction signal is valid 12 μ s before the first stepping pulse is generated.

The rates (shown in Table 3) can be applied to a Step-Direction Motor through the device interface.

TABLE 3. STEPPING RATES

CLK	2 MHz	2 MHz	1 MHz	1 MHz	2 MHz	1 MHz
DDEN	0	1	0	1	X	X
R1 R0	TEST=1	TEST=1	TEST=1	TEST=1	TEST=0	TEST=0
0 0	3 ms	3 ms	6 ms	6 ms	184 μ s	368 μ s
0 1	6 ms	6 ms	12 ms	12 ms	190 μ s	380 μ s
1 0	10 ms	10 ms	20 ms	20 ms	198 μ s	396 μ s
1 1	15 ms	15 ms	30 ms	30 ms	208 μ s	416 μ s

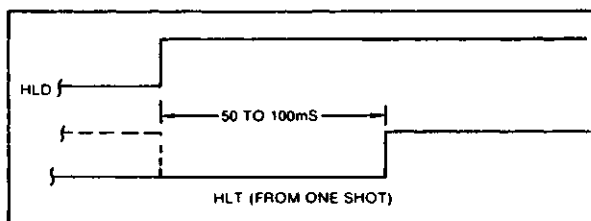
After the last directional step an additional 15 milliseconds of head settling time takes place if the Verify flag is set in Type I commands. Note that this time doubles to 30 ms for a 1 MHz clock. If $\overline{\text{TEST}} = 0$, there is zero settling time. There is also a 15 ms head settling time if the E flag is set in any Type II or III command.

When a Seek, Step or Restore command is executed an optional verification of Read-Write head position can be performed by settling bit 2 ($V = 1$) in the command word to a logic 1. The verification operation begins at the end of the 15 millisecond settling time after the head is loaded against the media. The track number from the first encountered ID Field is compared against the contents of the Track Register. If the track numbers compare and the ID Field Cyclic Redundancy Check (CRC) is correct, the verify operation is complete and an INTRQ is generated with no errors. If there is a match but not a valid CRC, the CRC error status bit is set (Status bit 3), and the next encountered ID field is read from the disk for the verification operation.

The FD179X must find an ID field with correct track number and correct CRC within 5 revolutions of the media; otherwise the seek error is set and an INTRQ is generated. If $V = 0$, no verification is performed.

The Head Load (HLD) output controls the movement of the read/write head against the media. HLD is activated at the beginning of a Type I command if the h flag is set ($h = 1$), at the end of the Type I command if the verify flag ($V = 1$), or upon receipt of any Type II or III command. Once HLD is active it remains active until either a Type I command is received with ($h = 0$ and $V = 0$); or if the FD179X is in an idle state (non-busy) and 15 index pulses have occurred.

Head Load timing (HLT) is an input to the FD179X which is used for the head engage time. When $\text{HLT} = 1$, the FD179X assumes the head is completely engaged. The head engage time is typically 30 to 100 ms depending on drive. The low to high transition on HLD is typically used to fire a one shot. The output of the one shot is then used for HLT and supplied as an input to the FD179X.



HEAD LOAD TIMING

When both HLD and HLT are true, the FD179X will then read from or write to the media. The "and" of HLD and HLT appears as status Bit 5 in Type I status.

In summary for the Type I commands: if $h = 0$ and $V = 0$, HLD is reset. If $h = 1$ and $V = 0$, HLD is set at the beginning of the command and HLT is not sampled nor is there an internal 15 ms delay. If $h = 0$ and $V = 1$, HLD is set near the end of the command, an internal 15 ms occurs, and the FD179X waits for HLT to be true. If $h = 1$ and $V = 1$, HLD is set at the beginning of the command. Near the end of the command, after all the steps have been issued, an internal 15 ms delay occurs and the FD179X then waits for HLT to occur.

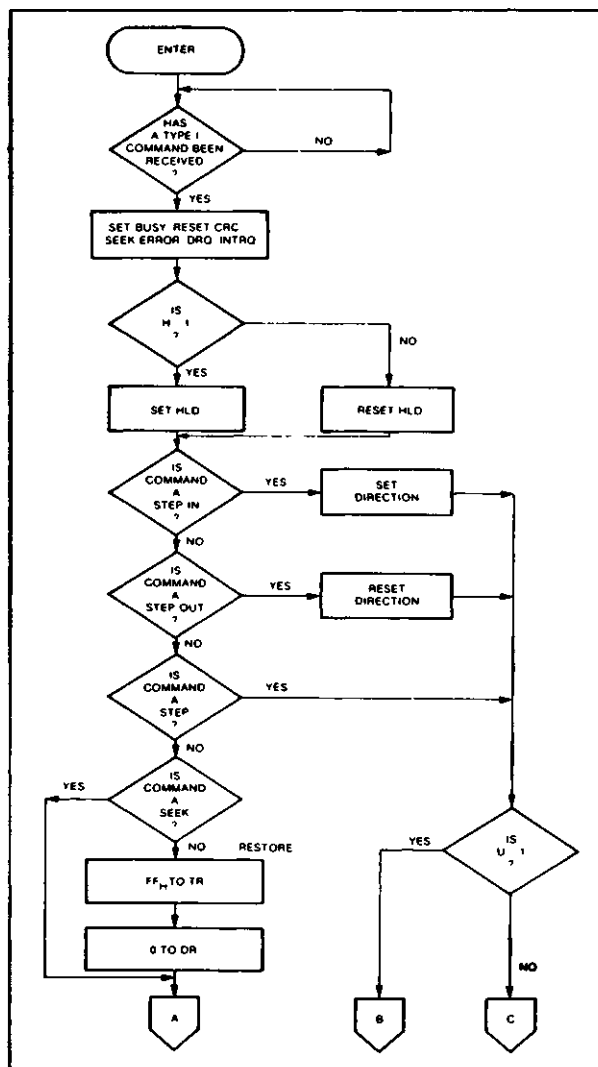
For Type II and III commands with E flag off, HLD is made active and HLT is sampled until true. With E flag on, HLD is made active, an internal 15 ms delay occurs and then HLT is sampled until true.

RESTORE (SEEK TRACK 0)

Upon receipt of this command the Track 00 ($\overline{\text{TR00}}$) input is sampled. If $\overline{\text{TR00}}$ is active low indicating the Read-Write head is positioned over track 0, the Track Register is loaded with zeroes and an interrupt is generated. If $\overline{\text{TR00}}$ is not active low, stepping pulses (pins 15 to 16) at a rate specified by the R1 R0 field are issued until the $\overline{\text{TR00}}$ input is activated. At this time the Track Register is loaded with zeroes and an interrupt is generated. If the $\overline{\text{TR00}}$ input does not go active low after 255 stepping pulses, the FD179X terminates operation, interrupts, and sets the Seek error status bit, providing the V flag is set. A verification operation also takes place if the V flag is set. The h bit allows the head to be loaded at the start of command. Note that the Restore command is executed when $\overline{\text{MR}}$ goes from an active to an inactive state and that the DRQ pin stays low.

SEEK

This command assumes that the Track Register contains the track number of the current position of the Read-Write head and the Data Register contains the desired track number. The FD179X will update the Track register and issue stepping pulses in the appropriate direction until the contents of the Track register are equal to the contents of



TYPE I COMMAND FLOW

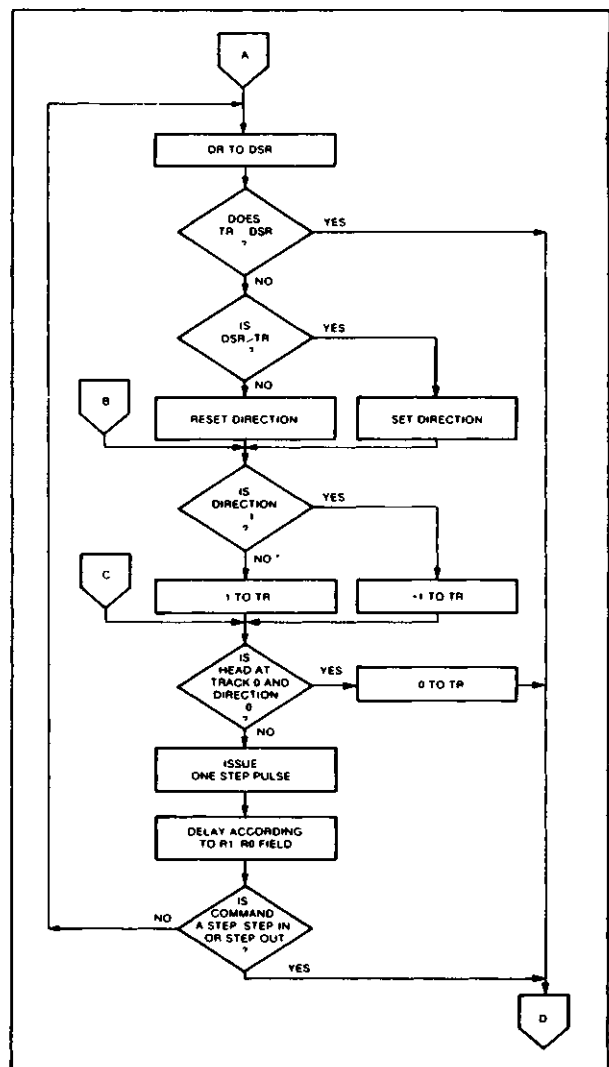
the Data Register (the desired track location). A verification operation takes place if the V flag is on. The h bit allows the head to be loaded at the start of the command. An interrupt is generated at the completion of the command. Note: When using multiple drives, the track register must be updated for the drive selected before seeks are issued.

STEP

Upon receipt of this command, the FD179X issues one stepping pulse to the disk drive. The stepping motor direction is the same as in the previous step command. After a delay determined by the T10 field, a verification takes place if the V flag is on. If the U flag is on, the Track Register is updated. The h bit allows the head to be loaded at the start of the command. An interrupt is generated at the completion of the command.

STEP-IN

Upon receipt of this command, the FD179X issues one stepping pulse in the direction towards track 76. If the U



TYPE I COMMAND FLOW

flag is on, the Track Register is incremented by one. After a delay determined by the T10 field, a verification takes place if the V flag is on. The h bit allows the head to be loaded at the start of the command. An interrupt is generated at the completion of the command.

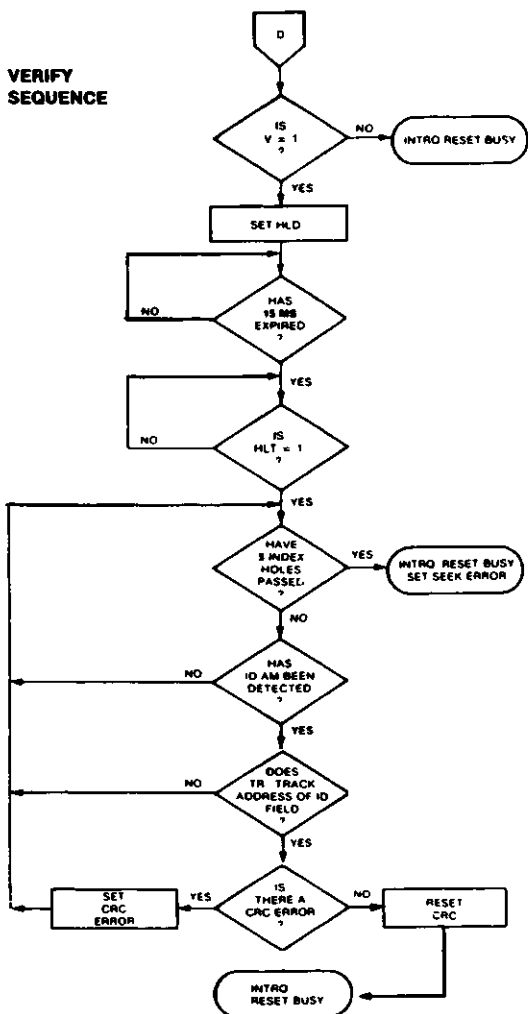
STEP-OUT

Upon receipt of this command, the FD179X issues one stepping pulse in the direction towards track 0. If the U flag is on, the Track Register is decremented by one. After a delay determined by the T10 field, a verification takes place if the V flag is on. The h bit allows the head to be loaded at the start of the command. An interrupt is generated at the completion of the command.

EXCEPTIONS

On the 1795/7 devices, the SSO output is not affected during Type 1 commands, and an internal side compare does not take place when the (V) Verify Flag is on.

VERIFY SEQUENCE



NOTE IF TEST = 0, THERE IS NO 15MS DELAY
IF TEST = 1 AND CLK = 1 MHZ, THERE IS A 30MS DELAY

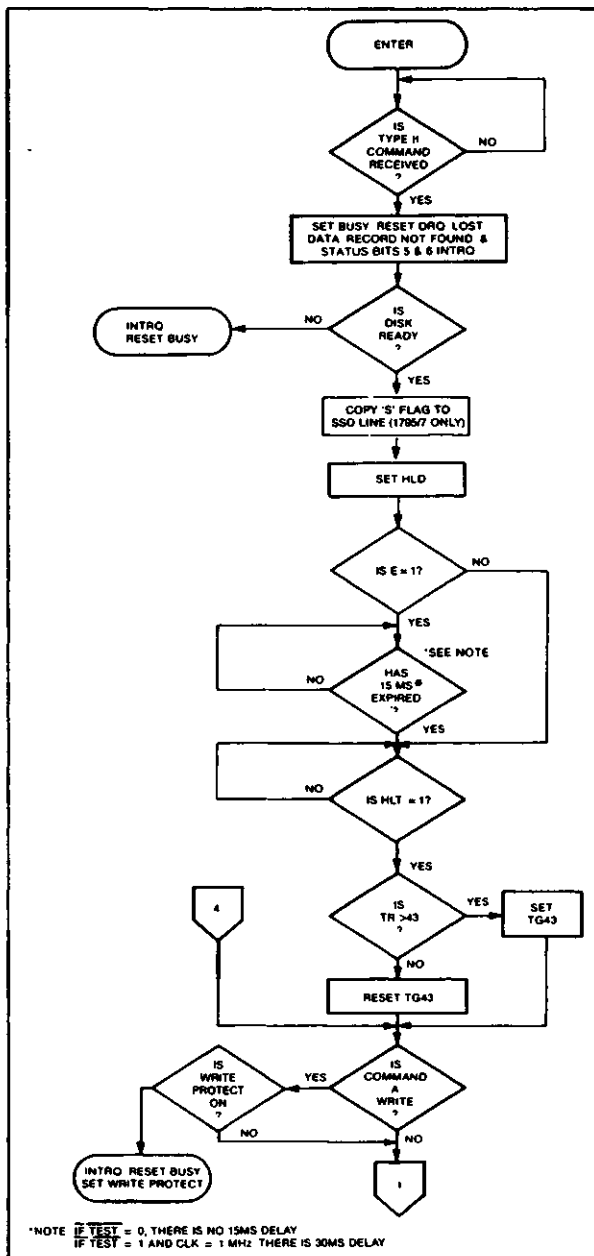
TYPE I COMMAND FLOW

TYPE II COMMANDS

The Type II Commands are the Read Sector and Write Sector commands. Prior to loading the Type II Command into the Command Register, the computer must load the Sector Register with the desired sector number. Upon receipt of the Type II command, the busy status Bit is set. If the E flag = 1 (this is the normal case) HLD is made active and HLT is sampled after a 15 msec delay. If the E flag is 0, the head is loaded and HLT sampled with no 15 msec delay. The ID field and Data Field format are shown on page 13.

When an ID field is located on the disk, the FD179X compares the Track Number on the ID field with the Track Register. If there is not a match, the next encountered ID field is read and a comparison is again made. If there was a match, the Sector Number of the ID field is compared with the Sector Register. If there is not a Sector match, the next encountered ID field is read off the disk and comparisons again made. If the ID field CRC is correct, the data field is

then located and will be either written into, or read from depending upon the command. The FD179X must find an ID field with a Track number, Sector number, side number, and CRC within four revolutions of the disk; otherwise, the Record not found status bit is set (Status bit 3) and the command is terminated with an interrupt.



*NOTE IF TEST = 0, THERE IS NO 15MS DELAY
IF TEST = 1 AND CLK = 1 MHZ THERE IS 30MS DELAY

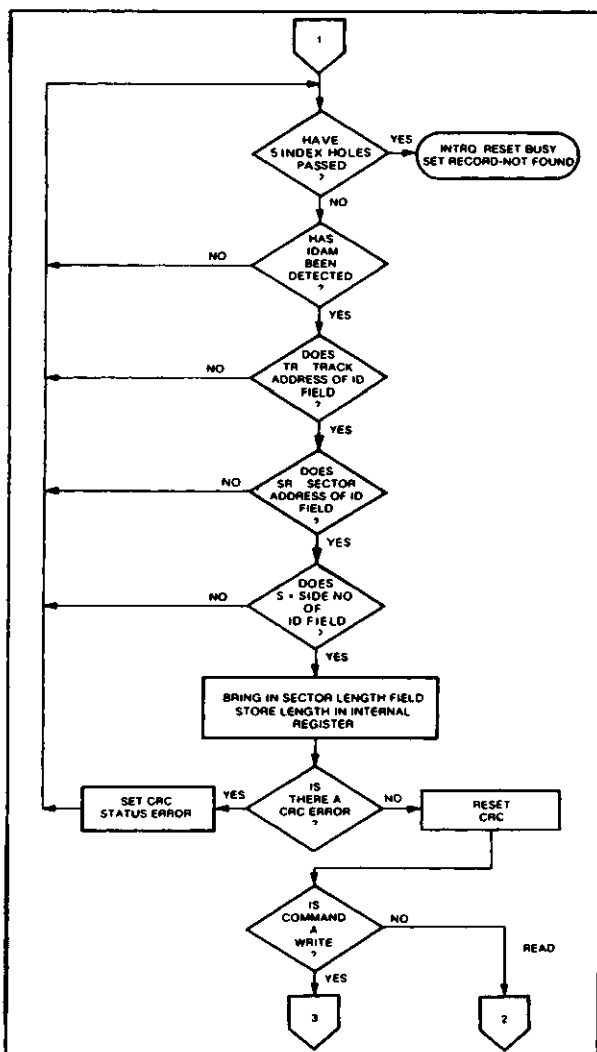
TYPE II COMMAND

Each of the Type II Commands contains an (m) flag which determines if multiple records (sectors) are to be read or written, depending upon the command. If m = 0, a single sector is read or written and an interrupt is generated at the completion of the command. If m = 1, multiple records are read or written with the sector register internally updated so that an address verification can occur on the next

record. The FD179X will continue to read or write multiple records and update the sector register in numerical ascending sequence until the sector register exceeds the number of sectors on the track or until the Force Interrupt command is loaded into the Command Register, which terminates the command and generates an interrupt.

For example: If the FD179X is instructed to read sector 27 and there are only 26 on the track, the sector register exceeds the number available. The FD179X will search for 5 disk revolutions, interrupt out, reset busy, and set the record not found status bit.

The Type II commands for 1791-94 also contain side select compare flags. When C = 0 (Bit 1) no side comparison is made. When C = 1, the LSB of the side number is read off the ID Field of the disk and compared with the contents of the (S) flag (Bit 3). If the S flag compares with the side number recorded in the ID field, the FD179X continues with the ID search. If a comparison is not made within 5 index pulses, the interrupt line is made active and the Record-Not-Found status bit is set.



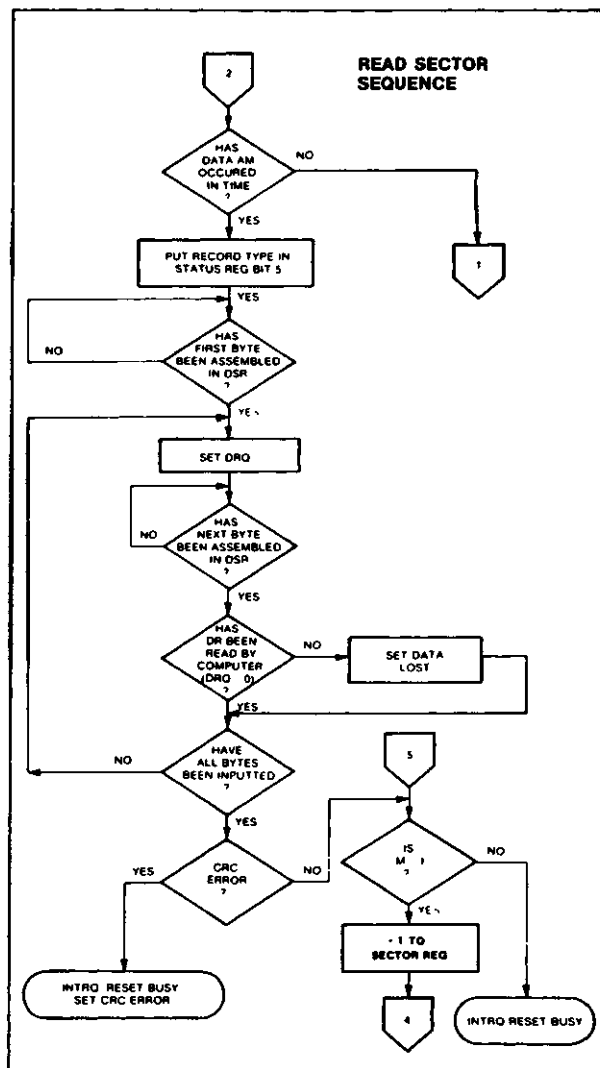
TYPE II COMMAND

The Type II and III commands for the 1795-97 contain a side select flag (Bit 1). When U = 0, SSO is updated to 0. Similarly, U = 1 updates SSO to 1. The chip compares the SSO to the ID field. If they do not compare within 5 revolutions the interrupt line is made active and the RNF status bit is set.

The 1795/7 READ SECTOR and WRITE SECTOR commands include a 'L' flag. The 'L' flag, in conjunction with the sector length byte of the ID Field, allows different byte lengths to be implemented in each sector. For IBM compatibility, the 'L' flag should be set to a one.

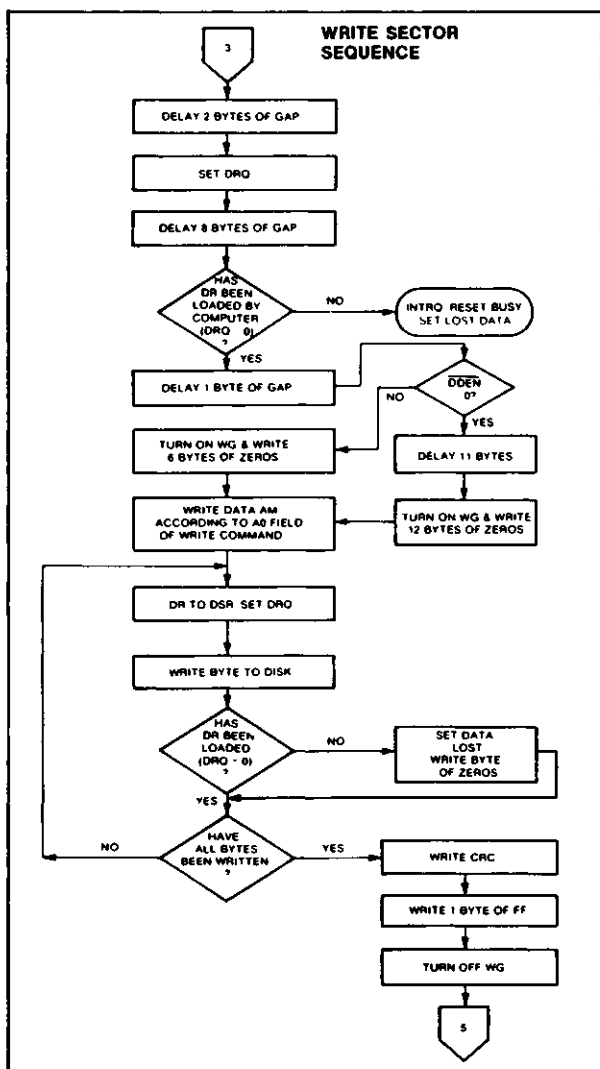
READ SECTOR

Upon receipt of the Read Sector command, the head is loaded, the Busy status bit set, and when an ID field is encountered that has the correct track number, correct sector number, correct side number, and correct CRC, the data field is presented to the computer. The Data Address



TYPE II COMMAND

WRITE SECTOR SEQUENCE



TYPE II COMMAND

Mark of the data field must be found within 30 bytes in single density and 43 bytes in double density of the last ID field CRC byte; if not, the ID field is searched for and verified again followed by the Data Address Mark search. If after 5 revolutions the DAM cannot be found, the Record Not Found status bit is set and the operation is terminated.

When the first character or byte of the data field has been shifted through the DSR, it is transferred to the DR, and DRQ is generated. When the next byte is accumulated in the DSR, it is transferred to the DR and another DRQ is generated. If the Computer has not read the previous contents of the DR before a new character is transferred that character is lost and the Lost Data Status bit is set. This sequence continues until the complete data field has been inputted to the computer. If there is a CRC error at the end of the data field, the CRC error status bit is set, and the command is terminated (even if it is a multiple record command).

At the end of the Read operation, the type of Data Address Mark encountered in the data field is recorded in the Status Register (Bit 5) as shown:

STATUS BIT 5

1	Deleted Data Mark
0	Data Mark

WRITE SECTOR

Upon receipt of the Write Sector command, the head is loaded (HLD active) and the Busy status bit is set. When an ID field is encountered that has the correct track number, correct sector number, correct side number, and correct CRC, a DRQ is generated. The FD179X counts off 11 bytes in single density and 22 bytes in double density from the CRC field and the Write Gate (WG) output is made active if the DRQ is serviced (i.e., the DR has been loaded by the computer). If DRQ has not been serviced, the command is terminated and the Lost Data status bit is set. If the DRQ has been serviced, the WG is made active and six bytes of zeroes in single density and 12 bytes in double density are then written on the disk. At this time the Data Address Mark is then written on the disk as determined by the a0 field of the command as shown below:

a0	Data Address Mark (Bit 0)
1	Deleted Data Mark
0	Data Mark

The FD179X then writes the data field and generates DRQ's to the computer. If the DRQ is not serviced in time for continuous writing the Lost Data Status Bit is set and a byte of zeroes is written on the disk. The command is not terminated. After the last data byte has been written on the disk, the two-byte CRC is computed internally and written on the disk followed by one byte of logic ones in FM or in MFM. The WG output is then deactivated. For a 2 MHz clock the INTRQ will set 8 to 12 μ sec after the last CRC byte is written. For partial sector writing, the proper method is to write the data and fill the balance with zeroes. By letting the chip fill the zeroes, errors may be masked by the lost data status and improper CRC Bytes.

TYPE III COMMANDS

READ ADDRESS

Upon receipt of the Read Address command, the head is loaded and the Busy Status Bit is set. The next encountered ID field is then read in from the disk, and the six data bytes of the ID field are assembled and transferred to the DR, and a DRQ is generated for each byte. The six bytes of the ID field are shown below:

TRACK ADDR	SIDE NUMBER	SECTOR ADDRESS	SECTOR LENGTH	CRC 1	CRC 2
1	2	3	4	5	6

Although the CRC characters are transferred to the computer, the FD179X checks for validity and the CRC error status bit is set if there is a CRC error. The Track Address of the ID field is written into the sector register so that a comparison can be made by the user. At the end of the operation an interrupt is generated and the Busy Status is reset.

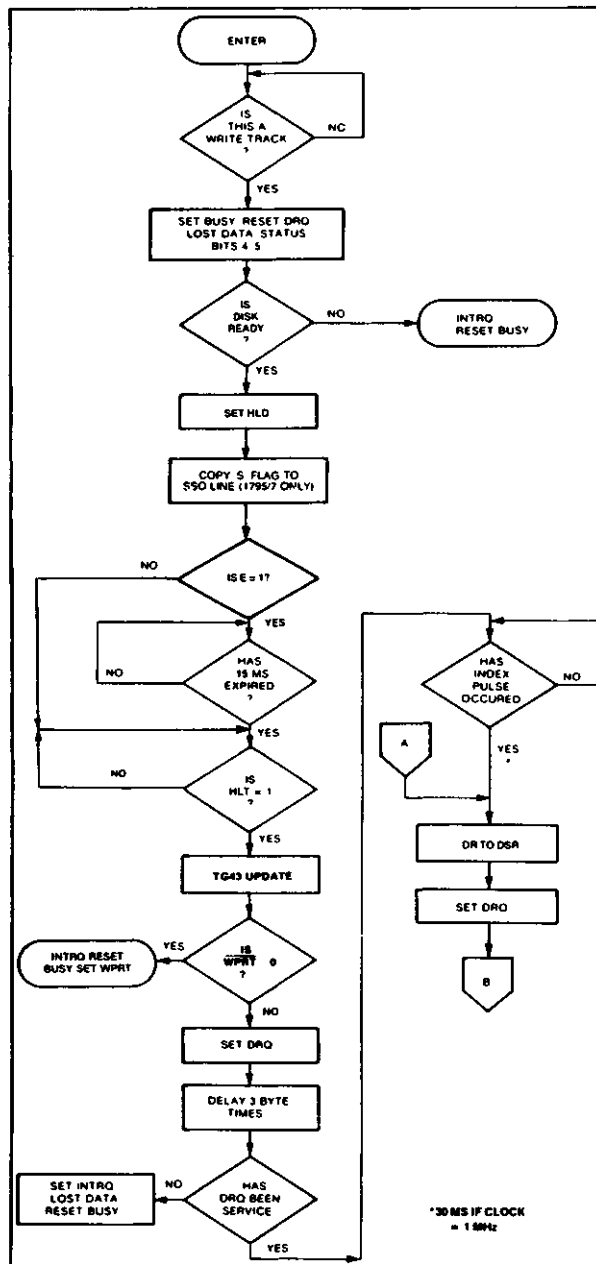
READ TRACK

Upon receipt of the READ track command, the head is loaded, and the Busy Status bit is set. Reading starts with the leading edge of the first encountered index pulse and continues until the next index pulse. All Gap, Header, and data bytes are assembled and transferred to the data register and DRQ's are generated for each byte. The accumulation of bytes is synchronized to each address mark encountered. An interrupt is generated at the completion of the command.

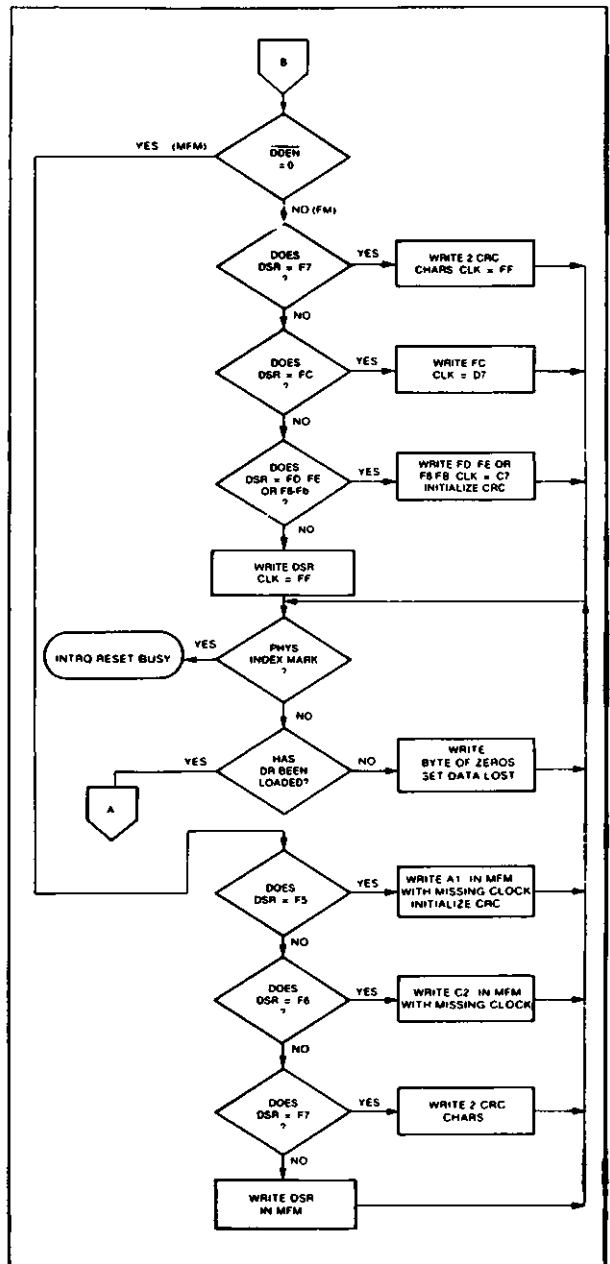
This command has several characteristics which make it suitable for diagnostic purposes. They are: the Read Gate

is not activated during the command; no CRC checking is performed; gap information is included in the data stream; the internal side compare is not performed; and the address mark detector is on for the duration of the command. Because the A.M. detector is always on, write splices or noise may cause the chip to look for an A.M. If an address mark does not appear on schedule the Lost Data status flag is set.

The ID A.M., ID field, ID CRC bytes, DAM, Data, and Data CRC Bytes for each sector will be correct. The Gap Bytes may be read incorrectly during write-splice time because of synchronization.



TYPE III COMMAND WRITE TRACK



TYPE III COMMAND WRITE TRACK

CONTROL BYTES FOR INITIALIZATION

DATA PATTERN IN DR (HEX)	FD179X INTERPRETATION IN FM (DDEN = 1)	FD1791/3 INTERPRETATION IN MFM (DDEN = 0)
00 thru F4	Write 00 thru F4 with CLK = FF	Write 00 thru F4, in MFM
F5	Not Allowed	Write A1* in MFM, Preset CRC
F6	Not Allowed	Write C2** in MFM
F7	Generate 2 CRC bytes	Generate 2 CRC bytes
F8 thru FB	Write F8 thru FB, Clk = C7, Preset CRC	Write F8 thru FB, in MFM
FC	Write FC with Clk = D7	Write FC in MFM
FD	Write FD with Clk = FF	Write FD in MFM
FE	Write FE, Clk = C7, Preset CRC	Write FE in MFM
FF	Write FF with Clk = FF	Write FF in MFM

*Missing clock transition between bits 4 and 5

**Missing clock transition between bits 3 & 4

WRITE TRACK FORMATTING THE DISK

(Refer to section on Type III commands for flow diagrams.)

Formatting the disk is a relatively simple task when operating programmed I/O or when operating under DMA with a large amount of memory. Data and gap information must be provided at the computer interface. Formatting the disk is accomplished by positioning the R/W head over the desired track number and issuing the Write Track command.

Upon receipt of the Write Track command, the head is loaded and the Busy Status bit is set. Writing starts with the leading edge of the first encountered index pulse and continues until the next index pulse, at which time the interrupt is activated. The Data Request is activated immediately upon receiving the command, but writing will not start until after the first byte has been loaded into the Data Register. If the DR has not been loaded by the time the index pulse is encountered the operation is terminated making the device Not Busy, the Lost Data Status Bit is set, and the Interrupt is activated. If a byte is not present in the DR when needed, a byte of zeroes is substituted.

This sequence continues from one index mark to the next index mark. Normally, whatever data pattern appears in the data register is written on the disk with a normal clock pattern. However, if the FD179X detects a data pattern of F5 thru FE in the data register, this is interpreted as data address marks with missing clocks or CRC generation.

The CRC generator is initialized when any data byte from F8 to FE is about to be transferred from the DR to the DSR in FM or by receipt of F5 in MFM. An F7 pattern will generate two CRC characters in FM or MFM. As a consequence, the patterns F5 thru FE must not appear in the gaps, data fields, or ID fields. Also, CRC's must be generated by an F7 pattern.

Disks may be formatted in IBM 3740 or System 34 formats with sector lengths of 128, 256, 512, or 1024 bytes.

TYPE IV COMMANDS

The Forced Interrupt command is generally used to terminate a multiple sector read or write command or to in-

sure Type I status in the status register. This command can be loaded into the command register at any time. If there is a current command under execution (busy status bit set) the command will be terminated and the busy status bit reset.

The lower four bits of the command determine the conditional interrupt as follows:

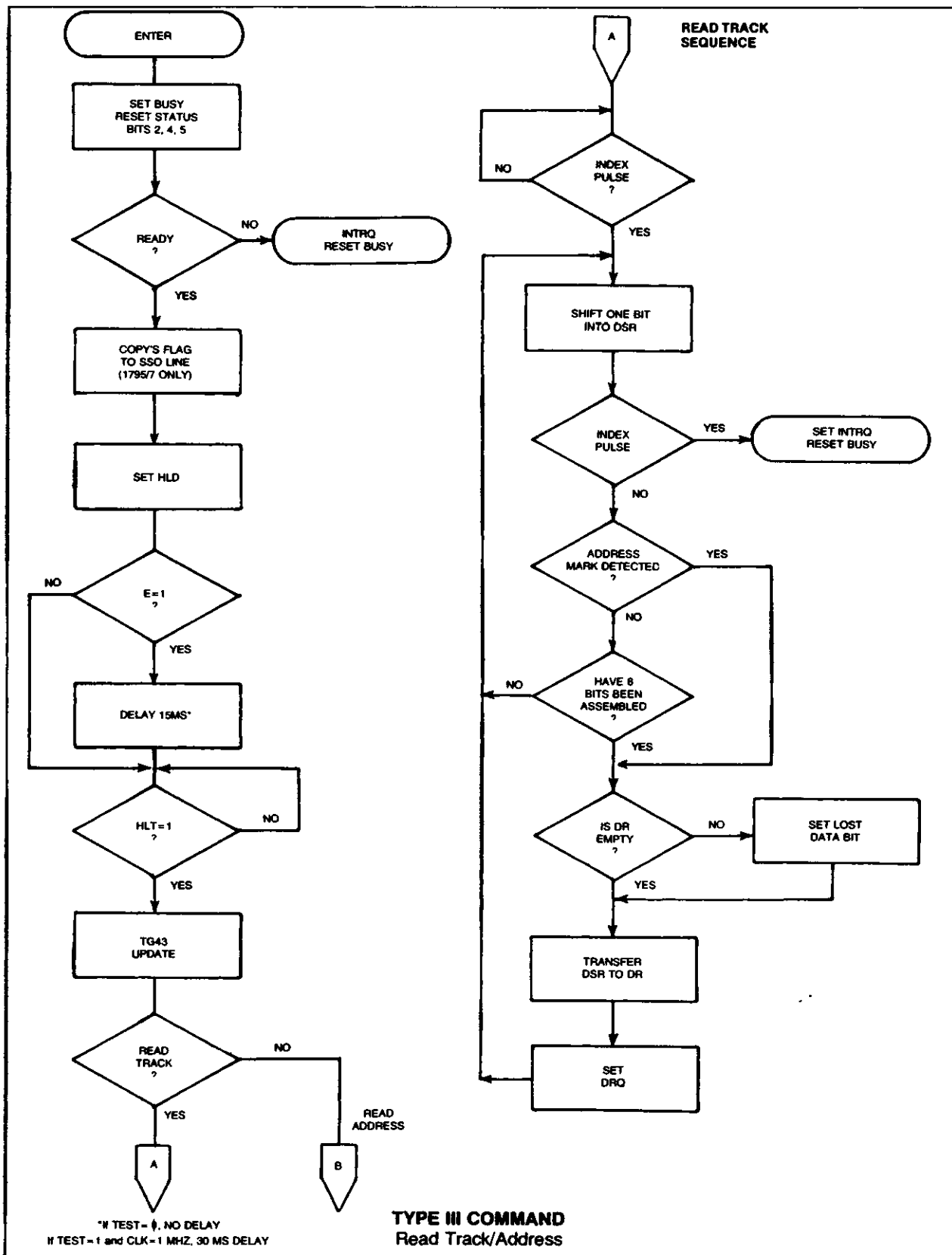
- I₀ = Not-Ready to Ready Transition
- I₁ = Ready to Not-Ready Transition
- I₂ = Every Index Pulse
- I₃ = Immediate Interrupt

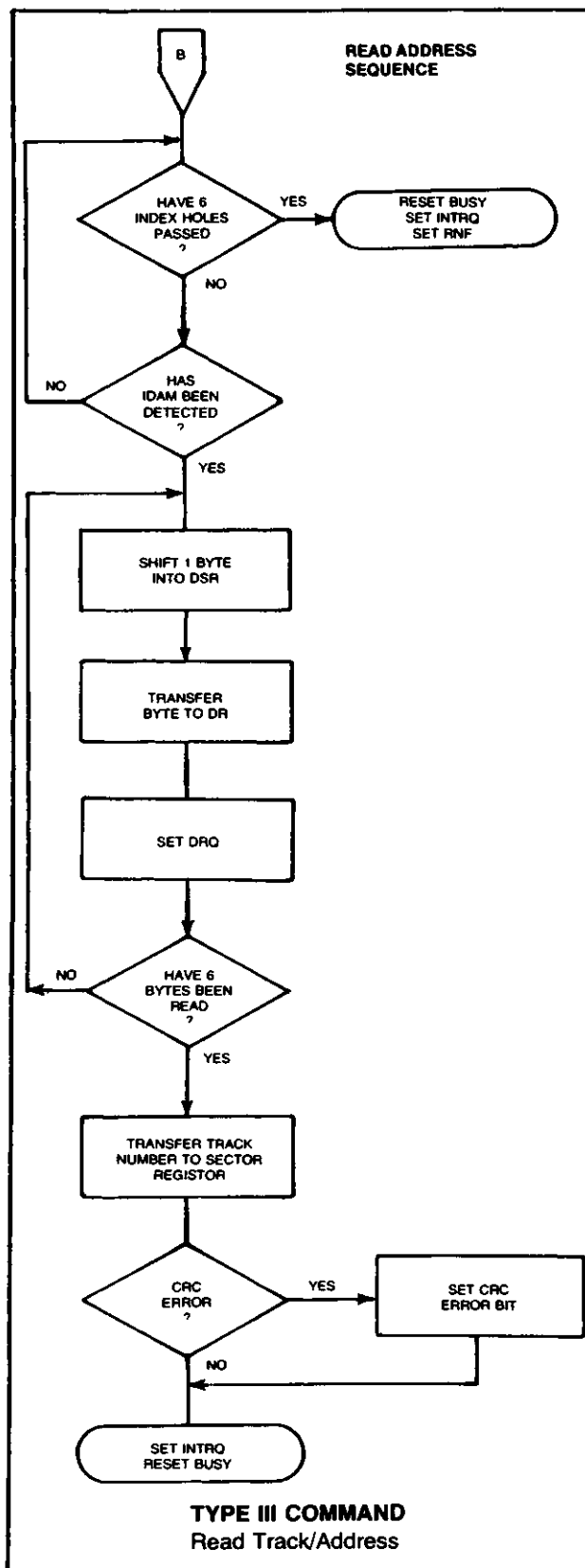
The conditional interrupt is enabled when the corresponding bit positions of the command (I₃ - I₀) are set to a 1. Then, when the condition for interrupt is met, the INTRQ line will go high signifying that the condition specified has occurred. If I₃ - I₀ are all set to zero (HEX D0), no interrupt will occur but any command presently under execution will be immediately terminated. When using the immediate interrupt condition (I₃ = 1) an interrupt will be immediately generated and the current command terminated. Reading the status or writing to the command register will not automatically clear the interrupt. The HEX D0 is the only command that will enable the immediate interrupt (HEX D8) to clear on a subsequent load command register or read status register operation. Follow a HEX D8 with D0 command.

Wait 8 micro sec (double density) or 16 micro sec (single density) before issuing a new command after issuing a forced interrupt (times double when clock = 1 MHz). Loading a new command sooner than this will nullify the forced interrupt.

Forced interrupt stops any command at the end of an internal micro-instruction and generates INTRQ when the specified condition is met. Forced interrupt will wait until ALU operations in progress are complete (CRC calculations, compares, etc.).

More than one condition may be set at a time. If for example, the READY TO NOT-READY condition (I₁ = 1) and the Every Index Pulse (I₂ = 1) are both set, the resultant command would be HEX "DA". The "OR" function is performed so that either a READY TO NOT-READY or the next Index Pulse will cause an interrupt condition.





STATUS REGISTER

Upon receipt of any command, except the Force Interrupt command, the Busy Status bit is set and the rest of the status bits are updated or cleared for the new command. If the Force Interrupt Command is received when there is a current command under execution, the Busy status bit is reset, and the rest of the status bits are unchanged. If the Force Interrupt command is received when there is not a current command under execution, the Busy Status bit is reset and the rest of the status bits are updated or cleared. In this case, Status reflects the Type I commands.

The user has the option of reading the status register through program control or using the DRQ line with DMA or interrupt methods. When the Data register is read the DRQ bit in the status register and the DRQ line are automatically reset. A write to the Data register also causes both DRQ's to reset.

The busy bit in the status may be monitored with a user program to determine when a command is complete, in lieu of using the INTRQ line. When using the INTRQ, a busy status check is not recommended because a read of the status register to determine the condition of busy will reset the INTRQ line.

The format of the Status Register is shown below:

(BITS)							
7	6	5	4	3	2	1	0
S7	S6	S5	S4	S3	S2	S1	S0

Status varies according to the type of command executed as shown in Table 4.

Because of internal sync cycles, certain time delays must be observed when operating under programmed I/O. They are: (times double when clock = 1 MHz)

Operation	Next Operation	Delay Req'd.	
		FM	MFM
Write to Command Reg.	Read Busy Bit (Status Bit 0)	12 μ s	6 μ s
Write to Command Reg.	Read Status Bits 1-7	28 μ s	14 μ s
Write Any Register	Read From Diff. Register	0	0

IBM 3740 FORMAT — 128 BYTES/SECTOR

Shown below is the IBM single-density format with 128 bytes/sector. In order to format a diskette, the user must issue the Write Track command, and load the data register with the following values. For every byte to be written, there is one Data Request.

Shown below is the IBM single-density format with 128 bytes/sector. In order to format a diskette, the user must issue the Write Track command, and load the data register with the following values. For every byte to be written, there is one Data Request.

- *Write bracketed field 26 times
- **Continue writing until FD179X interrupts out.
Approx. 247 bytes.
- 1-Optional '00' on 1795/7 only.

Shown below is the IBM dual-density format with 256 bytes/sector. In order to format a diskette the user must issue the Write Track command and load the data register with the following values. For every byte to be written, there is one data request.

- *Write bracketed field 26 times
- *Continue writing until FD179X interrupts out.
Approx. 598 bytes.



1. NON-IBM FORMATS

Variations in the IBM formats are possible to a limited extent if the following requirements are met:

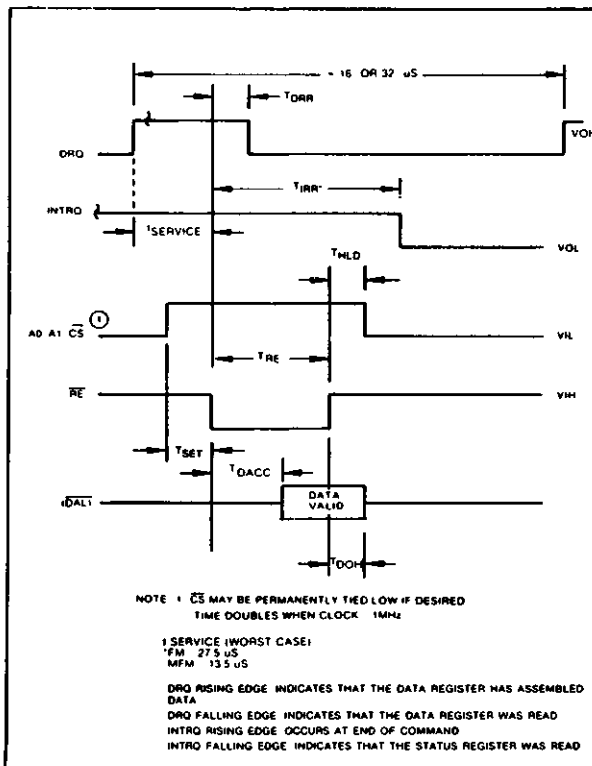
- 1) Sector size must be 128, 256, 512 or 1024 bytes.
- 2) Gap 2 cannot be varied from the IBM format.
- 3) 3 bytes of A1 must be used in MFM.

In addition, the Index Address Mark is not required for operation by the FD179X. Gap 1, 3, and 4 lengths can be as short as 2 bytes for FD179X operation, however PLL lock up time, motor speed variation, write-splice area, etc. will add more bytes to each gap to achieve proper operation. It is recommended that the IBM format be used for highest system reliability.

	FM	MFM
Gap I	16 bytes FF	32 bytes 4E
Gap II	11 bytes FF	22 bytes 4E
.	6 bytes 00	12 bytes 00
.		3 bytes A1
Gap III**	10 bytes FF 4 bytes 00	24 bytes 4E 8 bytes 00 3 bytes A1
Gap IV	16 bytes FF	16 bytes 4E

*Byte counts must be exact.

**Byte counts are minimum, except exactly 3 bytes of A1 must be written.



READ ENABLE TIMING

TIMING CHARACTERISTICS

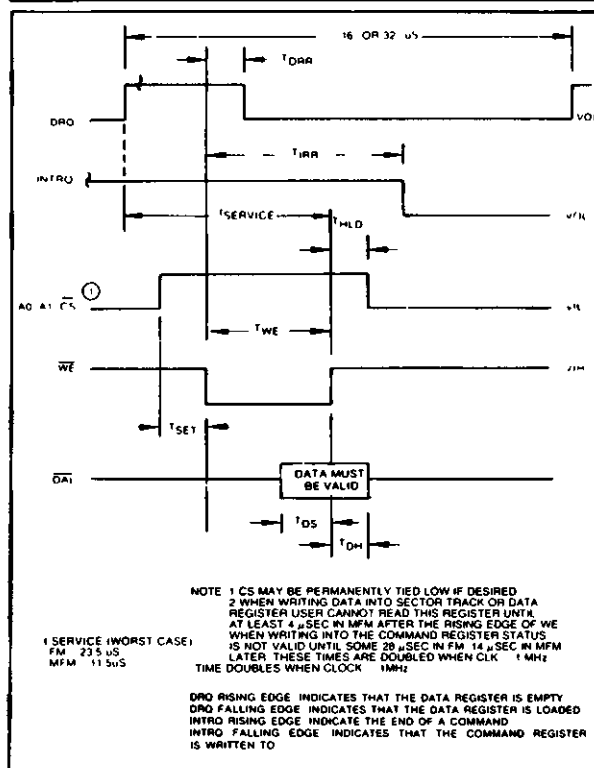
T_A = 0°C to 70°C, V_{DD} = +12V ± .6V, V_{SS} = 0V, V_{CC} = +5V ± .25V

READ ENABLE TIMING (See Note 6, Page 21)

SYMBOL	CHARACTERISTIC	MIN.	TYP.	MAX.	UNITS	CONDITIONS
TSET	Setup ADDR & CS to \overline{RE}	50			nsec	C _L = 50 pf
THLD	Hold ADDR & CS from \overline{RE}	10			nsec	
TRE	\overline{RE} Pulse Width	400			nsec	
TDRR	DRQ Reset from \overline{RE}		400	500	nsec	See Note 5 C _L = 50 pf C _L = 50 pf
TIRR	INTRQ Reset from \overline{RE}		500	3000	nsec	
TDACC	Data Access from \overline{RE}			350	nsec	
TDOH	Data Hold From \overline{RE}	50		150	nsec	

WRITE ENABLE TIMING (See Note 6, Page 21)

SYMBOL	CHARACTERISTIC	MIN.	TYP.	MAX.	UNITS	CONDITIONS
TSET	Setup ADDR & CS to \overline{WE}	50			nsec	See Note 5
THLD	Hold ADDR & CS from \overline{WE}	10			nsec	
TWE	\overline{WE} Pulse Width	350			nsec	
TDRR	DRQ Reset from \overline{WE}		400	500	nsec	
TIRR	INTRQ Reset from \overline{WE}		500	3000	nsec	
TDS	Data Setup to \overline{WE}	250			nsec	
TDH	Data Hold from \overline{WE}	70			nsec	



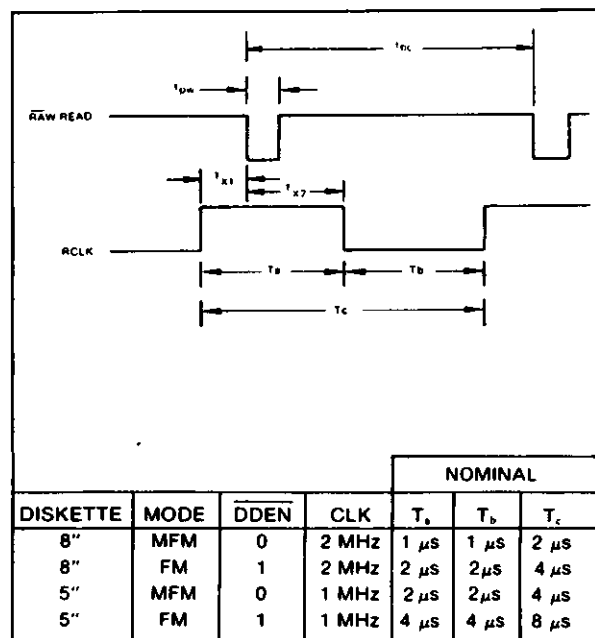
WRITE ENABLE TIMING

INPUT DATA TIMING:

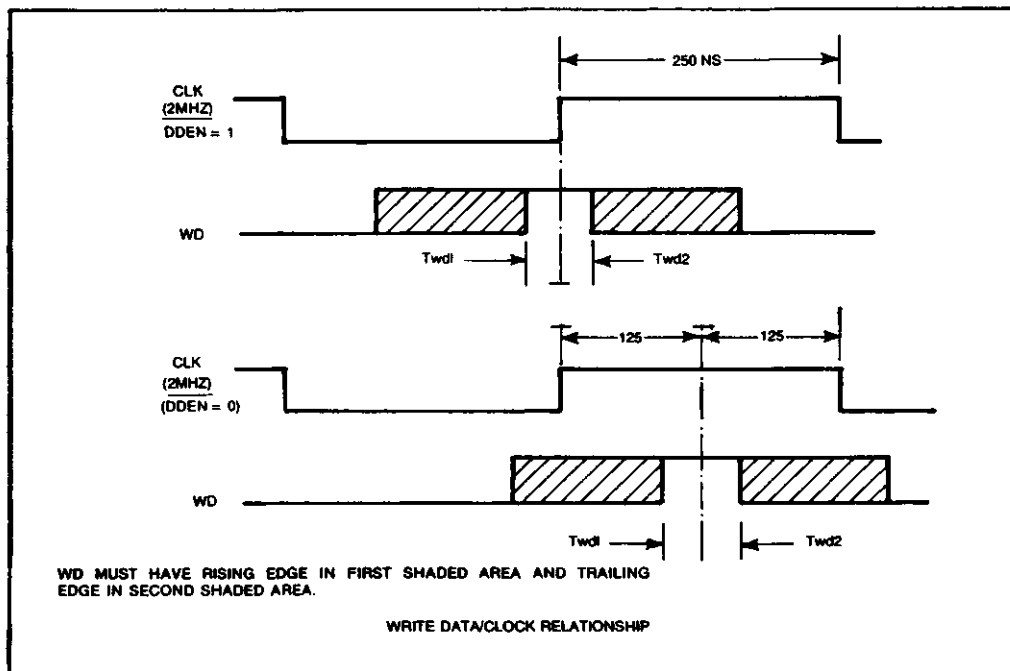
SYMBOL	CHARACTERISTIC	MIN.	TYP.	MAX.	UNITS	CONDITIONS
Tpw	Raw Read Pulse Width	100	200		nsec	See Note 1
tbc	Raw Read Cycle Time	1500	2000		nsec	1800 ns @ 70°C
Tc	RCLK Cycle Time	1500	2000		nsec	1800 ns @ 70°C
Tx ₁	RCLK hold to Raw Read	40			nsec	See Note 1
Tx ₂	Raw Read hold to RCLK	40			nsec	See Note 1

WRITE DATA TIMING: (ALL TIMES DOUBLE WHEN CLK = 1 MHz) (See Note 6, Page 21)

SYMBOL	CHARACTERISTICS	MIN.	TYP.	MAX.	UNITS	CONDITIONS
Twp	Write Data Pulse Width		500	650	nsec	FM
Twg	Write Gate to Write Data		200	350	nsec	MFM
Tbc	Write data cycle Time		2		μ sec	FM
Ts	Early (Late) to Write Data		1		μ sec	MFM
Th	Early (Late) From Write Data	125	2,3, or 4		μ sec	\pm CLK Error
Twf	Write Gate off from WD		2		nsec	MFM
Twd1	WD Valid to Clk		1		μ sec	FM
Twd2	WD Valid after CLK	100			nsec	MFM
		50			nsec	CLK=1 MHz
		100			nsec	CLK=2 MHz
		30			nsec	CLK=1 MHz
					nsec	CLK=2 MHz



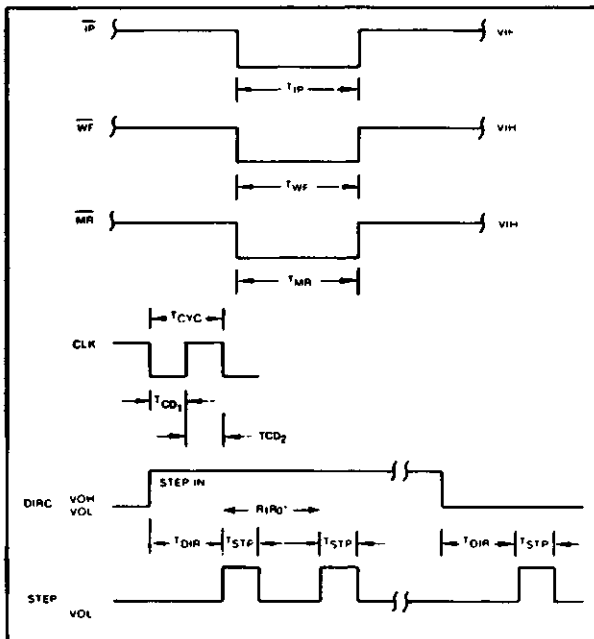
INPUT DATA TIMING



WRITE DATA TIMING

MISCELLANEOUS TIMING: (Times Double When Clock = 1 MHz) (See Note 6, Page 21)

SYMBOL	CHARACTERISTIC	MIN.	TYP.	MAX.	UNITS	CONDITIONS
TCD ₁	Clock Duty (low)	230	250	20000	nsec	See Note 5 ± CLK ERROR
TCD ₂	Clock Duty (high)	200	250	20000	nsec	
TSTP	Step Pulse Output	2 or 4			μsec	
TDIR	Dir Setup to Step		12		μsec	See Note 5
TMR	Master Reset Pulse Width	50			μsec	
TIP	Index Pulse Width	10			μsec	
TWF	Write Fault Pulse Width	10			μsec	



MISCELLANEOUS TIMING

*FROM STEP RATE TABLE

Table 4. STATUS REGISTER SUMMARY

BITS	ALL TYPE I COMMANDS	READ ADDRESS	READ SECTOR	READ TRACK	WRITE SECTOR	WRITE TRACK
S7	NOT READY	NOT READY	NOT READY	NOT READY	NOT READY	NOT READY
S6	WRITE PROTECT	0	0	0	WRITE PROTECT	WRITE PROTECT
S5	HEAD LOADED	0	RECORD TYPE	0	WRITE FAULT	WRITE FAULT
S4	SEEK ERROR	RNF	RNF	0	RNF	0
S3	CRC ERROR	CRC ERROR	CRC ERROR	0	CRC ERROR	0
S2	TRACK 00	LOST DATA	LOST DATA	LOST DATA	LOST DATA	LOST DATA
S1	INDEX PULSE	DRQ	DRQ	DRQ	DRQ	DRQ
S0	BUSY	BUSY	BUSY	BUSY	BUSY	BUSY

STATUS FOR TYPE I COMMANDS

BITS	NAME	MEANING
S7	NOT READY	This bit when set indicates the drive is not ready. When reset it indicates that the drive is ready. This bit is an inverted copy of the Ready input and logically 'ored' with MR.
S6	PROTECTED	When set, indicates Write Protect is activated. This bit is an inverted copy of WRPT input.
S5	HEAD LOADED	When set, it indicates the head is loaded and engaged. This bit is a logical "and" of HLD and HLT signals.
S4	SEEK ERROR	When set, the desired track was not verified. This bit is reset to 0 when updated.
S3	CRC ERROR	CRC encountered in ID field.
S2	TRACK 00	When set, indicates Read/Write head is positioned to Track 0. This bit is an inverted copy of the TROO input.
S1	INDEX	When set, indicates index mark detected from drive. This bit is an inverted copy of the IP input.
S0	BUSY	When set command is in progress. When reset no command is in progress.

NOTES:

1. Pulse width on RAW READ (Pin 27) is normally 100-300 ns. However, pulse may be any width if pulse is entirely within window. If pulse occurs in both windows, then pulse width must be less than 300 ns for MFM at CLK = 2 MHz and 600 ns for FM at 2 MHz. Times double for 1 MHz.
2. A PPL Data Separator is recommended for 8" MFM.
3. tbc should be 2 μ s, nominal in MFM and 4 μ s nominal in FM. Times double when CLK = 1 MHz.
4. RCLK may be high or low during RAW READ (Polarity is unimportant).
5. Times double when clock = 1 MHz.
6. Output timing readings are at $V_{OL} = 0.8v$ and $V_{OH} = 2.0v$.

STATUS FOR TYPE II AND III COMMANDS

BIT NAME	MEANING
S7 NOT READY	This bit when set indicates the drive is not ready. When reset, it indicates that the drive is ready. This bit is an inverted copy of the Ready input and 'ored' with MR. The Type II and III Commands will not execute unless the drive is ready.
S6 WRITE PROTECT	On Read Record: Not Used. On Read Track: Not Used. On any Write: It indicates a Write Protect. This bit is reset when updated.
S5 RECORD TYPE/ WRITE FAULT	On Read Record: It indicates the record-type code from data field address mark. 1 = Deleted Data Mark. 0 = Data Mark. On any Write: It indicates a Write Fault. This bit is reset when updated.
S4 RECORD NOT FOUND (RNF)	When set, it indicates that the desired track, sector, or side were not found. This bit is reset when updated.
S3 CRC ERROR	If S4 is set, an error is found in one or more ID fields; otherwise it indicates error in data field. This bit is reset when updated.
S2 LOST DATA	When set, it indicates the computer did not respond to DRQ in one byte time. This bit is reset to zero when updated.
S1 DATA REQUEST	This bit is a copy of the DRQ output. When set, it indicates the DR is full on a Read Operation or the DR is empty on a Write operation. This bit is reset to zero when updated.
S0 BUSY	When set, command is under execution. When reset, no command is under execution.

ELECTRICAL CHARACTERISTICS

Absolute Maximum Ratings

V_{DD} with respect to V_{SS} (ground): + 15 to - 0.3V

Voltage to any input with respect to V_{SS} = + 15 to - 0.3V

I_{CC} = 60 MA (35 MA nominal)

I_{DD} = 15 MA (10 MA nominal)

C_{IN} & C_{OUT} = 15 pF max with all pins grounded except one under test.

Operating temperature = 0°C to 70°C

Storage temperature = - 55°C to + 125°C

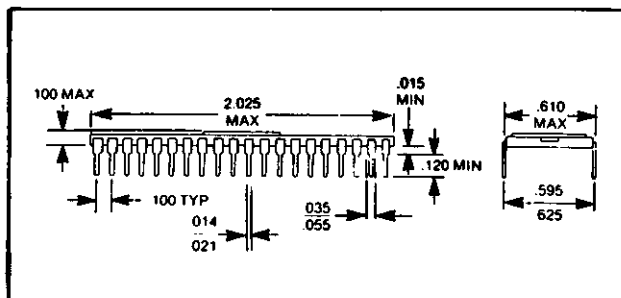
OPERATING CHARACTERISTICS (DC)

T_A = 0°C to 70°C, V_{DD} = + 12V \pm .6V, V_{SS} = 0V, V_{CC} = + 5V \pm .25V

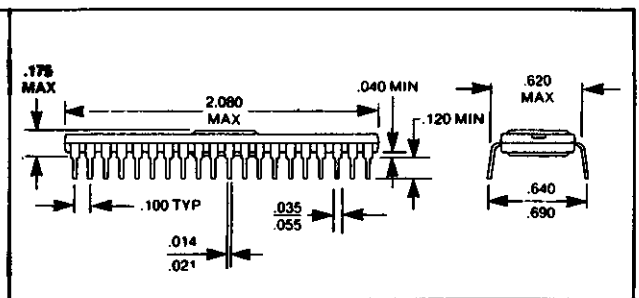
SYMBOL	CHARACTERISTIC	MIN.	MAX.	UNITS	CONDITIONS
I_L	Input Leakage		10	μA	$V_{IN} = V_{DD}^{**}$
I_{OL}	Output Leakage		10	μA	$V_{OUT} = V_{DD}$
V_{IH}	Input High Voltage	2.6		V	
V_{IL}	Input Low Voltage		0.8	V	
V_{OH}	Output High Voltage	2.8		V	$I_O = -100 \mu A$
V_{OL}	Output Low Voltage		0.45	V	$I_O = 1.6 mA^*$
P_D	Power Dissipation		0.6	W	

*1792 and 1794 $I_O = 1.0 mA$

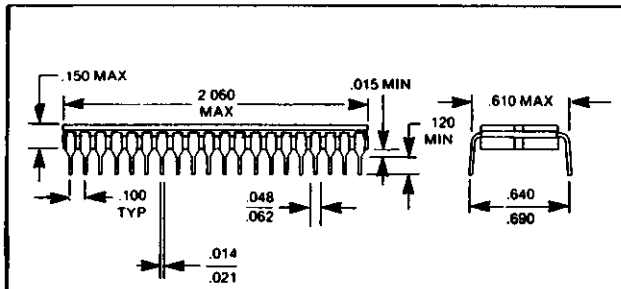
**Leakage conditions are for input pins without internal pull-up resistors. Pins 22, 23, 33, 36, and 37 have pull-up resistors. See Tech Memo #115 for testing procedures.



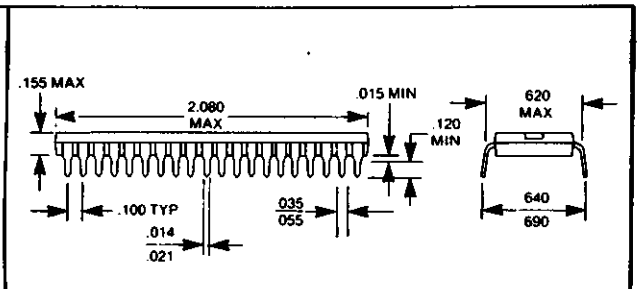
40 LEAD CERAMIC "A" or "AL"



40 LEAD RELPACK "B" or "BL"



40 LEAD CERDIP "CL"



40 LEAD PLASTIC "P" or "PL"



Information furnished by Western Digital Corporation is believed to be accurate and reliable. However, no responsibility is assumed by Western Digital Corporation for its use, nor for any infringements of patents or other rights of third parties which may result from its use. No license is granted by implication or otherwise under any patent or patent rights of Western Digital Corporation. Western Digital Corporation reserves the right to change specifications at anytime without notice.

WESTERN DIGITAL
C O R P O R A T I O N

2445 McCABE WAY
IRVINE, CALIFORNIA 92714

(714) 557-3550, TWX 910-595-1139



TECHNICAL MEMO

WESTERN DIGITAL
C O R P O R A T I O N

MEMO: 169

2445 McCabe Way
Irvine, California 92714
(714) 557-3550 TWX 910-595-1139

DEVICE: WD1770/1772/1773

TITLE: Preliminary Data Sheet Update

DATE: 8/29/83

The following information represents updates to the current WD1770/72/73 Preliminary Data sheet. These updates are performance enhancements.

1. TRE (Page 19) Changed from MIN 150NS to MIN 200NS.
2. TAH (Page 19) Changed from MIN 20NS to 10NS.
3. TWE (Page 19) Changed from MIN 150NS to MIN 200NS.
4. H bit in Command (Page 6 last paragraph)
Changed from: "If the hFlag is set and motor on
line (Pin 20)"

Changed to: "If the hFlag is NOT set and motor
on line (Pin 20)"

WESTERN DIGITAL

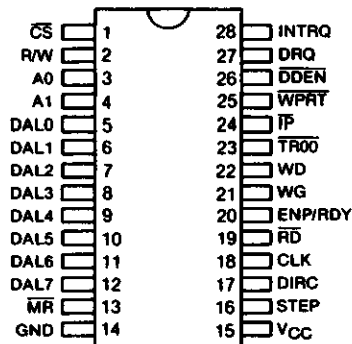
C O R P O R A T I O N

PRELIMINARY

WD1773 5¼" Floppy Disk Controller/Formatter

FEATURES

- 100% SOFTWARE COMPATIBILITY WITH WD1793
- BUILT-IN DATA SEPARATOR
- BUILT-IN WRITE PRECOMPENSATION
- SINGLE (FM) AND DOUBLE (MFM) DENSITY
- 28 PIN DIP, SINGLE +5V SUPPLY
- TTL COMPATIBLE INPUTS/OUTPUTS
- 128, 256, 512 OR 1024 SECTOR LENGTHS
- 8-BIT BI-DIRECTIONAL HOST INTERFACE



PIN DESIGNATION

DESCRIPTION

The WD1773 is an MOS/LSI device which performs the functions of a 5¼" Floppy Disk Controller/Formatter. It is fully software compatible with the Western Digital WD1793-02, allowing the designer to reduce parts count and board size on an existing WD1793 based design without software modifications.

With the exception of the enable Precomp/Ready line, the WD1773 is identical to the WD1770 controller. This line serves as both a READY input from the drive during READ/STEP operations, and as a Write Precompensation enable during Write operations. A built-in digital data separator virtually eliminates all external components associated with data recovery in previous designs.

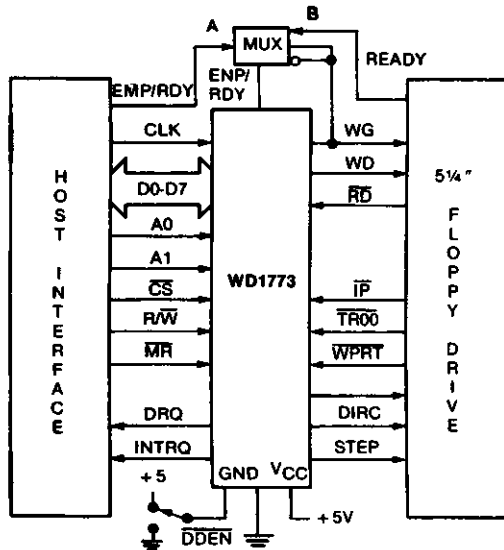
The WD1773 is implemented in NMOS silicon gate technology and is available in a 28 pin, dual-in-line package.

PIN DESCRIPTION

PIN NUMBER	PIN NAME	MNEMONIC	FUNCTION																									
1	CHIP SELECT	\overline{CS}	A logic low on this input selects the chip and enable Host communication with the device.																									
2	READ/WRITE	R/\overline{W}	A logic high on this input controls the placement of data on the $D0-D7$ lines from a selected register, while a logic low causes a write operation to a selected register.																									
3,4	ADDRESS 0,1	A0, A1	These two inputs select a register to Read/Write data: <table><tr><td>\overline{CS}</td><td>A1</td><td>A0</td><td>$R/\overline{W} = 1$</td><td>$R/\overline{W} = 0$</td></tr><tr><td>0</td><td>0</td><td>0</td><td>Status Reg</td><td>Command Reg</td></tr><tr><td>0</td><td>0</td><td>1</td><td>Track Reg</td><td>Track Reg</td></tr><tr><td>0</td><td>1</td><td>0</td><td>Sector Reg</td><td>Sector Reg</td></tr><tr><td>0</td><td>1</td><td>1</td><td>Data Reg</td><td>Data Reg</td></tr></table>	\overline{CS}	A1	A0	$R/\overline{W} = 1$	$R/\overline{W} = 0$	0	0	0	Status Reg	Command Reg	0	0	1	Track Reg	Track Reg	0	1	0	Sector Reg	Sector Reg	0	1	1	Data Reg	Data Reg
\overline{CS}	A1	A0	$R/\overline{W} = 1$	$R/\overline{W} = 0$																								
0	0	0	Status Reg	Command Reg																								
0	0	1	Track Reg	Track Reg																								
0	1	0	Sector Reg	Sector Reg																								
0	1	1	Data Reg	Data Reg																								
5-12	DATA ACCESS LINES 0 THROUGH 7	DAL0-DAL7	Eight bit bidirectional bus used for transfer of data, control, or status. This bus is enabled by \overline{CS} and R/\overline{W} . Each line will drive one TTL load.																									
13	MASTER RESET	\overline{MR}	A logic low pulse on this line resets the device and initializes the status register. Internal pull-up.																									
14	GROUND	GND	Ground.																									
15	POWER SUPPLY	VCC	+5V \pm 5% power supply input.																									
16	STEP	STEP	The Step output contains a pulse for each step of the drive's R/\overline{W} head.																									
17	DIRECTION	DIRC	The Direction output is high when stepping in towards the center of the diskette, and low when stepping out.																									
18	CLOCK	CLK	This input requires a free-running 40 to 60% duty cycle clock (for internal timing) at 8 MHZ \pm 1%.																									
19	READ DATA	\overline{RD}	This active low input is the raw data line containing both clock and data pulses from the drive.																									
20	ENABLE PRECOMP/READY LINE	ENP/RDY	Serves as a READY input from the drive during READ/STEP operations and as a Write Precomp enable during write operations.																									
21	WRITE GATE	WG	This output is made valid prior to writing on the diskette.																									
22	WRITE DATA	WD	FM or MFM clock and data pulses are placed on this line to be written on the diskette.																									
23	TRACK 00	$\overline{TR00}$	This active low input informs the WD1773 that the drive's R/\overline{W} heads are positioned over Track zero.																									
24	INDEX PULSE	\overline{IP}	This active low input informs the WD1773 when the physical index hole has been encountered on the diskette.																									
25	WRITE PROTECT	\overline{WPRT}	This input is sampled whenever a Write Command is received. A logic low on this line will prevent any Write Command from executing. Internal pull-up.																									
26	DOUBLE DENSITY ENABLE	\overline{DDEN}	This input pin selects either single (FM) or double (MFM) density. When $\overline{DDEN} = 0$, double density is selected. Internal pull-up.																									

PIN DESCRIPTION (CONTINUED)

PIN NUMBER	PIN NAME	MNEMONIC	FUNCTION
27	DATA REQUEST	DRQ	This active high output indicates that the Data Register is full (on a Read) or empty (on a Write operation).
28	INTERRUPT REQUEST	INTRQ	This active high output is set at the completion of any command or reset a read of the Status Register.



WD1773 SYSTEM BLOCK DIAGRAM

ARCHITECTURE

The Floppy Disk Formatter block diagram is illustrated on page 4. The primary sections include the parallel processor interface and the Floppy Disk interface.

Data Shift Register — This 8-bit register assembles serial data from the Read Data input (RD) during Read operations and transfers serial data to the Write Data output during Write operations.

Data Register — This 8-bit register is used as a holding register during Disk Read and Write operations. In Disk Read operations, the assembled data byte is transferred in parallel to the Data Register from the Data Shift Register. In Disk Write operations, information is transferred in parallel from the Data Register to the Data Shift Register.

When executing the Seek command, the Data Register holds the address of the desired Track position.

This register is loaded from the DAL and gated onto the DAL under processor control.

Track Register — This 8-bit register holds the track number of the current Read/Write head position. It is incremented by one every time the head is stepped in and decremented by one when the head is stepped out (towards track 00). The contents of the register are compared with the recorded track number in the ID field during disk Read, Write, and Verify operations. The Track Register can be loaded from or transferred to the DAL. This Register should not be loaded when the device is busy.

Sector Register (SR) — This 8-bit register holds the address of the desired sector position. The contents of the register are compared with the recorded sector number in the ID field during disk Read or Write operations. The Sector Register contents can be loaded from or transferred to the DAL. This register should not be loaded when the device is busy.

Command Register (CR) — This 8-bit register holds the command presently being executed. This register should not be loaded when the device is busy unless the new command is a force interrupt. The command register can be loaded from the DAL, but not read onto the DAL.

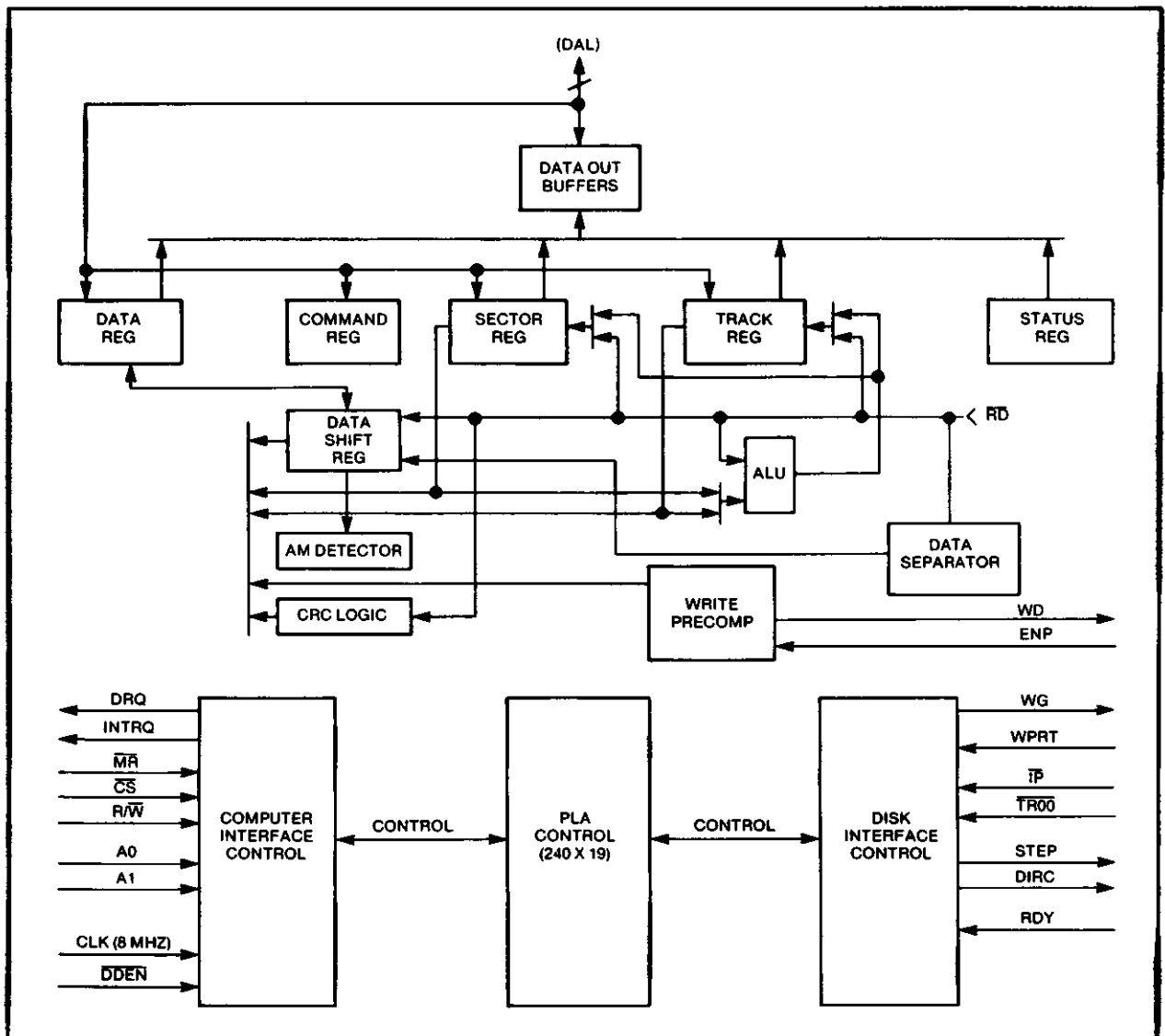
Status Register (STR) — This 8-bit register holds device Status information. The meaning of the Status bits is a function of the type of command previously executed. This register can be read onto the DAL, but not loaded from the DAL.

CRC Logic — This logic is used to check or to generate the 16-bit Cyclic Redundancy Check (CRC). The polynomial is:

$$G(x) = x^{16} + x^{12} + x^5 + 1.$$

The CRC includes all information starting with the address mark and up to the CRC characters. The CRC register is preset to ones prior to data being shifted through the circuit.

Arithmetic/Logic Unit (ALU) — The ALU is a serial comparator, incrementer, and decrements and is used for register modification and comparisons with the disk recorded ID field.



WD1773 BLOCK DIAGRAM

Timing and Control — All computer and Floppy Disk interface controls are generated through this logic. The internal device timing is generated from an external crystal clock. The WD1773 has two different modes of operation according to the state of DDEN. When DDEN = 0, double density (MFM) is enabled. When DDEN = 1, single density is enabled.

AM Detector — The address mark detector detects ID, data and index address marks during read and write operations.

Data Separator — A digital data separator consisting of a ring shift register and data window detection logic provides read data and a recovery clock to the AM detector.

PROCESSOR INTERFACE

The interface to the processor is accomplished through the eight Data Access Lines (DAL) and associated control signals. The DAL are used to transfer Data, Status, and Control words out of, or into the WD1773. The DAL are three state buffers that are enabled as output drivers when Chip Select (CS) and R/W = 1 are active or act as input receivers when CS and R/W = 0 are active.

When transfer of data with the Floppy Disk Controller is required by the host processor, the device address is decoded and CS is made low. The address bits A1 and A0, combined with the signal R/W during a Read operation or Write operation are interpreted as selecting the following registers:

A1 - A0	READ (R/W = 1)	WRITE (R/W = 0)
0 0	Status Register	Command Register
0 1	Track Register	Track Register
1 0	Sector Register	Sector Register
1 1	Data Register	Data Register

During Direct Memory Access (DMA) types of data transfers between the Data Register of the WD1773 and the processor, the Data Request (DRQ) output is used in Data Transfer control. This signal also appears as status bit 1 during Read and Write operations.

On Disk Read operations the Data Request is activated (set high) when an assembled serial input byte is transferred in parallel to the Data Register. This bit is cleared when the Data Register is read by the processor. If the Data Register is read after one or more characters are lost, by having new data transferred into the register prior to processor readout, the Lost Data bit is set in the Status Register. The Read operations continues until the end of sector is reached.

On Disk Write operations the Data Request is activated when the Data Register transfers its contents to the Data Shift Register, and requires a new data byte. It is reset when the Data Register is loaded with new data by the processor. If new data is not loaded at the time the next serial byte is required by the Floppy Disk, a byte of zeroes is written on the diskette and the Lost Data is set in the Status Register.

At the completion of every command an INTRQ is generated. INTRQ is reset by either reading the status register or by loading the command register with a new command. In addition, INTRQ is generated if a Force Interrupt command condition is met.

The WD1773 has two modes of operation according to the state DDEN (Pin 26). When DDEN = 1, single density is selected. In either case, the CLK input (Pin 18) is at 8 MHz.

GENERAL DISK READ OPERATIONS

Sector lengths of 128, 256, 512 or 1024 are obtainable in either FM or MFM formats. For FM, DDEN should be placed to logical "1." For MFM formats, DDEN should be placed to a logical "0." Sector lengths are determined at format time by the fourth byte in the "ID" field.

SECTOR LENGTH TABLE	
SECTOR LENGTH FIELD (HEX)	NUMBER OF BYTES IN SECTOR (DECIMAL)
00	128
01	256
02	512
03	1024

The number of sectors per track as far as the WD1773 is concerned can be from 1 to 255 sectors. The

number of tracks as far as the WD1773 is concerned is from 0 to 255 tracks.

GENERAL DISK WRITE OPERATION

When writing is to take place on the diskette the Write Gate (WG) output is activated, allowing current to flow into the Read/Write head. As a precaution to erroneous writing the first data byte must be loaded into the Data Register in response to a Data Request from the device before the Write Gate signal can be activated.

Writing is inhibited when the Write Protect input is a logic low, in which case any Write command is immediately terminated, an interrupt is generated and the Write Protect status bit is set.

For Write operations, the WD1773 provides Write Gate (Pin 21) to enable a Write condition, and Write Data (Pin 22) which consists of a series of active high pulses. These pulses contain both Clock and Data information in FM and MFM. Write Data provides the unique missing clock patterns for recording Address Marks.

If Precomp Enable (ENP) is active when WG is asserted, automatic Write Precompensation takes place. The outgoing Write Data stream is delayed or advanced from nominal by 125 nanoseconds according to the following table:

PATTERN				MFM	FM
X	1	1	0	Early	N/A
X	0	1	1	Late	N/A
0	0	0	1	Early	N/A
1	0	0	0	Late	N/A

Next Bit to be sent
Current Bit sending
Previous Bits sent

Precompensation is typically enabled on the innermost tracks where bit shifts usually occur and bit density is at its maximum.

COMMAND DESCRIPTION

The WD1773 will accept eleven commands. Command words should only be loaded in the Command Register when the Busy status bit is off (Status bit 0). The one exception is the Force Interrupt command. Whenever a command is being executed, the Busy status bit is set. When a command is completed, an interrupt is generated and the Busy status bit is reset. The Status Register indicates whether the completed command encountered an error or was fault free. For ease of discussion, commands are divided into four types. Commands and types are summarized in Table 1.

TABLE 1. COMMAND SUMMARY

TYPE	COMMAND	BITS							
		7	6	5	4	3	2	1	0
I	Restore	0	0	0	0	h	V	r1	r0
I	Seek	0	1	0	1	h	V	r1	r0
I	Step	0	0	1	T	h	V	r1	r0
I	Step-in	0	1	0	T	h	V	r1	r0
I	Step-out	0	1	1	T	h	V	r1	r0
II	Read Sector	1	0	0	m	L	E	U	a0
II	Write Sector	1	0	1	m	L	E	U	a0
III	Read Address	1	1	0	0	0	E	U	0
III	Read Track	1	1	1	0	0	E	U	0
III	Write Track	1	1	1	1	0	E	U	0
IV	Force Interrupt	1	1	0	1	l3	l2	l1	l0

FLAG SUMMARY

COMMAND TYPE	BIT NO(S)		DESCRIPTION																				
I	0, 1	r1 r0 = Stepping Motor Rate See Table 3 for Rate Summary																					
I	2	V = Track Number Verify Flag	V = 0, No verify V = 1, Verify on destination track																				
I	3	h = Don't Care																					
I	4	T = Track Update Flag	T = 0, No update T = 1, Update track register																				
II	0	a0 = Data Address Mark	a0 = 0, FB (DAM) a0 = 1, F8 (deleted DAM)																				
II	1	C = Side Compare Flag	C = 0, Disable side compare C = 1, Enable side compare																				
II & III	1	U = Update SSO	U = 0, Update SSO to 0 U = 1, Update SSO to 1																				
II & III	2	E = 15 MS Delay	E = 0, No 30 MS delay E = 1, 15 MS delay																				
II	3	S = Side Compare Flag	S = 0, Compare for side 0 S = 1, Compare for side 1																				
II	3	L = Sector Length Flag	<table><tr><td></td><td colspan="4">LSB's Sector Length in ID Field</td></tr><tr><td></td><td>00</td><td>01</td><td>10</td><td>11</td></tr><tr><td>L = 0</td><td>256</td><td>512</td><td>1024</td><td>128</td></tr><tr><td>L = 1</td><td>128</td><td>256</td><td>512</td><td>1024</td></tr></table>		LSB's Sector Length in ID Field					00	01	10	11	L = 0	256	512	1024	128	L = 1	128	256	512	1024
	LSB's Sector Length in ID Field																						
	00	01	10	11																			
L = 0	256	512	1024	128																			
L = 1	128	256	512	1024																			
II	4	m = Multiple Record Flag	m = 0, Single record m = 1, Multiple records																				
IV	0-3	lx = Interrupt Condition Flags l0 = 1 Not Ready To Ready Transition l1 = 1 Ready To Not Ready Transition l2 = 1 Index Pulse l3 = 1 Immediate Interrupt, Requires A Reset l3-l1 = 0 Terminate With No Interrupt (INTRQ)																					

*NOTE: See Type IV Command Description for further information.

TYPE I COMMANDS

The Type I Commands include the Restore, Seek, Step, Step-In, and Step-Out commands. Each of the Type I Commands contains a rate field ($r_0 r_1$), which determines the stepping motor rate as defined in Table 3.

A 4 μ s (MFM) or 8 μ s (FM) pulse is provided as an output to the drive. For every step pulse issued, the drive moves one track location in a direction determined by the direction output. The chip will step the drive in the same direction it last stepped unless the command changes the direction.

The Direction signal is active high when stepping in and low when stepping out. The Direction signal is valid 24 or 48 μ sec before the first stepping pulse is generated.

When a Seek, Step or Restore command is executed an optional verification of Read-Write head position can be performed by settling bit 2 ($V = 1$) in the command word to a logic 1. The verification operation begins at the end of the 30 msec settling time. The track number from the first encountered ID Field is compared against the contents of the Track Register. If the track numbers compare and the ID Field Cyclic Redundancy Check (CRC) is correct, the verify operation is complete and an INTRQ is generated with no errors. If there is a match but not a valid CRC, the CRC error status bit is set (Status bit 3), and the next encountered ID field is read from the disk for the verification operation.

The WD1773 must find an ID field with correct track number and correct CRC within 5 revolutions of the media; otherwise the seek error is set and an INTRQ is generated. If $V = 0$, no verification is performed.

RESTORE (SEEK TRACK 0)

Upon receipt of this command the Track 00 ($\overline{TR00}$) input is sampled. If $\overline{TR00}$ is active low indicating the Read-Write head is positioned over track 0, the Track Register is loaded with zeroes and an interrupt is generated. If $\overline{TR00}$ is not active low, stepping pulses at a rate specified by the $r_1 r_0$ field are issued until the $\overline{TR00}$ input is activated. At this time the Track Register is loaded with zeroes and an interrupt is generated. If the $\overline{TR00}$ input does not go active low after 255 stepping pulses, the WD1773 terminates operation, interrupts, and sets the Seek error status bit, providing the V flag is set. A verification operation also takes place if the V flag is set. Note that the Restore command is executed when \overline{MR} goes from an active to an inactive state and that the DRQ pin stays low.

SEEK

This command assumes that the Track Register contains the track number of the current position of the Read-Write head and the Data Register contains the desired track number. The WD1773 will update the Track register and issue stepping pulses in the appropriate direction until the contents of the Track

register are equal to the contents of the Data Register (the desired track location). A verification operation takes place if the V flag is on. An interrupt is generated at the completion of the command. Note: When using multiple drives, the track register must be updated for the drive selected before seeks are issued.

STEP

Upon receipt of this command, the WD1773 issues one stepping pulse to the disk drive. The stepping motor direction is the same as in the previous step command. After a delay determined by the $r_1 r_0$ field, a verification takes place if the V flag is on. If the U flag is on, the Track Register is updated. An interrupt is generated at the completion of the command.

STEP-IN

Upon receipt of this command, the WD1773 issues one stepping pulse in the direction towards track 76. If the U flag is on, the Track Register is incremented by one. After a delay determined by the $r_1 r_0$ field, a verification takes place if the V flag is on. An interrupt is generated at the completion of the command.

STEP-OUT

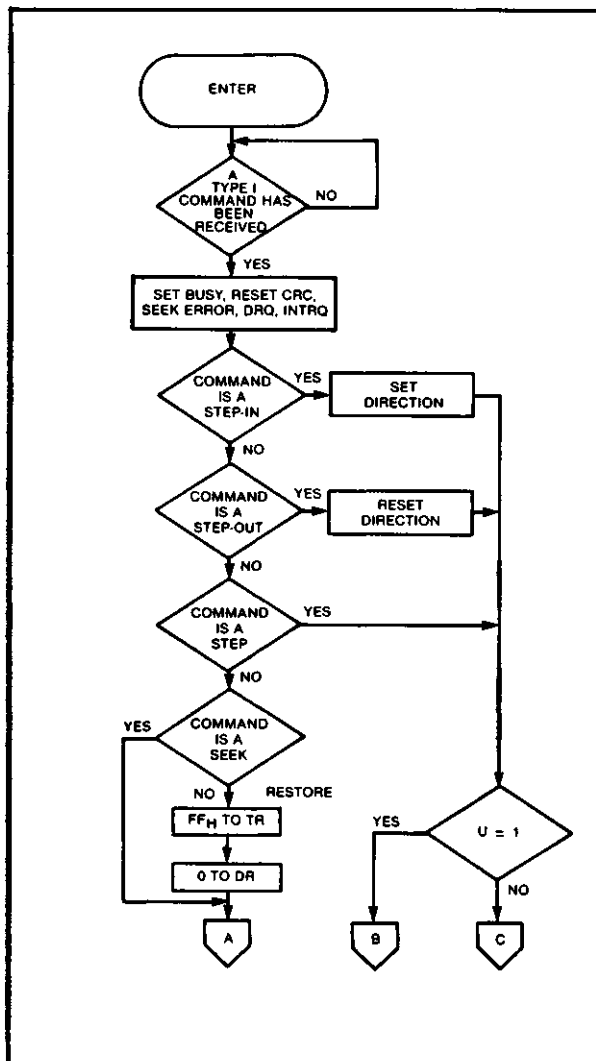
Upon receipt of this command, the WD1773 issues one stepping pulse in the direction towards track 0. If the U flag is on, the Track Register is decremented by one. After a delay determined by the $r_1 r_0$ field, a verification takes place if the V flag is on. An interrupt is generated at the completion of the command.

TYPE II COMMANDS

The Type II Commands are the Read Sector and Write Sector commands. Prior to loading the Type II Command into the Command Register, the computer must load the Sector Register with the desired sector number. Upon receipt of the Type II command, the busy status Bit is set. The E flag is still active providing a delay of 1 to 30 msec for head settling time.

When an ID field is located on the disk, the WD1773 compares the Track Number on the ID field with the Track Register. If there is not a match, the next encountered ID field is read and a comparison is again made. If there was a match, the Sector Number of the ID field is compared with the Sector Register. If there is not a Sector match, the next encountered ID field is read off the disk and comparisons again made. If the ID field CRC is correct, the data field is then located and will be either written into, or read from depending upon the command. The WD1773 must find an ID field with a Track number, Sector number, side number, and CRC within five revolutions of the disk; otherwise, the Record not found status bit is set (Status bit 3) and the command is terminated with an interrupt.

Each of the Type II Commands contains an (m) flag which determines if multiple records (sectors) are to be read or written, depending upon the command. If $m = 0$, a single sector is read or written and an inter-

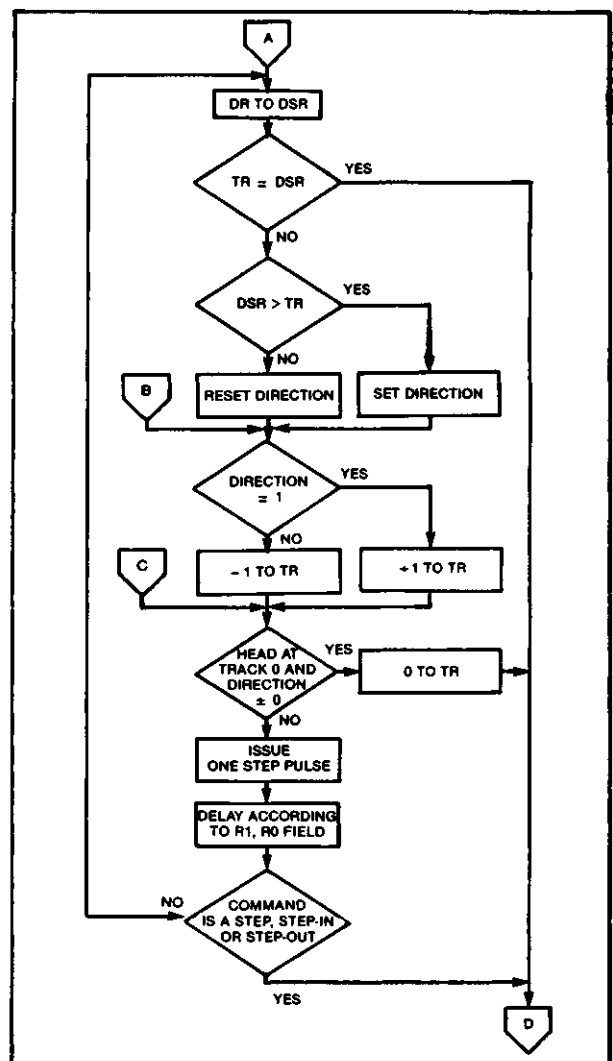


TYPE I COMMAND FLOW

rupt is generated at the completion of the command. If $m = 1$, multiple records are read or written with the sector register internally updated so that an address verification can occur on the next record. The WD1773 will continue to read or write multiple records and update the sector register in numerical ascending sequence until the sector register exceeds the number of sectors on the track or until the Force Interrupt command is loaded into the Command Register, which terminates the command and generates an interrupt.

For example: If the WD1773 is instructed to read sector 27 and there are only 26 on the track, the sector register exceeds the number available. The WD1773 will search for 5 disk revolutions, interrupt out, reset busy, and set the record not found status bit.

The Type II commands for WD1773 contain side compare flags. When $C = 0$ (Bit 1) no side comparison is made. When $C = 1$, the LSB of the side num-



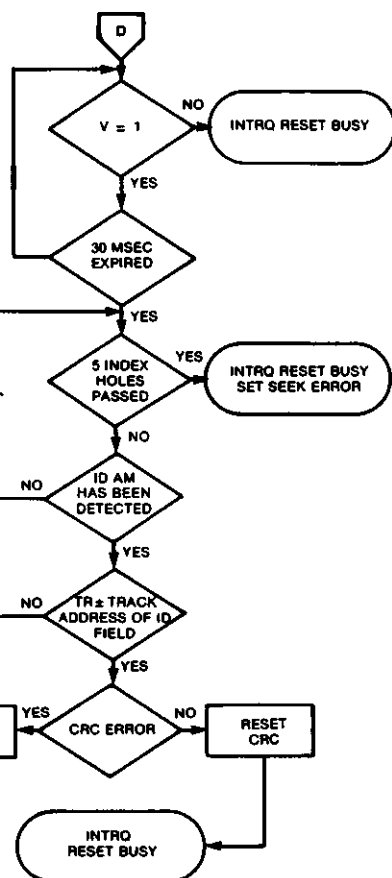
TYPE I COMMAND FLOW

ber is read off the ID Field of the disk and compared with the contents of the (S) flag (Bit 3). If the S flag compares with the side number recorded in the ID field, the WD1773 continues with the ID search. If a comparison is not made within 6 index pulses, the interrupt line is made active and the Record-Not-Found status bit is set.

READ SECTOR

Upon receipt of the Read Sector command, the Busy status bit is set, and when an ID field is encountered that has the correct track number, correct sector number, correct side number, and correct CRC, the data field is presented to the computer. The Data Address Mark of the data field must be found within 30 bytes in single density and 43 bytes in double density of the last ID field CRC byte; if not, the ID field is searched for and verified again followed by the Data Address Mark search. If after 5 revolutions the DAM cannot be found, the Record Not Found status bit is set and the operation is terminated.

VERIFY SEQUENCE



NOTE: IF TEST = 0, THERE IS NO 15MS DELAY
IF TEST = 1 AND CLK = 1 MHz, THERE IS A 30MS DELAY

TYPE I COMMAND FLOW

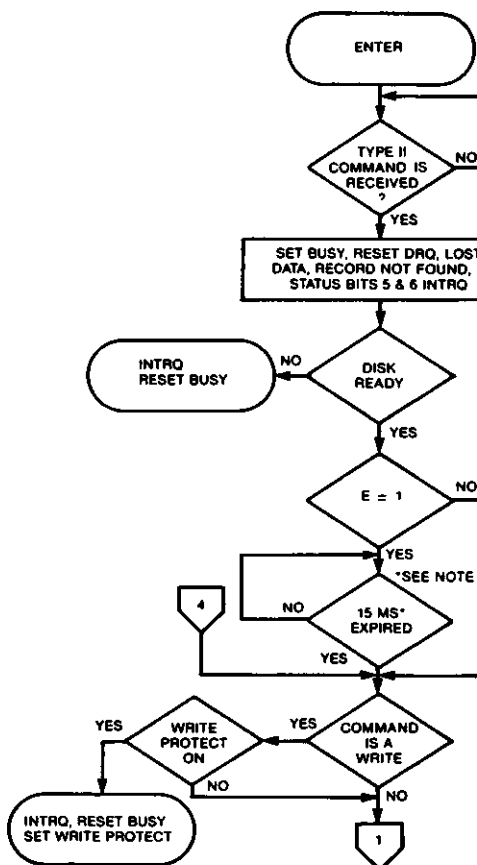
When the first character or byte of the data field has been shifted through the DSR, it is transferred to the DR, and DRQ is generated. When the next byte is accumulated in the DSR, it is transferred to the DR and another DRQ is generated. If the Computer has not read the previous contents of the DR before a new character is transferred that character is lost and the Lost Data Status bit is set. This sequence continues until the complete data field has been inputted to the computer. If there is a CRC error at the end of the data field, the CRC error status bit is set, and the command is terminated (even if it is a multiple record command).

At the end of the Read operation, the type of Data Address Mark encountered in the data field is recorded in the Status Register (Bit 5) as shown below:

STATUS BIT 5	
1	Deleted Data Mark
0	Data Mark

WRITE SECTOR

Upon receipt of the Write Sector command, the Busy status bit is set. When an ID field is encountered that has the correct track number, correct sector number, correct side number, and correct CRC, a DRQ is generated. The WD1773 counts off 11 bytes in single density and 22 bytes in double density from the CRC field and the Write Gate (WG) output is made active if the DRQ is serviced (i.e., the DR has been loaded by



NOTE: IF TEST = 0, THERE IS NO 15MS DELAY
IF TEST = 1 AND CLK = 1 MHz, THERE IS A 30MS DELAY

TYPE II COMMAND FLOW

the computer). If DRQ has not been serviced, the command is terminated and the Lost Data status bit is set. If the DRQ has been serviced, the WG is made active and six bytes of zeroes in single density and 12 bytes in double density are then written on the disk. At this time the Data Address Mark is then written on

the disk as determined by the a_0 field of the command as shown below:

a_0	Data Address Mark (Bit 0)
1	Deleted Data Mark
0	Data Mark

The WD1773 then writes the data field and generates DRQ's to the computer. If the DRQ is not serviced in time for continuous writing the Lost Data Status Bit is set and a byte of zeroes is written on the disk. The command is not terminated. After the last data byte has been written on the disk, the two-byte CRC is computed internally and written on the disk followed by one byte of logic ones in FM or in MFM. The WG output is then deactivated. The INTRQ will set 48 μ sec (MFM) or 96 μ sec (FM) after the last CRC byte is written. For partial sector writing, the proper method is to write the data and fill the balance with zeroes. By letting the chip fill the

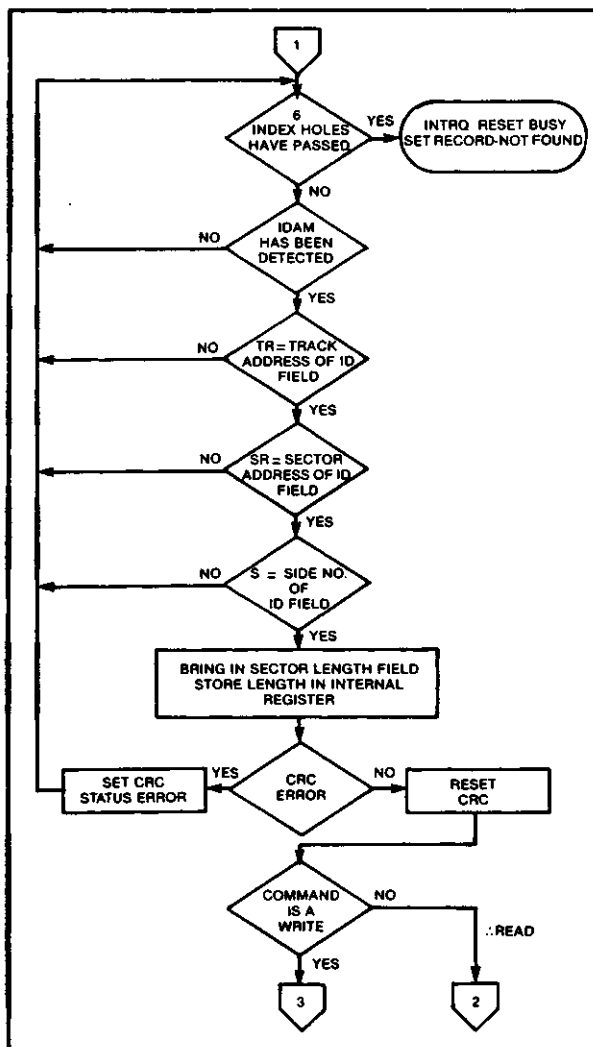
zeroes, errors may be masked by the lost data status and improper CRC Bytes.

TYPE III COMMANDS

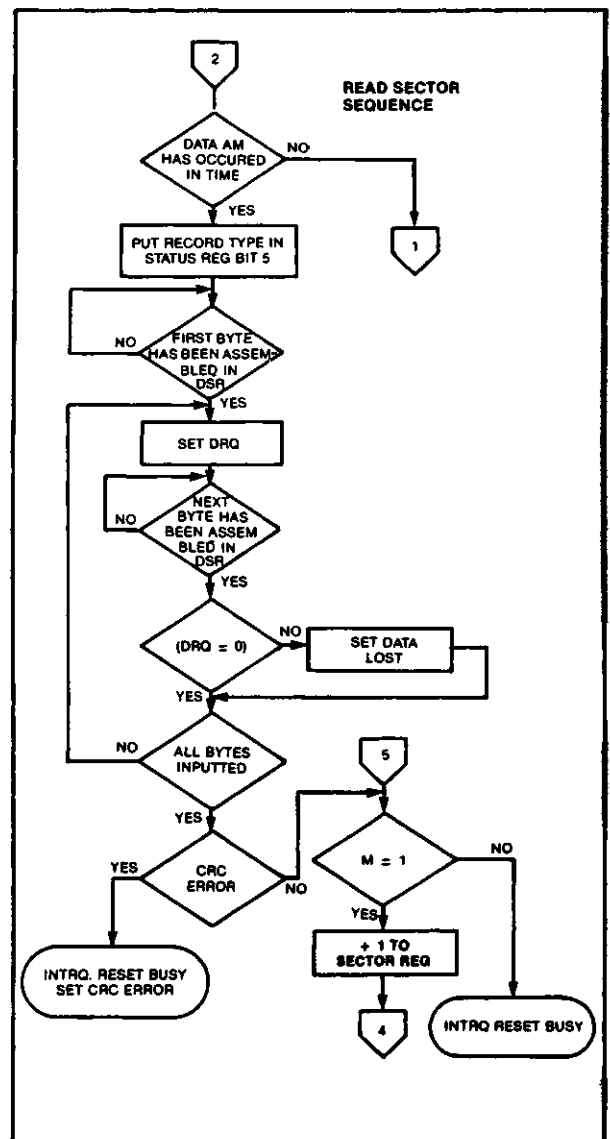
READ ADDRESS

Upon receipt of the Read Address command, the Busy Status Bit is set. The next encountered ID field is then read in from the disk, and the six data bytes of the ID field are assembled and transferred to the DR, and a DRQ is generated for each byte. The six bytes of the ID field are shown below:

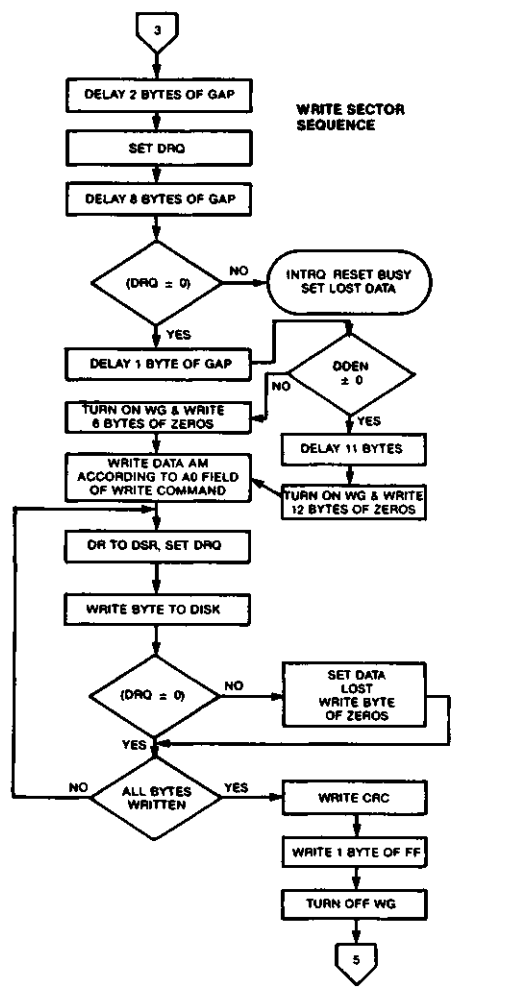
TRACK ADDR	SIDE NUMBER	SECTOR ADDRESS	SECTOR LENGTH	CRC 1	CRC 2
1	2	3	4	5	6



TYPE II COMMAND FLOW



TYPE II COMMAND FLOW



TYPE II COMMAND

Although the CRC characters are transferred to the computer, the WD1773 checks for validity and the CRC error status bit is set if there is a CRC error. The Track Address of the ID field is written into the sector register so that a comparison can be made by the user. At the end of the operation an interrupt is generated and the Busy Status is reset.

READ TRACK

Upon receipt of the READ track command, the Busy Status bit is set. Reading starts with the leading edge of the first encountered index pulse and continues until the next index pulse. All Gap, Header, and data bytes are assembled and transferred to the data register and DRQ's are generated for each byte. The accumulation of bytes is synchronized to each address mark encountered. An interrupt is generated at the completion of the command.

This command has several characteristics which

make it suitable for diagnostic purposes. They are: the Read Gate is not activated during the command; no CRC checking is performed; gap information is included in the data stream; the internal side compare is not performed; and the address mark detector is on for the duration of the command. Because the A.M. detector is always on, write splices or noise may cause the chip to look for an A.M. If an address mark does not appear on schedule the Lost Data status flag is set.

The ID A.M., ID field, ID CRC bytes, DAM, Data, and Data CRC Bytes for each sector will be correct. The Gap Bytes may be read incorrectly during write-splice time because of synchronization.

WRITE TRACK FORMATTING THE DISK

(Refer to section on Type III commands for flow diagrams.)

Formatting the disk is a relatively simple task when operating programmed I/O or when operating under DMA with a large amount of memory. Data and gap information must be provided at the computer interface. Formatting the disk is accomplished by positioning the RW head over the desired track number and issuing the Write Track command.

Upon receipt of the Write Track command, the Busy Status bit is set. Writing starts with the leading edge of the first encountered index pulse and continues until the next index pulse, at which time the interrupt is activated. The Data Request is activated immediately upon receiving the command, but writing will not start until after the first byte has been loaded into the Data Register. If the DR has not been loaded by the time the index pulse is encountered the operation is terminated making the device Not Busy, the Lost Data Status Bit is set, and the Interrupt is activated. If a byte is not present in the DR when needed, a byte of zeroes is substituted.

This sequence continues from one index mark to the next index mark. Normally, whatever data pattern appears in the data register is written on the disk with a normal clock pattern. However, if the WD1773 detects a data pattern of F5 thru FE in the data register, this is interpreted as data address marks with missing clocks or CRC generation.

The CRC generator is initialized when any data byte from F8 to FE is about to be transferred from the DR to the DSR in FM or by receipt of F5 in MFM. An F7 pattern will generate two CRC characters in FM or MFM. As a consequence, the patterns F5 thru FE must not appear in the gaps, data fields, or ID fields. Also, CRC's must be generated by an F7 pattern.

Disks may be formatted in IBM 3740 or System 34 formats with sector lengths of 128, 256, 512, or 1024 bytes.

TYPE IV COMMANDS

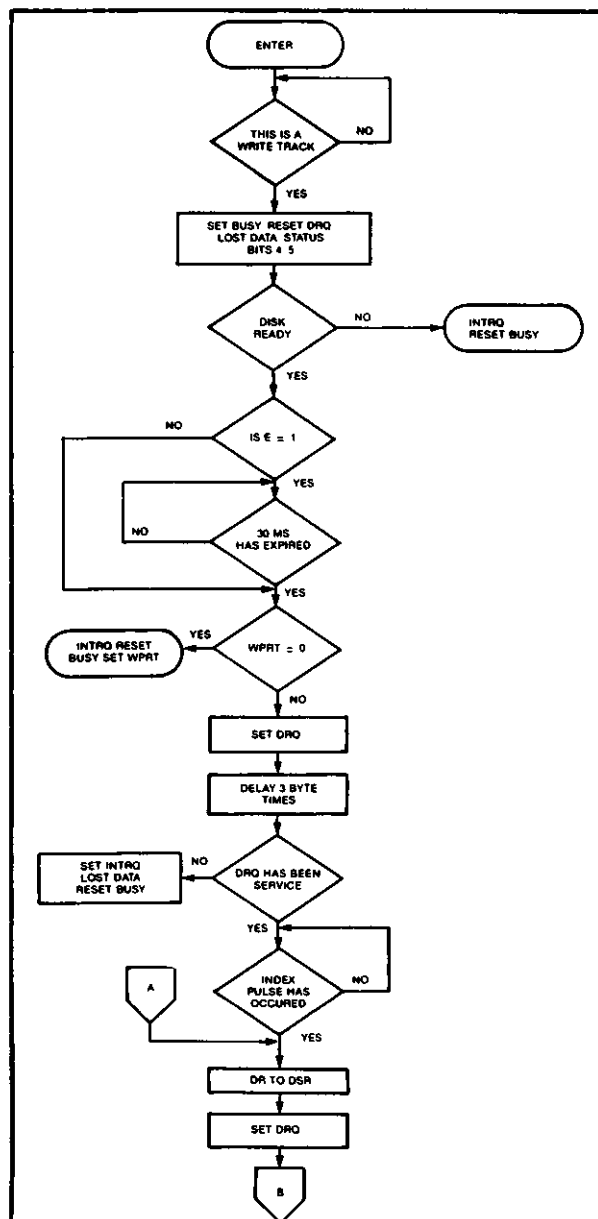
The Forced Interrupt command is generally used to

terminate a multiple sector read or write command or to insure Type I status in the status register. This command can be loaded into the command register at any time. If there is a current command under execution (busy status bit set) the command will be terminated and the busy status bit reset.

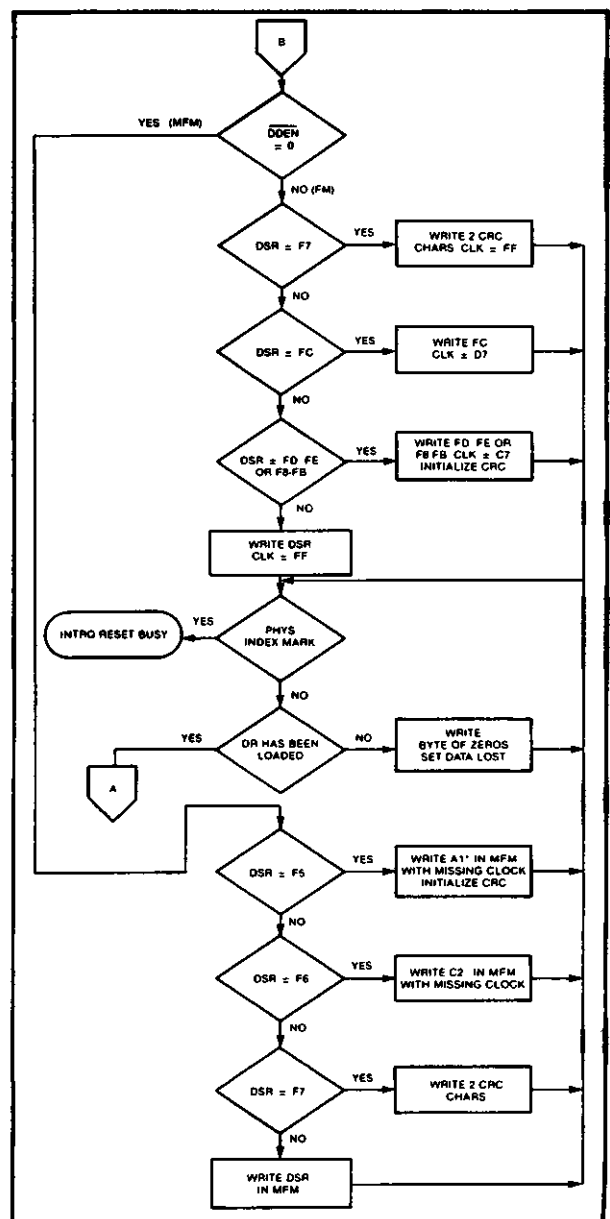
The lower four bits of the command determine the conditional interrupt as follows:

- l_0 = Not-Ready to Ready Transition
- l_1 = Ready to Not-Ready Transition
- l_2 = Every Index Pulse
- l_3 = Immediate interrupt

The conditional interrupt is enabled when the corresponding bit positions of the command ($l_3 - l_0$) are set to a 1. Then, when the condition for interrupt is met, the INTRQ line will go high signifying that the condition specified has occurred. If $l_3 - l_0$ are all set to zero (HEX D0), no interrupt will occur but any command presently under execution will be immediately terminated. When using the immediate interrupt condition ($l_3 = 1$) an interrupt will be immediately generated and the current command terminated. Reading the status or writing to the command register will not automatically clear the interrupt. The HEX D0 is the only command that will enable the



TYPE III COMMAND WRITE TRACK



TYPE III COMMAND WRITE TRACK

immediate interrupt (HEX D8) to clear on a subsequent load command register or read status register operation. Follow a HEX D8 with D0 command.

Wait 16 μ sec (double density) or 32 μ sec (single density) before issuing a new command after issuing a forced interrupt. Loading a new command sooner than this will nullify the forced interrupt.

Forced interrupt stops any command at the end of an internal micro-instruction and generates INTRQ when the specified condition is met. Forced interrupt will wait until ALU operations in progress are complete (CRC calculations, compares, etc.).

More than one condition may be set at a time. If for example, the READY TO NOT-READY condition (I1 = 1) and the Every Index Pulse (I2 = 1) are both set, the resultant command would be HEX "DA". The "OR" function is performed so that either a READY TO NOT-READY or the next Index Pulse will cause an interrupt condition.

STATUS REGISTER

Upon receipt of any command, except the Force Interrupt command, the Busy Status bit is set and the rest of the status bits are updated or cleared for the new command. If the Force Interrupt Command is received when there is a current command under execution, the Busy status bit is reset, and the rest of the status bits are unchanged. If the Force Interrupt command is received when there is not a current command under execution, the Busy Status bit is reset and the rest of the status bits are updated or cleared. In this case, Status reflects the Type I commands.

The user has the option of reading the status register through program control or using the DRQ line with DMA or interrupt methods. When the Data register is read the DRQ bit in the status register and the DRQ line are automatically reset. A write to the Data register also causes both DRQ's to reset.

The busy bit in the status may be monitored with a user program to determine when a command is complete, in lieu of using the INTRQ line. When using the INTRQ, a busy status check is not recommended because a read of the status register to determine the condition of busy will reset the INTRQ line.

The format of the Status Register is shown below:

(BITS)							
7	6	5	4	3	2	1	0
S7	S6	S5	S4	S3	S2	S1	S0

Status varies according to the type of command executed as shown in Table 4.

Because of internal sync cycles, certain time delays must be observed when operating under programmed I/O. They are: (times double when clock = 1 MHz)

Operation	Next Operation	Delay Req'd.	
		FM	MFM
Write to Command Reg.	Read Busy Bit (Status Bit 0)	48 μ s	24 μ s
Write to Command Reg.	Read Status Bits 1-7	64 μ s	32 μ s
Write Register	Read Any Register	32 μ s	16 μ s

IBM 3740 FORMAT — 128 BYTES/SECTOR

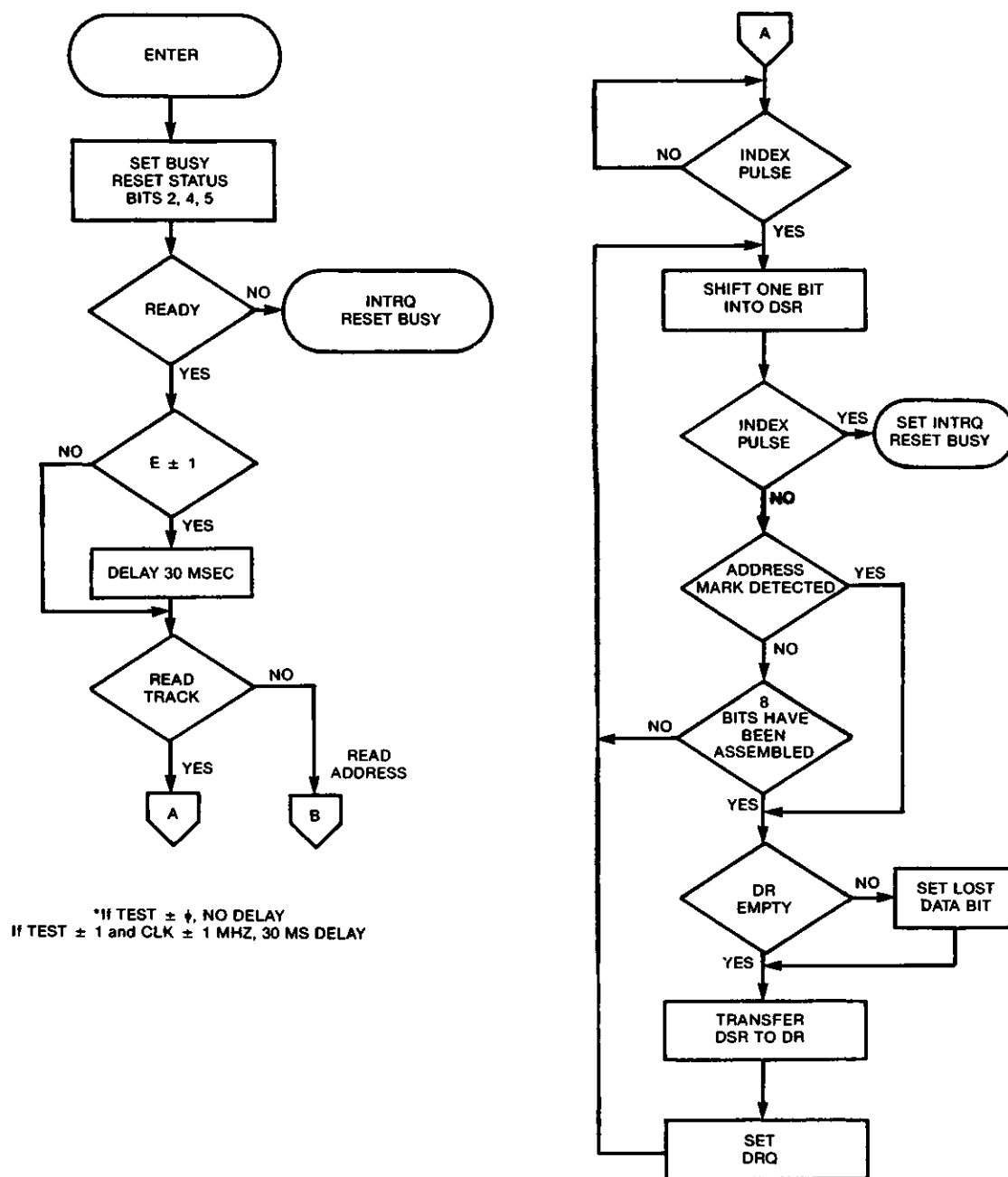
Shown below is the IBM single-density format with 128 bytes/sector. In order to format a diskette, the user must issue the Write Track command, and load the data register with the following values. For every byte to be written, there is one Data Request.

NUMBER OF BYTES	HEX VALUE OF BYTE WRITTEN
40	FF (or 00)*
6	00
1	FC (Index Mark)
26	FF (or 00)*
6	00
1	FE (ID Address Mark)
1	Track Number
1	Side Number (00 or 01)
1	Sector Number (1 thru 1A)
1	00 (Sector Length)
1	F7 (2 CRC's written)
11	FF (or 00)*
6	00
1	FB (Data Address Mark)
128	Data (IBM uses E5)
1	F7 (2 CRC's written)
27	FF (or 00)*
247**	FF (or 00)*

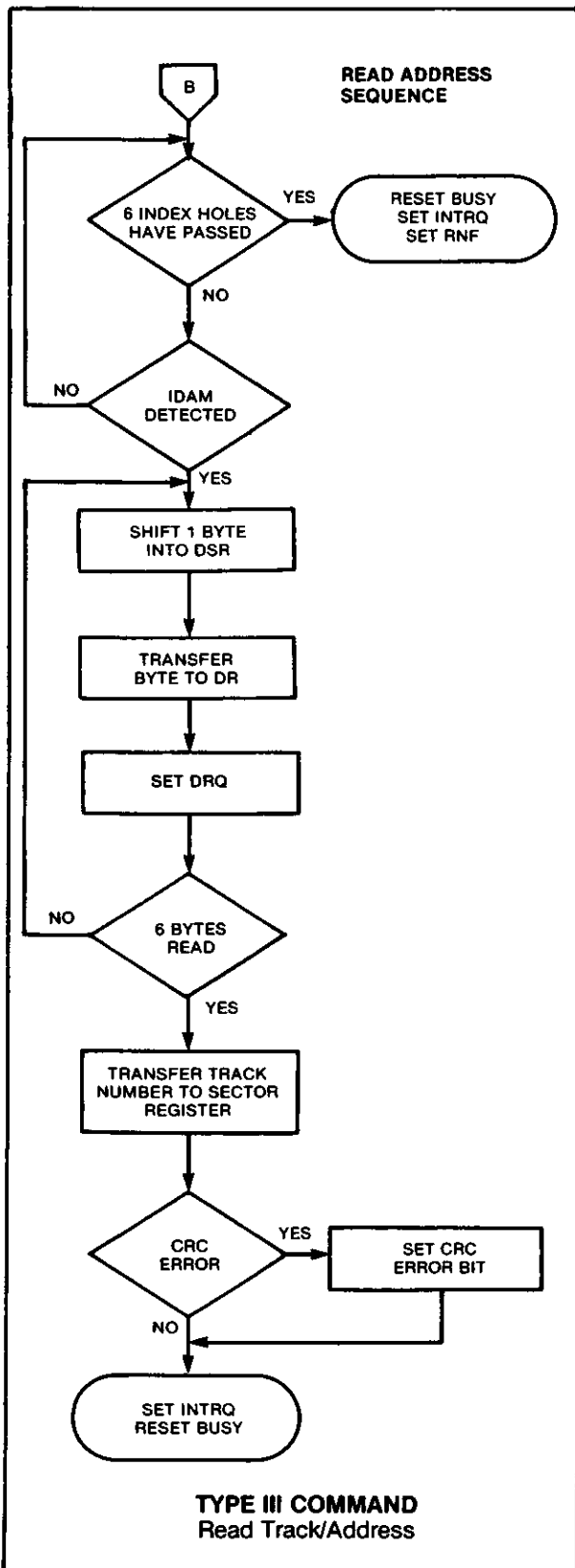
*Write bracketed field 26 times
 **Continue writing until WD1773 interrupts out.
 Approx. 247 bytes.

IBM SYSTEM 34 FORMAT — 256 BYTES/SECTOR

Shown below is the IBM dual-density format with 256 bytes/sector. In order to format a diskette the user must issue the Write Track command and load the data register with the following values. For every byte to be written, there is one data request.



TYPE III COMMAND
Read Track/Address



NUMBER OF BYTES	HEX VALUE OF BYTE WRITTEN
80	4E
12	00
3	F6 (Writes C2)
1	FC (Index Mark)
* 50	4E
12	00
3	F5 (Writes A1)
1	FE (ID Address Mark)
1	Track Number (0 thru 4C)
1	Side Number (0 or 1)
1	Sector Number (1 thru 1A)
1	01 (Sector Length)
1	F7 (2 CRCs written)
22	4E
12	00
3	F5 (Writes A1)
1	FB (Data Address Mark)
256	DATA
1	F7 (2 CRCs written)
54	4E
598**	4E

*Write bracketed field 26 times

**Continue writing until WD1773 interrupts out.
Approx. 598 bytes.

1. NON-IBM FORMATS

Variations in the IBM formats are possible to a limited extent if the following requirements are met:

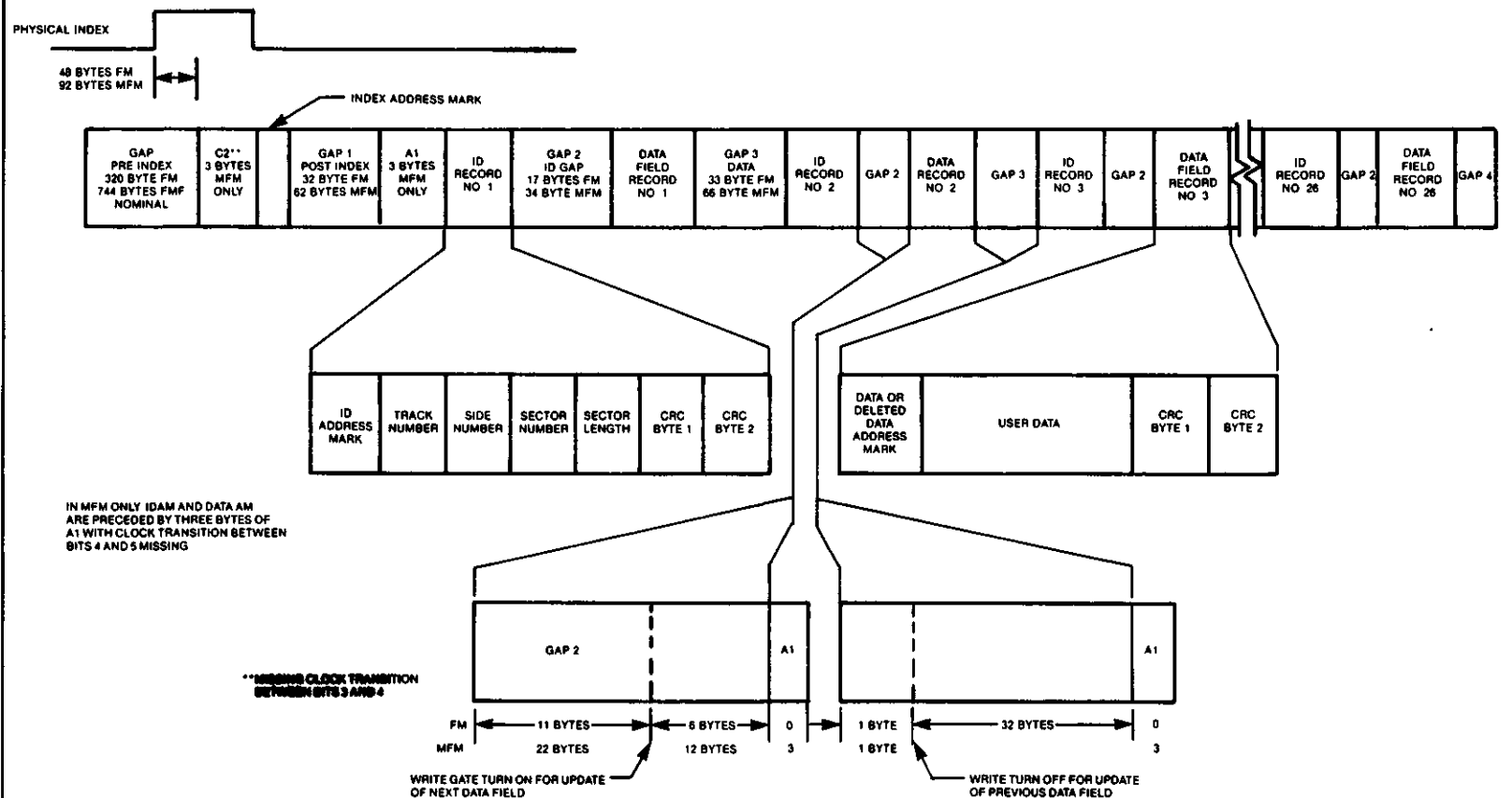
- 1) Sector size must be 128, 256, 512 or 1024 bytes.
- 2) Gap 2 cannot be varied from the IBM format.
- 3) 3 bytes of A1 must be used in MFM.

In addition, the Index Address Mark is not required for operation. Gap 1, 3, and 4 lengths can be as short as 2 bytes, however PLL lock up time, motor speed variation, write-splice area, etc. will add more bytes to each gap to achieve proper operation. It is recommended that the IBM format be used for highest system reliability.

	FM	MFM
Gap I	16 bytes FF	32 bytes 4E
Gap II	11 bytes FF	22 bytes 4E
*	6 bytes 00	12 bytes 00
*		3 bytes A1
Gap III**	10 bytes FF 4 bytes 00	24 bytes 4E 8 bytes 00 3 bytes A1
Gap IV	16 bytes FF	16 bytes 4E

*Byte counts must be exact.

**Byte counts are minimum, except exactly 3 bytes of A1 must be written.



IBM TRACK FORMAT

DC ELECTRICAL CHARACTERISTICS

MAXIMUM RATINGS

Storage Temperature -55°C to +125°C
Operating Temperature 0°C to 70°C Ambient

Maximum Voltage to Any Input
with Respect to V_{SS} (-15 to -0.3V)

DC OPERATING CHARACTERISTICS

T_A = 0°C to 70°C, V_{SS} = 0V, V_{CC} = +5V ± .25V

SYMBOL	CHARACTERISTIC	MIN.	MAX.	UNITS	CONDITIONS
I _{IL}	Input Leakage		10	μA	V _{IN} = V _{CC}
I _{OL}	Output Leakage		10	μA	V _{OUT} = V _{CC}
V _{IH}	Input High Voltage	2.0		V	
V _{IL}	Input Low Voltage		0.8	V	
V _{OH}	Output High Voltage	2.4		V	I _O = -100 μA
V _{OL}	Output Low Voltage		0.40	V	I _O = 1.6 mA
P _D	Power Dissipation		.75	W	
R _{PU}	Internal Pull-Up	100	1700	μA	V _{IN} = 0V
I _{CC}	Supply Current	75 (Typ)	150	mA	

AC TIMING CHARACTERISTICS

T_A = 0°C to 70°C, V_{SS} = 0V, V_{CC} = +5V ± .25V

READ ENABLE TIMING — RE such that : R_W = 1, CS = 0.

SYMBOL	CHARACTERISTIC	MIN.	TYP.	MAX.	UNITS	CONDITIONS
T _{RE}	RE Pulse Width of \overline{CS}	200			nsec	C _L = 50 pf
T _{DRR}	DRQ Reset from \overline{RE}		25	100	nsec	
T _{IRR}	INTRQ Reset from \overline{RE}			8000	nsec	
T _{DV}	Data Valid from \overline{RE}		100	200	nsec	C _L = 50 pf
T _{DOH}	Data Hold from \overline{RE}	50		150	nsec	C _L = 50 pf

Note: DRQ and INTRQ reset are from rising edge (lagging) of RE, whereas resets are from falling edge (leading) of WE.

WRITE ENABLE TIMING — WE such that : R_W = 0, CS = 0.

SYMBOL	CHARACTERISTIC	MIN.	TYP.	MAX.	UNITS	CONDITIONS
T _{AS}	Setup ADDR to \overline{CS}	50			nsec	
T _{SET}	Setup R \overline{W} to \overline{CS}	0			nsec	
T _{AH}	Hold ADDR from \overline{CS}	10			nsec	
T _{HL}	Hold R \overline{W} from \overline{CS}	0			nsec	
T _{WE}	\overline{WE} Pulse Width	200			nsec	
T _{DRW}	DRQ Reset from \overline{WE}		100	200	nsec	
T _{IRW}	INTRQ Reset from \overline{WE}			8000	nsec	
T _{DS}	Data Setup to \overline{WE}	150			nsec	
T _{DH}	Data Hold from \overline{WE}	0			nsec	

WRITE DATA TIMING:

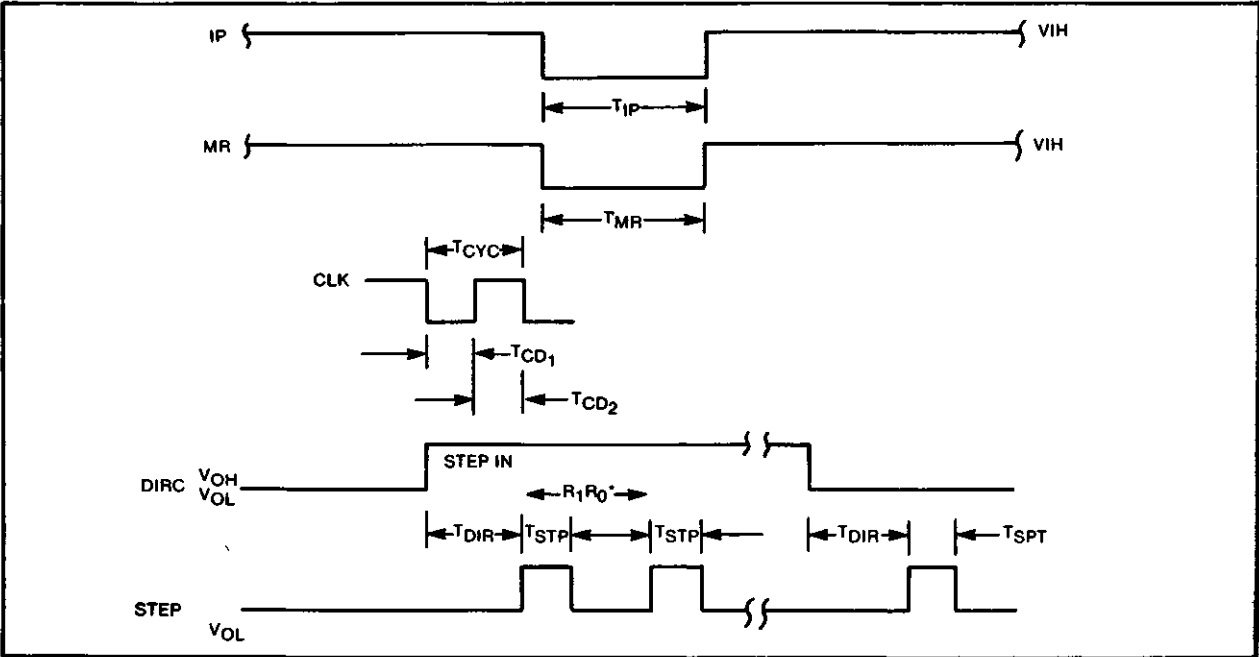
SYMBOL	CHARACTERISTIC	MIN.	TYP.	MAX.	UNITS	CONDITIONS
TWG	Write Gate to Write Data		4		μsec	FM
			2		μsec	MFM
TBC	Write Data Cycle Time		4,6,8		μsec	
TWF	Write Gate off from WD		4		μsec	FM
			2		μsec	MFM
TWP	Write Data Pulse Width		820		nsec	Early MFM
			690		nsec	Nominal MFM
			570		nsec	Late MFM
			1380		nsec	FM

INPUT DATA TIMING:

SYMBOL	CHARACTERISTIC	MIN.	TYP.	MAX.	UNITS	CONDITIONS
TPW	Raw Read Pulse Width	200		3000	nsec	
TBC	Raw Read Cycle Time	3000			nsec	

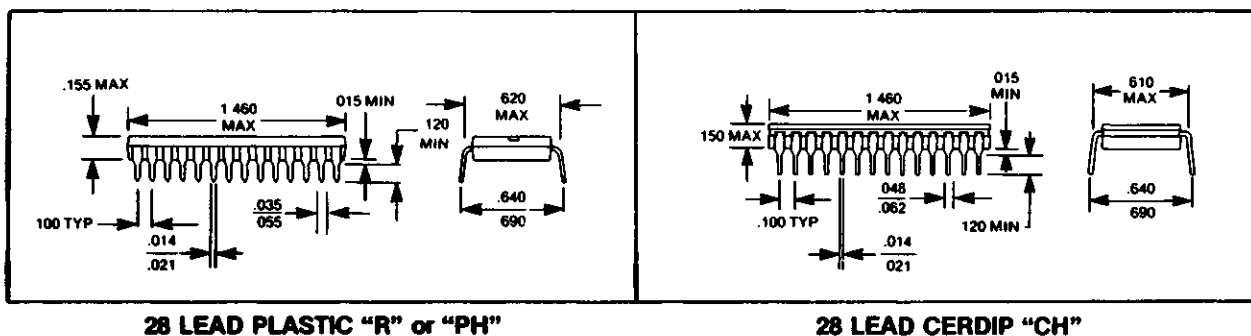
MISCELLANEOUS TIMING:

SYMBOL	CHARACTERISTIC	MIN.	TYP.	MAX.	UNITS	CONDITIONS
TCD ₁	Clock Duty (low)	50	67		nsec	(60/40)
TCD ₂	Clock Duty (high)	50	67		nsec	(40/60)
TSTP	Step Pulse Output		4		μsec	MFM
			8		μsec	FM
TDIR	Dir Setup to Step		24		μsec	MFM
			48		μsec	FM
TMR	Master Reset Pulse Width	50			μsec	
TIP	Index Pulse Width	20			μsec	



MISCELLANEOUS TIMING

Package Diagrams



Information furnished by Western Digital Corporation is believed to be accurate and reliable. However, no responsibility is assumed by Western Digital Corporation for its use; nor for any infringements of patents or other rights of third parties which may result from its use. No license is granted by implication or otherwise under any patent or patent rights of Western Digital Corporation. Western Digital Corporation reserves the right to change specifications at anytime without notice.

WESTERN DIGITAL
CORPORATION

2445 McCABE WAY
IRVINE, CALIFORNIA 92714

(714) 863-0102, TWX 910-595-1139

CP-DS/84221/1-84

Printed in U S A

WESTERN DIGITAL

C O R P O R A T I O N

WD9216-00/WD9216-01

Floppy Disk Data Separator — FDDS

PRELIMINARY

WD9216-00/WD9216-01

FEATURES

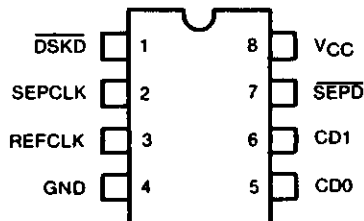
- PERFORMS COMPLETE DATA SEPARATION FUNCTION FOR FLOPPY DISK DRIVES
- SEPARATES FM OR MFM ENCODED DATA FROM ANY MAGNETIC MEDIA
- ELIMINATES SEVERAL SSI AND MSI DEVICES NORMALLY USED FOR DATA SEPARATION
- NO CRITICAL ADJUSTMENTS REQUIRED
- COMPATIBLE WITH WESTERN DIGITAL 179X, 176X AND OTHER FLOPPY DISK CONTROLLERS
- SMALL 8-PIN DUAL-IN-LINE PACKAGE
- +5 VOLT ONLY POWER SUPPLY
- TTL COMPATIBLE INPUTS AND OUTPUTS

GENERAL DESCRIPTION

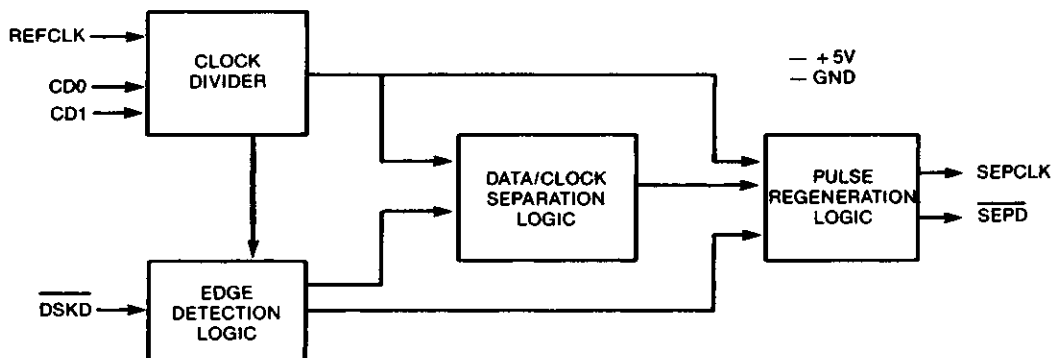
The Floppy Disk Data Separator provides a low cost solution to the problem of converting a single stream of pulses from a floppy disk drive into separate Clock and Data inputs for a Floppy Disk Controller.

The FDDS consists primarily of a clock divider, a long-term timing corrector, a short-term timing corrector, and reclocking circuitry. Supplied in an 8-pin Dual-In-Line package to save board real estate, the FDDS operates on +5 volts only and is TTL compatible on all inputs and outputs.

The WD9216 is available in two versions; the WD9216-00, which is intended for 5¼" disks and the WD9216-01 for 5½" and 8" disks.



PIN CONFIGURATION



FLOPPY DISK DATA SEPARATOR BLOCK DIAGRAM

ELECTRICAL CHARACTERISTICS**MAXIMUM RATINGS***

Operating Temperature Range. 0°C to +70°C
 Storage Temperature Range. -55°C to 125°C
 Positive Voltage on any Pin,
 with respect to ground +8.0V
 Negative Voltage on any Pin,
 with respect to ground -0.3V

*Stresses above those listed may cause permanent damage to the device. This is a stress rating only and functional operation of the device at these or at any other condition above those indicated in the operational sections of this specification is not implied.

NOTE: When powering this device from laboratory or system power supplies, it is important that the Absolute Maximum Ratings not be exceeded or device failure can result. Some power supplies exhibit voltage spikes or "glitches" on their outputs when the AC power is switched on and off. In addition, voltage transients on the AC power line may appear on the DC output. If this possibility exists it is suggested that a clamp circuit be used.

OPERATING CHARACTERISTICS ($T_A = 0^\circ\text{C}$ to 70°C , $V_{CC} = +5\text{V} \pm 5\%$, unless otherwise noted)

PARAMETER	MIN.	TYP.	MAX.	UNITS	COMMENTS
D.C. CHARACTERISTICS					
INPUT VOLTAGE LEVELS					
Low Level V_{IL}			0.8	V	
High Level V_{IH}	2.0			V	
OUTPUT VOLTAGE LEVELS					
Low Level V_{OL}			0.4	V	$I_{OL} = 1.6\text{mA}$
High Level V_{OH}	2.4			V	$I_{OH} = -100\mu\text{A}$
INPUT CURRENT					
Leakage I_{IL}			10	μA	$0 \leq V_{IN} \leq V_{DD}$
INPUT CAPACITANCE					
All Inputs			10	pF	
POWER SUPPLY CURRENT					
I_{DD}			50	mA	
A.C. CHARACTERISTICS					
Symbol					
f_{CY}	REFCLK Frequency	0.2	4.3	MHz	WD 9216-00
f_{CY}	REFCLK Frequency	0.2	8.3	MHz	WD 9216-01
t_{CKH}	REFCLK High Time	50	2500	ns	
t_{CKL}	REFCLK Low Time	50	2500	ns	
t_{SDON}	REFCLK to $\overline{\text{SEPD}}$ "ON" Delay		100	ns	
t_{SDOFF}	REFCLK to $\overline{\text{SEPD}}$ "OFF" Delay		100	ns	
t_{SPCK}	REFCLK to $\overline{\text{SEPCLK}}$ Delay			ns	
t_{DLL}	DSKD Active Low Time	0.1	100	μs	
t_{DLH}	DSKD Active High Time	0.2	100	μs	

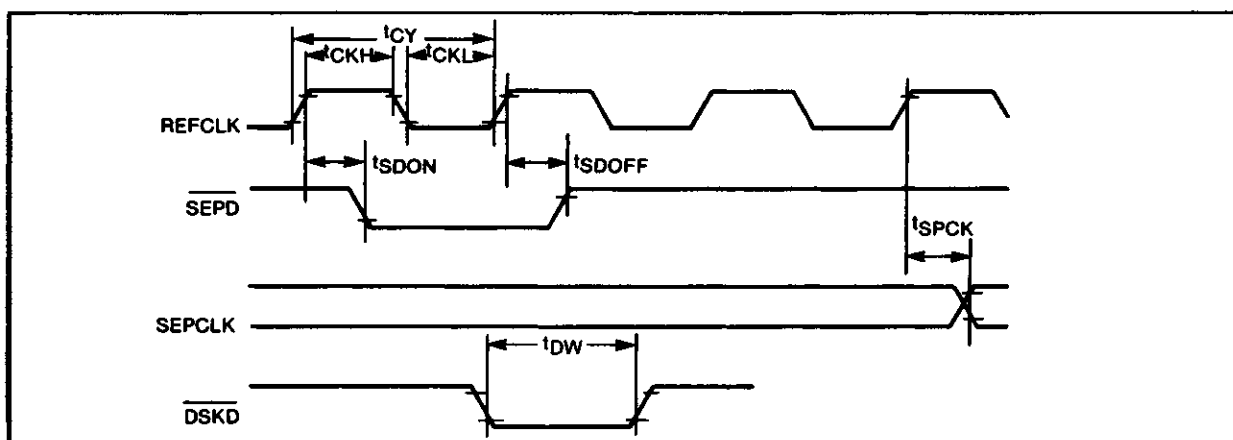


Figure 3. AC CHARACTERISTICS

DESCRIPTION OF PIN FUNCTIONS

PIN NUMBER	PIN NAME	SYMBOL	FUNCTION															
1	Disk Data	DSKD	Data input signal direct from disk drive. Contains combined clock and data waveform.															
2	Separated Clock	SEPCLK	Clock signal output from the FDDS derived from floppy disk drive serial bit stream.															
3	Reference Clock	REFCLK	Reference clock input.															
4	Ground	GND	Ground.															
5,6	Clock Divisor	CD0, CD1	CD0 and CD1 control the internal clock divider circuit. The internal clock is a submultiple of the REFCLK according to the following table: <table><tr><td>CD1</td><td>CD0</td><td>Divisor</td></tr><tr><td>0</td><td>0</td><td>1</td></tr><tr><td>0</td><td>1</td><td>2</td></tr><tr><td>1</td><td>0</td><td>4</td></tr><tr><td>1</td><td>1</td><td>8</td></tr></table>	CD1	CD0	Divisor	0	0	1	0	1	2	1	0	4	1	1	8
CD1	CD0	Divisor																
0	0	1																
0	1	2																
1	0	4																
1	1	8																
7	Separated Data	SEPD	SEPD is the data output of the FDDS															
8	Power Supply	VCC	+ 5 volt power supply															

WD9216-00/WD9216-01

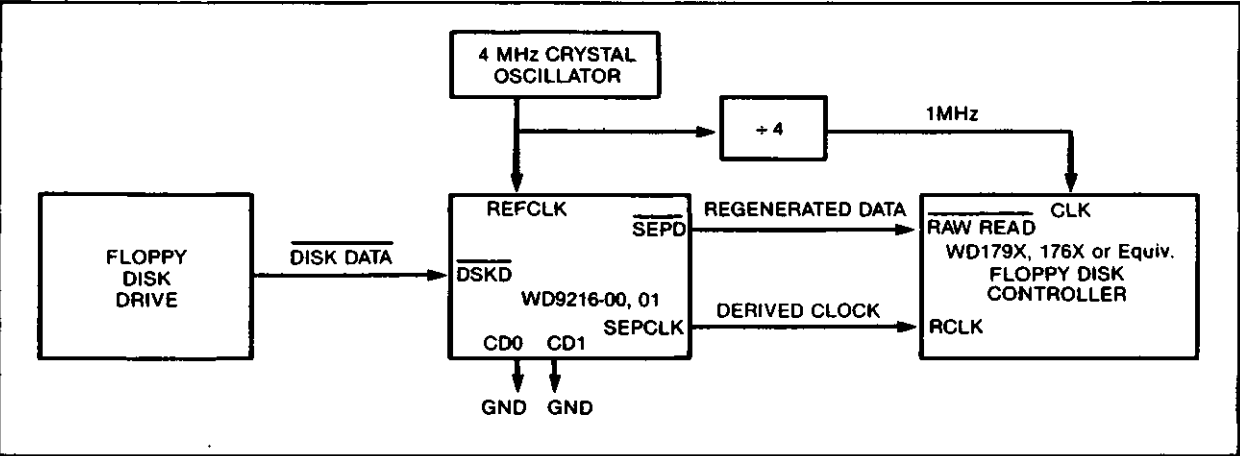


Figure 1.
TYPICAL SYSTEM CONFIGURATION
(5 1/4 inch Drive, Double Density)

OPERATION

A reference clock (REFCLK) of between 2 and 8 MHz is divided by the FDDS to provide an internal clock. The division ratio is selected by inputs CD0 and CD1. The reference clock and division ratio should be chosen per table 1. The FDDS detects the leading edges of the disk data pulses and adjusts the phase of the internal clock to provide the SEPARATED CLOCK output.

Separate short and long term timing correctors assure accurate clock separation. The internal clock frequency is nominally 16 times the SEPCLK frequency. Depending on the internal timing correction, the internal clock may be a minimum of 12 times to a maximum of 22 times the SEPCLK frequency. The reference clock (REFCLK) is divided to provide the internal clock according to pins CD0 and CD1.

TABLE 1:
CLOCK DIVIDER SELECTION TABLE

DRIVE (8" or 5¼")	DENSITY (DD or SD)	REFCLK MHz	CD1	CD0	REMARKS
8	DD	8	0	0	} Select either one
8	SD	8	0	1	
8	SD	4	0	0	
5¼	DD	8	0	1	} Select either one
5¼	DD	4	0	0	
5¼	SD	8	1	0	} Select any one
5¼	SD	4	0	1	
5¼	SD	2	0	0	

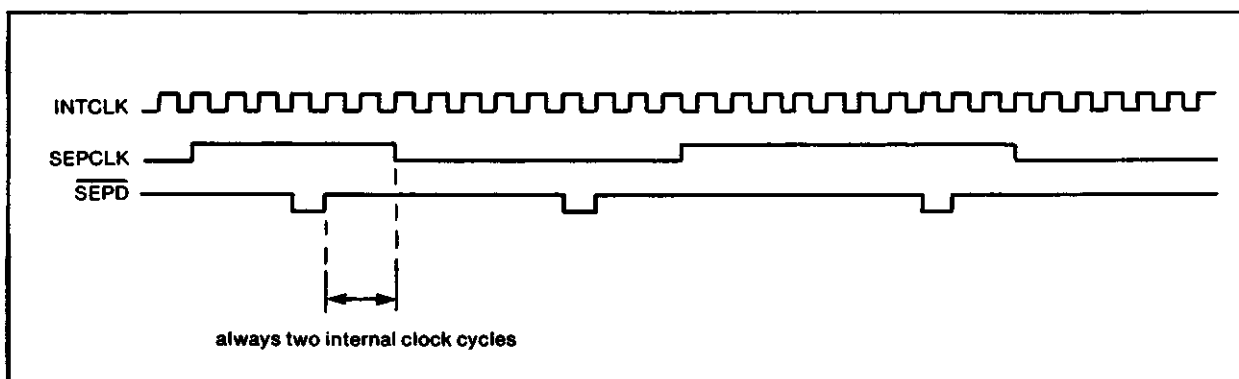


Figure 2.

See page 725 for ordering information.

Information furnished by Western Digital Corporation is believed to be accurate and reliable. However, no responsibility is assumed by Western Digital Corporation for its use, nor for any infringements of patents or other rights of third parties which may result from its use. No license is granted by implication or otherwise under any patent or patent rights of Western Digital Corporation. Western Digital Corporation reserves the right to change specifications at anytime without notice.

WESTERN DIGITAL

C O R P O R A T I O N

TR1863/TR1865

Universal Asynchronous Receiver/Transmitter (UART)

TR1863/TR1865

FEATURES

- SINGLE POWER SUPPLY — +5VDC
- D.C. TO 1 MHZ (64 KB) (STANDARD PART)
TR1863/5
- FULL DUPLEX OR HALF DUPLEX OPERATION
- AUTOMATIC INTERNAL SYNCHRONIZATION OF DATA AND CLOCK
- AUTOMATIC START BIT GENERATION
- EXTERNALLY SELECTABLE
Word Length
Baud Rate
Even/Odd Parity (Receiver/Verification —
Transmitter/Generation)
Parity Inhibit
One, One and One-Half, or Two Stop Bit
Generation (1½ at 5 Bit Level)
- AUTOMATIC DATA RECEIVED/TRANSMITTED
STATUS GENERATION
Transmission Complete
Buffer Register Transfer Complete
Received Data Available
Parity Error
Framing Error
Overrun Error
- BUFFERED RECEIVER AND TRANSMITTER
REGISTERS

THREE-STATE OUTPUTS

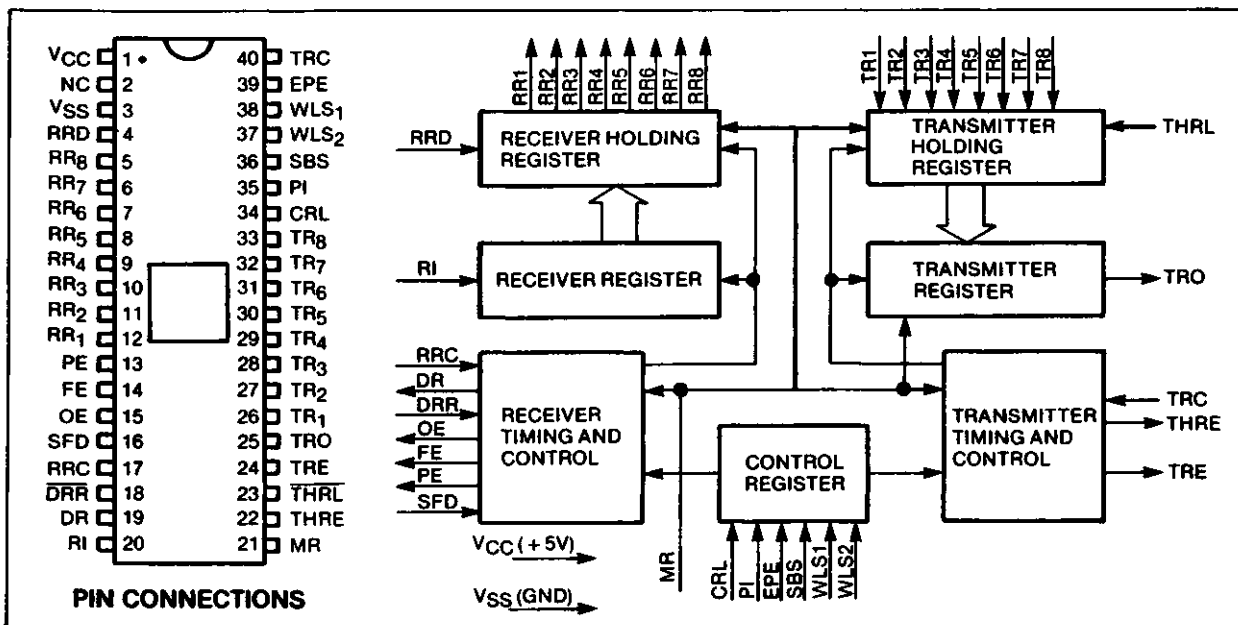
Receiver Register Outputs
Status Flags

TTL COMPATIBLE

- TR1865 HAS PULL-UP RESISTORS ON ALL INPUTS

APPLICATIONS

- PERIPHERALS
- TERMINALS
- MINI COMPUTERS
- FACSIMILE TRANSMISSION
- MODEMS
- CONCENTRATORS
- ASYNCHRONOUS DATA MULTIPLEXERS
- CARD AND TAPE READERS
- PRINTERS
- DATA SETS
- CONTROLLERS
- KEYBOARD ENCODERS
- REMOTE DATA ACQUISITION SYSTEMS
- ASYNCHRONOUS DATA CASSETTES



TR1863/TR1865 BLOCK DIAGRAM

GENERAL DESCRIPTION

The Universal Asynchronous Receiver/Transmitter (UART) is a general purpose, programmable or hardwired MOS/LSI device. The UART is used to convert parallel data to a serial data format on the transmit side, and converts a serial data format to parallel data on the receive side.

The serial format in order of transmission and reception is a start bit, followed by five to eight data bits, a parity bit (if selected) and one, one and one-half, or two stop bits.

Three types of error conditions are available on each received character: parity error, framing error (no valid stop bit) and overrun error.

The transmitter and receiver operate on external 16X clocks, where 16 clock times are equal to one bit time. The receiver clock is also used to sample in the center of the serial data bits to allow for line distortion.

Both transmitter and receiver are double buffered allowing a one character time maximum between a data read or write. Independent handshake lines for receiver and transmitter are also included. All inputs and outputs are TTL compatible with three-state outputs available on the receiver, and error flags for bussing multiple devices.

PIN DEFINITIONS

PIN NUMBER	NAME	SYMBOL	FUNCTION
1	POWER SUPPLY	VCC	+ 5 volts supply
2	NC	NC	No Internal Connection
3	GROUND	VSS	Ground = 0V
4	RECEIVER REGISTER DISCONNECT	RRD	A high level input voltage, V_{IH} , applied to this line disconnects the RECEIVER HOLDING REGISTER outputs from the RR1-8 data outputs (pins 5-12).
5-12	RECEIVER HOLDING REGISTER DATA	RR8-RR1	The parallel contents of the RECEIVER HOLDING REGISTER appear on these lines if a low-level input voltage, V_{IL} , is applied to RRD. For character formats of fewer than eight bits received characters are right-justified with RR1 (pin 12) as the least significant bit and the truncated bits are forced to a low level output voltage, V_{OL} .
13	PARITY ERROR	PE	A high level output voltage, V_{OH} , on this line indicates that the received parity differ from that which is programmed by the EVEN PARITY ENABLE control line (pin 39). This output is updated each time a character is transferred to the RECEIVER HOLDING REGISTER. PE lines from a number of arrays can be bussed together since an output disconnect capability is provided by Status Flag Disconnect line (pin 16).
14	FRAMING ERROR	FE	A high-level output voltage, V_{OH} , on this line indicates that the received character has no valid stop bit, i.e., the bit (if programmed) is not a high level voltage. This output is updated each time a character is transferred to the Receiver Holding Register, FE lines from a number of arrays can be bussed together since an output disconnect capability is provided by the Status Flag Disconnect line (pin 16).

PIN DEFINITIONS

TR1863/TR1865

PIN NUMBER	NAME	SYMBOL	FUNCTION
15	OVERRUN ERROR	OE	A high-level output voltage, V_{OH} , on this line indicates that the Data Received Flag (pin 19) was not reset before the next character was transferred to the Receiver Holding Register. OE lines from a number of arrays can be bussed together since an output disconnect capability is provided by the Status Flag Disconnect line (pin 16).
16	STATUS FLAGS DISCONNECT	SFD	A high-level input voltage, V_{IH} , applied to this pin disconnects the PE, FE, OE, DR and THRE allowing them to be buss connected.
17	RECEIVER REGISTER CLOCK	RRC	The receiver clock frequency is sixteen (16) times the desired receiver shift rate.
18	DATA RECEIVED RESET	\overline{DRR}	A low-level input voltage, V_{IL} , applied to this line resets the DR line.
19	DATA RECEIVED	DR	A high-level output voltage, V_{OH} , indicates that an entire character has been received and transferred to the RECEIVER HOLDING REGISTER.
20	RECEIVER INPUT	RI	Serial input data. A high-level input voltage, V_{IH} , must be present when data is not being received.
21	MASTER RESET	MR	This line is strobed to a high-level input voltage, V_{IH} , to clear the logic. It resets the TRANSMITTER and RECEIVER HOLDING REGISTERS, the TRANSMITTER REGISTER, FE, OE, PE, DR and sets TRO, THRE, and TRE to a high-level output voltage, V_{OH} .
22	TRANSMITTER HOLDING REGISTER EMPTY	THRE	A high-level output voltage, V_{OH} , on this line indicates the TRANSMITTER HOLDING REGISTER has transferred its contents to the TRANSMITTER REGISTER and may be loaded with a new character.
23	TRANSMITTER HOLDING REGISTER LOAD	\overline{THRL}	A low-level input voltage, V_{IL} , applied to this line enters a character into the TRANSMITTER HOLDING REGISTER. A transition from a low-level input voltage, V_{IL} , to a high-level input voltage, V_{IH} , transfers the character into the TRANSMITTER REGISTER if it is not in the process of transmitting a character. If a character is being transmitted, the transfer is delayed until its transmission is completed. Upon completion, the new character is automatically transferred simultaneously with the initiation of the serial transmission of the new character.
24	TRANSMITTER REGISTER EMPTY	TRE	A high-level output voltage, V_{OH} , on this line indicates that the TRANSMITTER REGISTER has completed serial transmission of a full character including STOP bit(s). It remains at this level until the start of transmission of the next character.

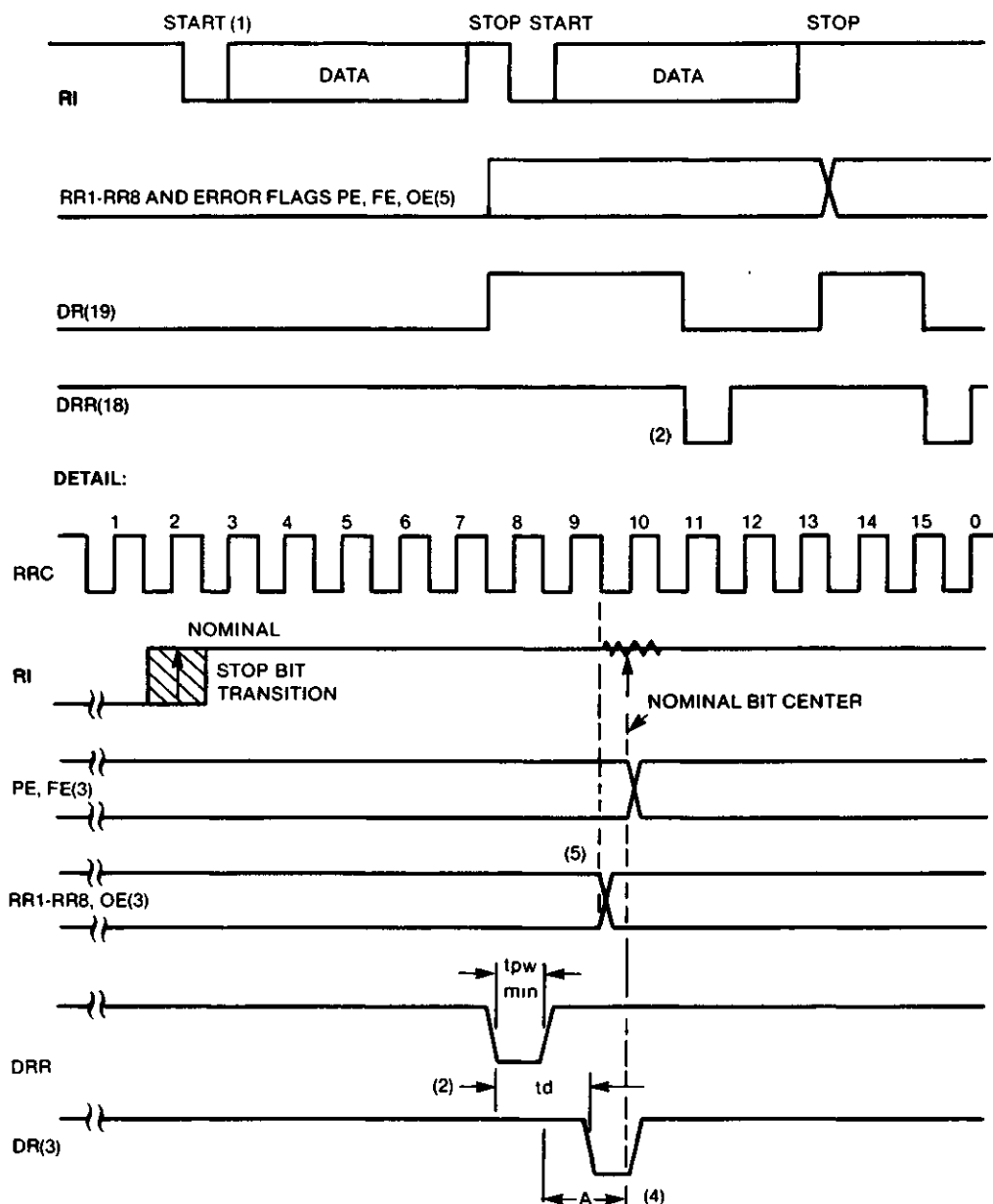
PIN DEFINITIONS

PIN NUMBER	NAME	SYMBOL	FUNCTION															
25	TRANSMITTER REGISTER OUTPUT	TRO	The contents of the TRANSMITTER REGISTER (START bit, DATA bits, PARITY bit, and STOP bits) are serially shifted out on this line. When no data is being transmitted, this line will remain at a high-level output voltage, V_{OH} . Start of transmission is defined as the transition of the START bit from a high-level output voltage V_{OH} to a low-level output voltage V_{OL} .															
26-33	TRANSMITTER REGISTER DATA INPUTS	TR ₁ -TR ₈	The character to be transmitted is loaded into the TRANSMITTER HOLDING REGISTER on these lines with the THRL Strobe. If a character of less than 8 bits has been selected (by WLS ₁ and WLS ₂), the character is right justified to the least significant bit, TR ₁ , and the excess bits are disregarded. A high-level input voltage, V_{IH} , will cause a high-level output voltage, V_{OH} , to be transmitted.															
34	CONTROL REGISTER LOAD	CRL	A high-level input voltage, V_{IH} , on this line loads the CONTROL REGISTER with the control bits (WLS ₁ , WLS ₂ , EPE, PI, SBS). This line may be strobed or hard wired to a high-level input voltage, V_{IH} .															
35	PARITY INHIBIT	PI	A high-level input voltage, V_{IH} , on this line inhibits the parity generation and verification circuits and will clamp the PE output (pin 13) to V_{OL} . If parity is inhibited, the STOP bit(s) will immediately follow the last data bit of transmission.															
36	STOP BIT(S) SELECT	SBS	This line selects the number of STOP bits to be transmitted after the parity bit. A high-level input voltage V_{IH} , on this line selects two STOP bits, and a low-level input voltage, V_{IL} , selects a single STOP bit. The TR1863 and TR1865 generate 1½ stop bits when word length is 5 bits and SBS is High V_{IH} .															
37-38	WORD LENGTH SELECT	WLS ₂ -WLS ₁	These two lines select the character length (exclusive of parity) as follows: <table><tr><td>WLS₂</td><td>WLS₁</td><td>Word Length</td></tr><tr><td>V_{IL}</td><td>V_{IL}</td><td>5 bits</td></tr><tr><td>V_{IL}</td><td>V_{IH}</td><td>6 bits</td></tr><tr><td>V_{IH}</td><td>V_{IL}</td><td>7 bits</td></tr><tr><td>V_{IH}</td><td>V_{IH}</td><td>8 bits</td></tr></table>	WLS ₂	WLS ₁	Word Length	V _{IL}	V _{IL}	5 bits	V _{IL}	V _{IH}	6 bits	V _{IH}	V _{IL}	7 bits	V _{IH}	V _{IH}	8 bits
WLS ₂	WLS ₁	Word Length																
V _{IL}	V _{IL}	5 bits																
V _{IL}	V _{IH}	6 bits																
V _{IH}	V _{IL}	7 bits																
V _{IH}	V _{IH}	8 bits																
39	EVEN PARITY ENABLE	EPE	This line determines whether even or odd PARITY is to be generated by the transmitter and checked by the receiver. A high-level input voltage, V_{IH} , selects even PARITY and a low-level input voltage, V_{IL} , selects odd PARITY.															
40	TRANSMITTER REGISTER	TRC	The transmitter clock frequency is sixteen (16) times the desired transmitter shift rate.															

(1) NOT VALID FOR 5.0 MHZ OPTION

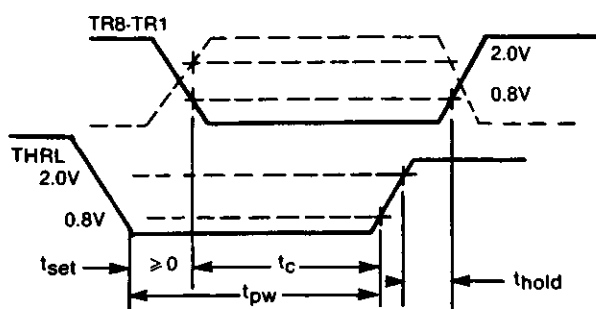
NOTE IT IS ADVISABLE TO CONSIDER
CASE II FOR $f_{CLOCK} > 40 \text{ MHz}$

TRANSMITTER TIMING

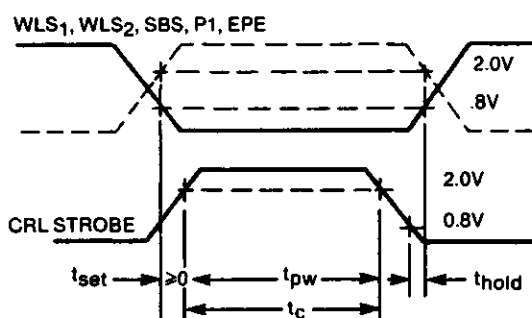


- (1) SEE APPLICATION FLAGS REPORT NO. 1 FOR DESCRIPTION OF START BIT DETECTION
- (2) THE DELAY BETWEEN DRR AND DR = t_d = 500 NS
- (3) DR, ERROR FLAGS, AND DATA ARE VALID AT THE NOMINAL CENTER OF THE FIRST STOP BIT
- (4) DRR SHOULD BE HIGH A MINIMUM OF "A" NS (ONE-HALF CLOCK TIME PLUS t_{pd}) PRIOR TO THE RISING EDGE OF DR
- (5) DATA AND OE PRECEDES DR, PE, AND FE FLAGS BY $\frac{1}{2}$ CLOCK
- (6) DATA FLAGS WILL REMAIN SET UNTIL A GOOD CHARACTER IS RECEIVED OR MASTER RESET IS APPLIED.

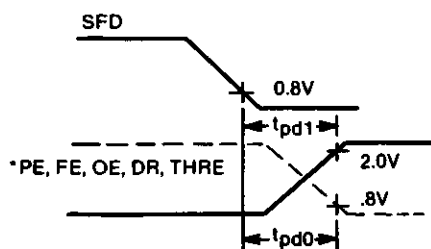
RECEIVER TIMING



DATA INPUT LOAD CYCLE

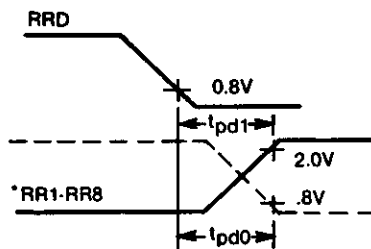


CONTROL REGISTER LOAD CYCLE



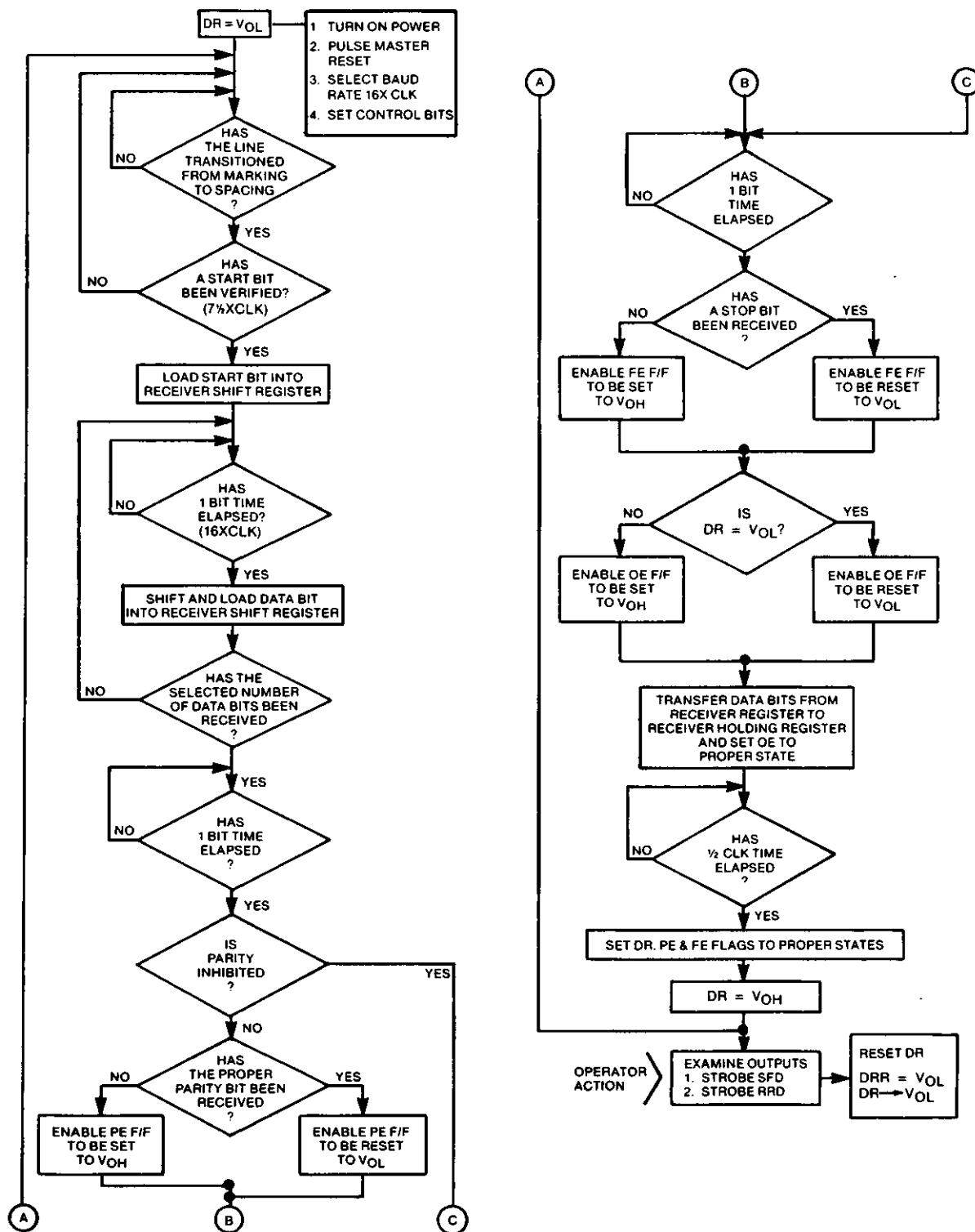
*OUTPUTS PE, FE, OE, DR, THRE ARE DISCONNECTED AT TRANSITION OF SFD FROM 0.8V TO 2.0V.

STATUS FLAG OUTPUT DELAYS

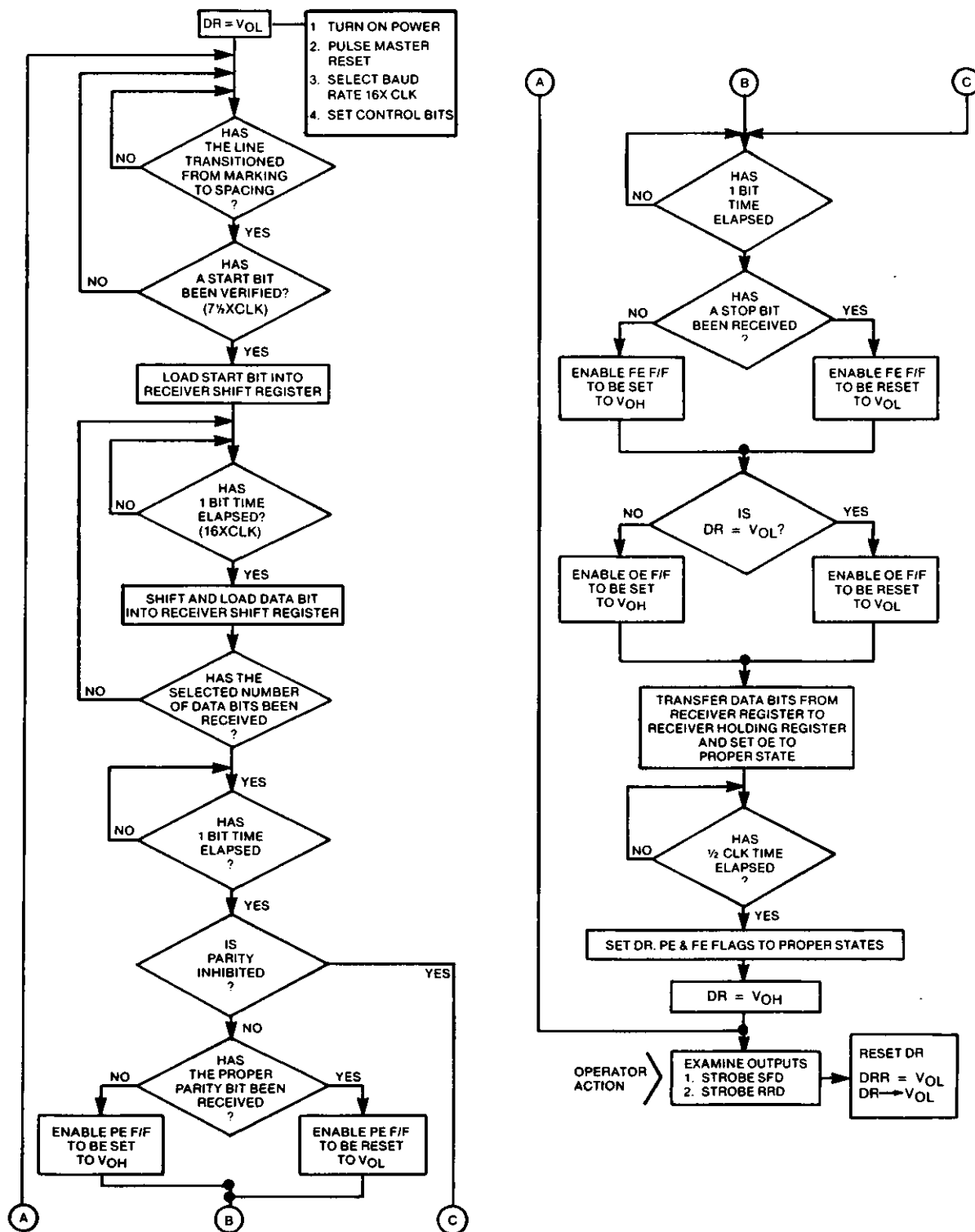


*RR1-RR3 ARE DISCONNECTED AT TRANSITION OF RRD FROM 0.8V TO 2.0V.

DATA OUTPUT DELAYS



RECEIVER FLOW CHART



RECEIVER FLOW CHART

ABSOLUTE MAXIMUM RATINGS

NOTE: These voltages are measured with respect to GND

Storage Temperature

Plastic - 55°C to + 125°C

Ceramic - 65°C to + 150°C

VCC Supply Voltage - 0.3V to + 7.0V

Input Voltage at any pin - 0.3V to + 7.0V

Operating Free-Air Temperature

T_A Range 0°C to 70°C

Lead Temperature (Soldering, 10 sec.) 300°C

ELECTRICAL CHARACTERISTICS

(VCC = 5V ± 5%, VSS = 0V)

SYMBOL	PARAMETER	TR1863/5		
	OPERATING CURRENT	MIN	MAX	CONDITIONS
ICC	Supply Current		35ma	VCC = 5.25V
	LOGIC LEVELS			
V _{IH}	Logic High	2.4V		VCC = 4.75V
V _{IL}	Logic Low		0.6V	
	OUTPUT LOGIC LEVELS			
V _{OH}	Logic High	2.4V		VCC = 4.75V, I _{OH} = 100μa
V _{OL}	Logic Low		0.4V	VCC = 5.25V, I _{OL} = 1.6 ma
I _{OC}	Output Leakage (High Impedance State)		± 10μa	V _{OUT} = 0V, V _{OUT} = 5V SFD = RRD = V _{IH}
I _{IL}	Low Level Input Current	100μa	1.6ma	V _{IN} = 0.4V TR 1865 only
			10μa	V _{IN} = V _{IL} , TR 1863 only
I _{IH}	High Level Input Current		- 10μa	V _{IN} = V _{IH} , TR 1863 only

SWITCHING CHARACTERISTICS
(See "Switching Waveforms")

SYMBOL	PARAMETER	MIN	MAX	CONDITIONS
f_{clock}	Clock Frequency			$V_{CC} = 4.75V$
	TR1863-00	DC	1.0 MHz	
	TR1863-02	DC	2.5 MHz	
	TR1863-04	DC	3.5 MHz	
	TR1863-06	DC	5.0 MHz	
	TR1865-00	DC	1.0 MHz	with internal pull-ups on all inputs
	TR1865-02	DC	2.5 MHz	with internal pull-ups on all inputs
	TR1865-04	DC	3.5 MHz	with internal pull-ups on all inputs
t_{pw}	Pulse Widths			with internal pull-ups on all inputs
	CRL	200 ns		
	THRL	200 ns		
	DRR	200 ns		
	MR	500 ns		
t_c	Coincidence Time	200 ns		
t_{hold}	Hold Time	20 ns		
t_{set}	Set Time	0		
	OUTPUT PROPAGATION DELAYS			
t_{pd0}	To Low State		250 ns	
t_{pd1}	To High State		250 ns	$C_L = 20 \text{ pf}$, plus one TTL load
	CAPACITANCE			
c_{in}	Inputs		20 pf	$f = 1 \text{ MHz}$, $V_{IN} = 5V$
c_o	Outputs		20 pf	$f = 1 \text{ MHz}$, $V_{IN} = 5V$

See page 725 for ordering information.

Information furnished by Western Digital Corporation is believed to be accurate and reliable. However, no responsibility is assumed by Western Digital Corporation for its use, nor for any infringements of patents or other rights of third parties which may result from its use. No license is granted by implication or otherwise under any patent or patent rights of Western Digital Corporation. Western Digital Corporation reserves the right to change specifications at anytime without notice.



Part 2 / Software

1/	Disk Organization	1
	Single Density Floppy Diskette	1
	Double Density Floppy Diskette	1
	5" 5-Meg Hard Disk	2
	Disk Space Available to the User	2
	Unit of Allocation	2
2/	Disk Files.	3
	Methods of File Allocation	3
	Dynamic Allocation.	3
	Pre-Allocation.	3
	Record Length	3
	Record Processing Capabilities	4
	Record Numbers.	4
3/	TRSDOS File Descriptions	5
	System Files (/SYS).	5
	Utility Programs	7
	Device Driver Programs.	7
	Filter Programs.	7
	Creating a Minimum Configuration Disk	7
4/	Device Access.	9
	Device Control Block (DCB)	9
	Memory Header	10
5/	Drive Access.	11
	Drive Code Table (DCT).	11
	Disk I/O Table	13
	Directory Records	13
	Granule Allocation Table (GAT)	16
	Hash Index Table (HIT)	18
6/	File Control	23
	File Control Block (FCB)	23
7/	TRSDOS Version 6 Programming Guidelines	27
	Converting to TRSDOS Version 6.	27
	Programming With Restart Vectors	29
	KFLAG\$ (BREAK)(PAUSE), and (ENTER) Interfacing.	29
	Interfacing to @ICNFG.	32
	Interfacing to @KITSK.	33
	Interfacing to the Task Processor	34
	Interfacing RAM Banks 1 and 2	36
	Device Driver and Filter Templates.	40
	@CTL Interfacing to Device Drivers	42

8/ Using the Supervisor Calls	45
Calling Procedure	45
Program Entry and Return Conditions	45
Supervisor Calls	46
Numerical List of SVCs	49
Alphabetical List of SVCs	52
Sample Programs	54
9/ Technical Information on TRSDOS Commands and Utilities	189
Appendix A/ TRSDOS Error Messages	193
Appendix B/ Memory Map	199
Appendix C/ Character Codes	201
Appendix D/ Keyboard Code Map	211
Appendix E/ Programmable SVCs	213
Appendix F/ Using SYS 13/SYS	215
Index	217

1/Disk Organization

TRSDOS Version 6 can be used with 5¼" single-sided floppy diskettes and with hard disk. Floppy diskettes can be either single- or double-density. See the charts below for the number of sectors per track, number of cylinders, and so on for each type of disk. (Sectors and cylinders are numbered starting with 0.)

Single-Density Floppy Diskette

Bytes per Sector	Sectors per Granule	Sectors per Track*	Granules per Track	Tracks per Cylinder	Cylinders per Drive	Total Bytes
256						256
	5					1,280
		(10)	2			2,560
				1		2,560
					40	102,400
256	5	(10)	2	1	40	102,400 (100K)**

Double-Density Floppy Diskette

Bytes per Sector	Sectors per Granule	Sectors per Track*	Granules per Track	Tracks per Cylinder	Cylinders per Drive	Total Bytes
256						256
	6					1,536
		(18)	3			4,608
				1		4,608
					40	184,320
256	6	(18)	3	1	40	184,320 (180K)**

*The number of sectors per track is not included in the calculation because it is equal to the number of sectors per granule times the number of granules per track. ($5 \times 2 = 10$ for single density, $6 \times 3 = 18$ for double density, and $16 \times 2 = 32$ for hard disk.)

**Note that this figure is the total amount of space in the given format. Keep in mind that an entire cylinder is used for the directory and at least one granule is used for the bootstrap code. This leaves 96.25K available for use on a single-density data disk and 174K on a double-density data disk.

5" 5-Meg Hard Disk

Note: Because of continual advancements in hard disk technology, the number of tracks and the number of tracks per cylinder may change. Therefore, any information that comes with your hard disk drive(s) supersedes the information in the table below.

Bytes per Sector	Sectors per Granule	Sectors per Track*	Granules per Track	Tracks per Cylinder	Cylinders per Drive	Total Bytes
256						256
	16					4,096
		(32)	2			8,192
				4		32,768
					153	5,013,504
256	16	(32)	2	4	153	5,013,504 (4,896K)

*The number of sectors per track is not included in the calculation because it is equal to the number of sectors per granule times the number of granules per track. ($5 \times 2 = 10$ for single density, $6 \times 3 = 18$ for double density, and $16 \times 2 = 32$ for hard disk.)

Disk Space Available to the User

One granule on cylinder 0 of each disk is reserved for the system. It contains information about where the directory is located on that disk. If the disk contains an operating system, then all of cylinder 0 is reserved. This area contains information used to load TRSDOS when you press the reset button.

One complete cylinder is reserved for the directory, the granule allocation table (GAT), and the hash index table (HIT). (On single-sided diskettes, one cylinder is the same as one track.) The number of this cylinder varies, depending on the size and type of disk. Also, if any portion of the cylinder normally used for the directory is flawed, TRSDOS uses another cylinder for the directory. You can find out where the FORMAT utility has placed the directory by using the Free :drive command.

On hard disks, an additional cylinder (cylinder 1) is reserved for use in case your disk drive requires service. This provides an area for the technician to write on the disk without harming any data. (If you bring your hard disk in for service, you should try to back up the contents of the disk first, just to be safe.)

Unit of Allocation

The smallest unit of disk space that the system can allocate to a file is a granule. A granule is made up of a set of sectors that are adjacent to one another on the disk. The number of sectors in a granule depends on the type and size of the disk. See the charts on the previous two pages for some typical sizes.

2/Disk Files

Methods of File Allocation

TRSDOS provides two ways to allocate disk space for files: dynamic allocation and pre-allocation.

Dynamic Allocation

With dynamic allocation, TRSDOS allocates granules only at the time of write. For example, when a file is first opened for output, no space is allocated. The first allocation of space is done at the first write. Additional space is added as required by further writes.

With dynamically allocated files, unused granules are de-allocated (recovered) when the file is closed.

Unless you execute the CREATE system command, TRSDOS uses dynamic allocation.

Pre-Allocation

With pre-allocation, the file is allocated a specified number of granules when it is created. Pre-allocated files can be created only by the system command CREATE. (See the *Disk System Owner's Manual* for more information on CREATE.)

TRSDOS automatically extends a pre-allocated file as needed. However, it does not de-allocate unused granules when a pre-allocated file is closed. To reduce the size of a pre-allocated file, you must copy it to a dynamically allocated file. The COPY (CLONE = N) system command does this automatically.

Files that have been pre-allocated have a 'C' by their names in a directory listing.

Record Length

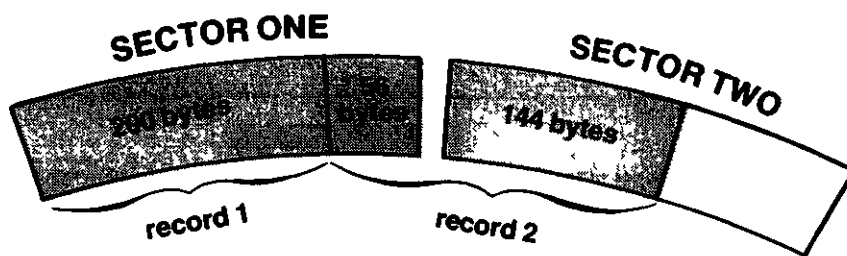
TRSDOS transfers data to and from disks one sector at a time. These sectors are 256-byte blocks, and are also called the system's "physical" records.

You deal with records that are 256 bytes in length or smaller, depending on what size record you want to work with. These are known as "logical" records.

You set the size of the logical records in a file when you open the file for the first time. The size is the number of bytes to be kept in each record. There may be from 1 to 256 bytes per logical record.

The operating system automatically accumulates your logical records and stores them in physical records. Since physical records are always 256 bytes in length, there may be one or more logical records stored in each physical record. When the records are read back from disk, the system automatically returns one logical record at a time. These actions are known as "blocking" and "de-blocking," or "spanning."

For example, if the logical record length is 200, sectors 1 and 2 look like this:



Since they are completely handled by the operating system, you do not need to concern yourself with physical records, sectors, granules, tracks, and so on. This is to your benefit, as the number of sectors per granule varies from disk to disk. Also, physical record lengths may change in future versions of TRSDOS, but the concept of logical records will not.

Note: All files are fixed-length record files with TRSDOS Version 6.

Record Processing Capabilities

TRSDOS allows both direct and sequential file access.

Direct access (sometimes called "random access") lets you process records in any sequence you specify.

Sequential access allows you to process records in sequence: record n , $n + 1$, $n + 2$, and so on. With sequential access, you do not specify a record number. Instead, TRSDOS accesses the record that follows the last record processed, starting with record 0.

With sequential access files, use the @READ supervisor call to read the next record, and the @WRITE or @VER supervisor call to write the next record. (When the file is first opened, processing starts at record 0. You can use @PEOF to position to the end of file.)

To read or write to a direct access file, use the @POSN supervisor call to position to a specified record. Then use @READ, @WRITE, or @VER as desired. Once @POSN has been used, the End of File (EOF) marker will not move, unless the file is extended by writing past the current EOF position.

Record Numbers

Using direct (random) access, you can access up to 65,536 records. Record numbers start at 0 and go to 65535.

Using a file sequentially, you can access up to 16,777,216 bytes. To calculate the number of records you can access sequentially, use the formula:

$$16,777,216 \div \text{logical record length} = \text{number of sequential records allowed}$$

Below are some examples.

If the LRL = 256, then:

$$16,777,216 \div 256 = 65,536 \text{ records}$$

If the LRL = 128, then:

$$16,777,216 \div 128 = 131,072 \text{ records}$$

If the LRL = 50, then:

$$16,777,216 \div 50 = 335,544 \text{ records}$$

If the LRL = 1, then:

$$16,777,216 \div 1 = 16,777,216 \text{ records}$$

3/TRSDOS File Descriptions

This section describes four types of files found on your TRSDOS master diskette (system files, utilities, driver programs, and filter programs) and explains their functions. It also describes how to construct a minimum system disk for running applications packages.

System Files (/SYS)

TRSDOS Version 6 would occupy considerable memory space if all of it were resident in memory at any one time. To minimize the amount of memory reserved for system use, TRSDOS uses overlays.

Using an overlay-driven system involves some compromise. While a user's application is in progress, different overlays may need to be loaded to perform certain activities requested of the system. This could cause the system to run slightly slower than a system which has more of its file access routines always resident in memory.

The use of overlays also requires that a SYSTEM disk usually be available in Drive 0 (the system drive). Since the disk containing the operating system and its utilities leaves little space available to the user, you may want to remove certain parts of the system software not needed while a particular application is running. You may in fact discover that your day-to-day operations need only a minimal TRSDOS configuration. The greater the number of system functions unnecessary for your application, the more space you can have available for a "working" system disk. Use the PURGE or REMOVE library command to eliminate unneeded system files from the disk.

The following paragraphs describe the functions performed by each system overlay. (In the display produced by the DIR (SYS) library command, the system overlays are identified by the file extension /SYS.)

Note: Two system files are put on the disk during formatting. They are DIR/SYS and BOOT/SYS. These files should *never* be copied from one disk to another or REMOVED. TRSDOS automatically updates any information necessary when performing a backup.

SYS0/SYS

This is not an overlay. It contains the resident part of the operating system (SYSRES). It is also needed to dynamically allocate file space used when writing files. Any disk used for booting the system *must* contain SYS0. It can be purged from disks not used for booting.

SYS1/SYS

This overlay contains the TRSDOS command interpreter and the routines for processing the @CMNDI, @CMNDR, @FEXT, @FSPEC, and @PARAM system vectors. This overlay must be available on all SYSTEM disks.

SYS2/SYS

This overlay is used for opening or initializing disk files and logical devices. It also contains routines for processing the @CKDRV, @GTDCB, and @RENAM system vectors, and routines for hashing file specifications and passwords. This overlay must be available on all SYSTEM disks.

SYS3/SYS

This overlay contains all of the system routines needed to close files and logical devices. It also contains the routines needed to service the @FNAME system vector. This overlay must not be removed from the disk.

SYS4/SYS

This overlay contains the system error dictionary. It is needed to issue such messages as "File not found," "Directory read error," etc. If you decide to remove this overlay from your working SYSTEM disk, all system errors will produce the error message "SYS ERROR." It is recommended that you not remove this overlay, especially since it occupies only one granule of space.

SYS5/SYS

This is the "ghost" debugger. It is needed if you intend to test out machine language application software by using the TRSDOS DEBUG library command. If your operation will not require this debugging tool, you may purge this overlay.

SYS6/SYS

This overlay contains all of the routines necessary to service the library commands identified as "Library A" by the LIB command. This represents the primary library functions. Only very limited use can be made of TRSDOS if this overlay is removed from your working SYSTEM disk.

SYS7/SYS

This overlay contains all of the routines necessary to service the library commands identified as "Library B" by the LIB command. A great deal of use can be made of TRSDOS even without this overlay. It performs specialized functions that may not be needed in the operation of specific applications. You can purge this overlay if you decide it is not needed on a working SYSTEM disk.

SYS8/SYS

This overlay contains all of the routines necessary to service the library commands identified as "Library C" by the LIB command. A great deal of use can be made of TRSDOS even without this overlay. It performs specialized functions that may not be needed in the operation of specific applications. You can purge this overlay if you decide it is not needed on a working SYSTEM disk.

SYS9/SYS

This overlay contains the routines necessary to service the extended DEBUG commands available after a DEBUG (EXT) is performed. This overlay may be purged if you will not need the extended DEBUG commands while running your application. If you remove SYS5/SYS, then you may as well remove SYS9/SYS, as it would serve no useful purpose.

SYS10/SYS

This system overlay contains the procedures necessary to service the request to remove a file. It should remain on your working SYSTEM disks.

SYS11/SYS

This overlay contains all of the procedures necessary to perform the Job Control Language execution phase. You may remove this overlay from your working disks if you do not intend to execute any JCL functions. If SYS6/SYS (which contains the DO command) has been removed, keeping this overlay would serve no purpose.

SYS12/SYS

This system overlay contains the routines that service the @DODIR, @GTMOD, and @RAMDIR system vectors. It should remain on your disks.

SYS13/SYS

This overlay is reserved for future system use. It contains no code and takes up no space on the disk. You may remove this overlay if you wish to free up its directory slot.

- SYS2 must be on the system disk if a configuration file is to be loaded.
- SYS11 must be present only if any JCL files will be used.
- All three libraries (SYS files 6, 7, and 8) may be purged if no library command will be used.
- SYS5 and SYS9 may be purged if the system DEBUG package is not needed.
- SYS0 may be removed from any disk not used for booting.
- SYS11 (the JCL processor) and SYS6 (containing the DO library command) must both be on the disk if the DO command is to be used. Also, if you remove SYS6, you may as well remove SYS11.
- SYS13 may be removed if you have not implemented an ECI, an IEP file, or if you do not intend to use them.

The presence of any utility, driver, or filter program is dependent upon your individual needs. You can save most of the TRSDOS features in a configuration file using the SYSTEM (SYSGEN) command, so the driver and filter programs will not be needed in run time applications. If you intend to use the HELP utility, your disk must contain the DOS/HLP file.

The owner (update) passwords for TRSDOS files are as follows:

File Type	Extension	Owner Password
System files	(/SYS)	LSIDOS
Filter files	(/FLT)	FILTER
Driver files	(/DVR)	DRIVER
Utility files	(/CMD)	UTILITY
BASIC		BASIC
BASIC overlays	(/OV\$)	BASIC
CONFIG/SYS		CCC
Drive Code Table Initializer	(/DCT)	UTILITY

Device Control Block (DCB)

The Device Control Block (DCB) is an area of memory that contains information used to interface the operating system with various logical devices. These devices include the keyboard (*KI), the video display (*DO), a printer (*PR), a communications line (*CL), and other devices that you may define.

The following information describes each assigned DCB byte.

DCB + 0 (TYPE Byte)

Bit 7 — If set to "1," the Device Control Block is actually a File Control Block (FCB) with the file open. Since DCBs and FCBs are similar, and devices may be routed to files, a "device" with this bit set indicates a routing to a file.

Bit 6 — If set to "1," the device defined by the DCB is filtered or is a device filter.

Bit 5 — If set to "1," the device defined by the DCB is linked.

Bit 4 — If set to "1," the device defined by the DCB is routed.

Bit 3 — If set to "1," the device defined by the DCB is a NIL device. Any output directed to the device is discarded. For any input request, the character returned is a null (ASCII value 0).

Bit 2 — If set to "1," the device defined by the DCB can handle requests generated by the @CTL supervisor call. See the section on Supervisor Calls for more information.

Bit 1 — If set to "1," the device defined by the DCB can handle output requests which normally come from the @PUT supervisor call.

Bit 0 — If set to "1," the device defined by the DCB can handle requests for input which normally come from the @GET supervisor call.

DCB + 1 and DCB + 2

Contain the address of the driver routine that supports the hardware assigned to this DCB. (In the case of a routed or linked device, the vector may point to another DCB.)

DCB + 3 through DCB + 5

Reserved for system use.

DCB + 6 and DCB + 7

These locations normally contain the two alphabetic characters of the devspec. The system uses the devspec as a reference in searching the device control block tables.

Memory Header

Modules that TRSDOS loads into memory (filters, drivers, and other memory modules such as a SPOOL buffer or the extended DEBUG code) are identified by a standard front-end header:

```
BEGIN:  JR    START           ;Go to actual code
                                   ;beginning
        DEFW END-1           ;Contains the highest byte
                                   ;of memory
                                   ;used by the module
        DEFB 10              ;Length of name, 1-15
                                   ;characters;
                                   ;bits 4-7 reserved for
                                   ;system use
        DEFM 'NAMESTRING'    ;Up to 15 alphanumeric
                                   ;characters, with the first
                                   ;character A-Z. This should
                                   ;be a unique name to
                                   ;positively identify the
                                   ;module.
MODDCB: DEFW $-$             ;DCB pointing to this
                                   ;module (if applicable)
        DEFW 0               ;Spare system pointer -
                                   ;RESERVED
;
;      Any additional data storage goes here
;
START:  Start of actual program code

END:    EQU $
```

As explained under the @GTMOD SVC in the "Supervisor Call" section, the location of a specific header can be found provided all modules that are put into memory use this header structure. You can locate the data area for a module by using @GTMOD to find the start of the header and then indexing in to the data area.

Drive Code Table (DCT)

TRSDOS uses a Drive Code Table (DCT) to interface the operating system with specific disk driver routines. Note especially the fields that specify the allocation scheme for a given drive. This data is essential in the allocation and accessibility of file records.

The DCT contains eight 10-byte positions — one for each logical drive designated 0-7. TRSDOS supports a standard configuration of two-floppy drives. You may have up to four floppy drives. This is the default initialization when TRSDOS is loaded.

Here is the Drive Code Table layout:

DCT + 0

This is the first byte of a 3-byte vector to the disk I/O driver routines. This byte is normally X'C3'. If the drive is disabled or has not been configured (see the SYSTEM command in the *Disk System Owner's Manual*), this byte is a RET instruction (X'C9').

DCT + 1 and DCT + 2

Contain the entry address of the routines that drive the physical hardware.

DCT + 3

Contains a series of flags for drive specifications.

Bit 7 — Set to "1" if the drive is software write protected, "0" if it is not. (See the SYSTEM command in the *Disk System Owner's Manual*.)

Bit 6 — Set to "1" for DDEN (double density), or "0" for SDEN (single density).

Bit 5 — Set to "1" if the drive is an 8" drive. Set to "0" if it is a 5¼" drive.

Bit 4 — A "1" causes the selection of the disk's second side. The first side is selected if this bit is "0." This bit value matches the side indicator bit in the sector header written by the Floppy Disk Controller (FDC).

Bit 3 — A "1" indicates a hard drive (Winchester). A "0" denotes a floppy drive (5¼" or 8").

Bit 2 — Indicates the time delay between selection of a 5¼" drive and the first poll of the status register. A "1" value indicates 0.5 second and a "0" indicates 1.0 second. See the SYSTEM command in the *Disk System Owner's Manual* for more details.

If the drive is a hard drive, this bit indicates either a fixed or removable disk: "1" = fixed, "0" = removable.

Bits 1 and 0 — Contain the step rate specification for the Floppy Disk Controller. (See the SYSTEM command in the *Disk System Owner's Manual*.) In the case of a hard drive, this field may indicate the drive address (0-3).

DCT + 4

Contains additional drive specifications.

Bit 7 — (Version 6.2 only) If "1", no @CKDRV is done when accessing the drive. If an application opens several files on a drive, this bit can be set to speed I/O on that drive after the first successful open is performed.

In versions prior to TRSDOS 6.2, this bit is reserved for future use. In order to maintain compatibility with future releases of TRSDOS, do not use this bit.

Bit 6 — If "1", the controller is capable of double-density mode.

Bit 5 — "1" indicates that this is a 2-sided floppy diskette; "0" indicates a 1-sided floppy disk. Do not confuse this bit with Bit 4 of DCT + 3. This bit shows if the disk is double-sided; Bit 4 of DCT + 3 tells the controller what side the current I/O is to be on.

If the hard drive bit (DCT + 3, Bit 3) is set, a "1" denotes double the cylinder count stored in DCT + 6. (This implies that a logical cylinder is made up of two physical cylinders.)

Bit 4 — If "1," indicates an alien (non-standard) disk controller.

Bits 0-3 — Contain the physical drive address by bit selection (0001, 0010, 0100, and 1000 equal logical Drives 0, 1, 2, and 3, respectively, in a default system). The system supports a translation only where no more than one bit can be set.

If the alien bit (Bit 4) is set, these bits may indicate the starting head number.

DCT + 5

Contains the current cylinder position of the drive. It normally stores a copy of the Floppy Disk Controller's track register contents whenever the FDC is selected for access to this drive. It can then be used to reload the track register whenever the FDC is reselected.

If the alien bit (DCT + 4, Bit 4) is set, DCT + 5 may contain the drive select code for the alien controller.

DCT + 6

Contains the highest numbered cylinder on the drive. Since cylinders are numbered from zero, a 35-track drive is recorded as X'22,' a 40-track drive as X'27,' and an 80-track drive as X'4F.' If the hard drive bit (DCT + 3, Bit 3) is set, the true cylinder count depends on DCT + 4, Bit 5. If that bit is a "1," DCT + 6 contains only half of the true cylinder count.

DCT + 7

Contains allocation information.

Bits 5-7 — Contain the number of heads for a hard drive.

Bits 0-4 — Contain the highest numbered sector relative to zero. A 10-sector-per-track drive would show X'09.' If DCT + 4, Bit 5 indicates 2-sided operation, the sectors per cylinder equals twice this number.

DCT + 8

Contains additional allocation information.

Bits 5-7 — Contain the number of granules per track allocated in the formatting process. If DCT + 4, Bit 5 indicates 2-sided operation, the granules per cylinder equals twice this number. For a hard drive, this number is the total granules per cylinder.

Bits 0-4 — Contain the number of sectors per granule that was used in the formatting operation.

DCT + 9

Contains the number of the cylinder where the directory is located. For any directory access, the system first attempts to use this value to read the directory. If this operation is unsuccessful, the system examines the BOOT granule (cylinder 0) directory address byte.

Bytes DCT + 6, DCT + 7, and DCT + 8 must relate without conflicts. That is, the highest numbered sector (+ 1) divided by the number of sectors per granule (+ 1) must equal the number of granules per track (+ 1).

Disk I/O Table

TRSDOS interfaces with hardware peripherals by means of software drivers. The drivers are, in general, coupled to the operating system through data parameters stored in the system's many tables. In this way, hardware not currently supported by TRSDOS can easily be supported by generating driver software and updating the system tables.

Disk drive sub-systems (such as controllers for 5¼" drives, 8" drives, and hard disk drives) have many parameters addressed in the Drive Code Table (DCT). Besides those operating parameters, controllers also require various commands (SELECT, SECTOR READ, SECTOR WRITE, and so on) to control the physical devices. TRSDOS has defined command conventions to deal with most commands available on standard Disk Controllers.

The function value (hexadecimal or decimal) you wish to pass to the driver should go in register B. The available functions are:

Hex	Dec	Function	Operation Performed
X'00'	0	DCSTAT	Test to see if drive is assigned in DCT
X'01'	1	SELECT	Select a new drive and return status
X'02'	2	DCINIT	Set to cylinder 0, restore, set side 0
X'03'	3	DCRES	Reset the Floppy Disk Controller
X'04'	4	RSTOR	Issue FDC RESTORE command
X'05'	5	STEPI	Issue FDC STEP IN command
X'06'	6	SEEK	Seek a cylinder
X'07'	7	TSTBSY	Test to see if requested drive is busy
X'08'	8	RDHDR	Read sector header information
X'09'	9	RDSEC	Read sector
X'0A'	10	VRSEC	Verify if the sector is readable
X'0B'	11	RDTRK	Issue an FDC track read command
X'0C'	12	HDFMT	Format the device
X'0D'	13	WRSEC	Write a sector
X'0E'	14	WRSYS	Write a system sector (for example, directory)
X'0F'	15	WRTRK	Issue an FDC track write command

Function codes X'10' to X'FF' are reserved for future use.

Directory Records (DIREC)

The directory contains information needed to access all files on the disk. The directory records section is limited to a maximum of 32 sectors because of physical limitations in the Hash Index Table. Two additional sectors in the directory cylinder are used by the system for the Granule Allocation Table and the Hash Index Table. The directory is contained on one cylinder. Thus, a 10-sector-per-cylinder formatted disk has, at most, eight directory sectors. See the sec-

tion on the Hash Index Table for the formula to calculate the number of directory sectors.

A directory record is 32 bytes in length. Each directory sector contains eight directory records ($256/32 = 8$). On system disks, the first two directory records of the first eight directory sectors are reserved for system overlays. The total number of files possible on a disk equals the number of directory sectors times eight (since $256/32 = 8$). The number available for use is reduced by 16 on system disks to account for those record slots reserved for the operating system. The following table shows the directory record capacity (file capacity) of each format type. The dash suffix (-1 or -2) on the items in the density column represents the number of sides formatted (for example, SDEN-1 means single density, 1-sided).

	Sectors per Cylinder	Directory Sectors	User Files on Data Disk**	User Files on SYS Disk
5" SDEN-1	10	8	62	48
5" SDEN-2	20	18	142	128
5" DDEN-1	18	16	126	112
5" DDEN-2	36	32	254	240
8" SDEN-1	16	14	110	96
8" SDEN-2	32	30	238	224
8" DDEN-1	30	28	222	208
8" DDEN-2	60	32	254	240
Hard Disk*				

*Hard drive format depends on the drive size and type, as well as the user's division of the physical drive into logical drives. After setting up and formatting the drive, you can use the FREE library command to see the available files.

**Note: Two directory records are reserved for BOOT/SYS and DIR/SYS, and are included in the figures for this column.

TRSDOS Version 6 is upward compatible with other TRSDOS 2.3 compatible operating systems in its directory format. The data contained in the directory has been extended. An SVC is included to either display an abbreviated directory or place its data in a user-defined buffer area. For detailed information, see the @DODIR and @RAMDIR SVCs.

The following information describes the contents of each directory field:

DIR + 0

Contains all attributes of the designated file.

Bit 7 — If "0," this flag indicates that the directory record is the file's primary directory entry (FPDE). If "1," the directory record is one of the file's extended directory entries (FXDE). Since a directory entry can contain information on up to four extents (see notes on the extent fields, beginning with DIR + 22), a file that is fractured into more than four extents requires additional directory records.

Bit 6 — Specifies a SYStem file if "1," a nonsystem file if "0."

Bit 5 — If set to "1," indicates a Partition Data Set (PDS) file.

Bit 4 — Indicates whether the directory record is in use or not. If set to "1," the record is in use. If "0," the directory record is not active, although it may appear to contain directory information. In contrast to some operating systems that zero out the directory record when you remove a file, TRSDOS only resets this bit to zero.

Bit 3 — Specifies the visibility. If "1," the file is INVisible to a directory display or other library function where visibility is a parameter. If a "0," then the file is VISible. (The file can be referenced if specified by name by an @INIT or @OPEN SVC.)

Bits 0-2 — Contain the USER protection level of the file. The 3-bit binary value is one of the following:

0 = FULL	2 = RENAME	4 = UPDATE	6 = EXECUTE
1 = REMOVE	3 = WRITE	5 = READ	7 = NO ACCESS

DIR + 1

Contains various file flags and the month field of the packed date of last modification.

Bit 7 — Set to "1" if the file was "CREATED" (see CREATE library command in the *Disk System Owner's Manual*). Since the CREATE command can reference a file that is currently existing but non-CREATED, it can turn a non-CREATED file into a CREATED one. You can achieve the same effect by changing this bit to a "1."

Bit 6 — If set to "1," the file has not been backed up since its last modification. The BACKUP utility is the only TRSDOS facility that resets this flag. It is set during the close operation if the File Control Block (FCB + 0, Bit 2) shows a modification of file data.

Bit 5 — If set to "1," indicates a file in an open condition with UPDATE access or greater.

Bit 4 — If the file was modified during a session where the system date was not maintained, this bit is set to "1." This specifies that the packed date of modification (if any) stored in the next three fields is not the actual date the modification occurred. If this bit is "1," the directory command displays plus signs (+) between the date fields.

Bits 0-3 — Contain the binary month of the last modification date. If this field is a zero, DATE was not set when the file was established or since if it was updated.

DIR + 2

Contains the remaining date of modification fields.

Bits 3-7 — Contain the binary day of last modification.

Bits 0-2 — Contain the binary year minus 80. For example, 1980 is coded as 000, 1981 as 001, 1982 as 010, and so on.

DIR + 3

Contains the end-of-file offset byte. This byte and the ending record number (ERN) form a pointer to the byte position that follows the last byte written. This assumes that programmers, interfacing in machine language, properly maintain the next record number (NRN) offset pointer when the file is closed.

DIR + 4

Contains the logical record length (LRL) specified when the file was generated or when it was later changed with a CLONE parameter.

DIR + 5 through DIR + 12

Contain the name field of the filespec. The filename is left justified and padded with trailing blanks.

DIR + 13 through DIR + 15

Contain the extension field of the filespec. It is left justified and padded with trailing blanks.

DIR + 16 and DIR + 17

Contain the OWNER password hash code.

DIR + 18 and DIR + 19

Contain the USER password hash code. The protection level in DIR + 0 is associated with this password.

DIR + 20 and DIR + 21

Contain the ending record number (ERN), which is based on full sectors. If the ERN is zero, it indicates that no writing has taken place (or that the file was not closed properly). If the LRL is not 256, the ERN represents the sector where the EOF occurs. You should use ERN minus 1 to account for a value relative to sector 0 of the file.

DIR + 22 and DIR + 23

This is the first extent field. Its contents indicate which cylinder stores the first granule of the extent, which relative granule it is, and how many contiguous grans are in use in the extent.

DIR + 22 — Contains the cylinder value for the starting gran of that extent.

DIR + 23, Bits 5-7 — Contain the number of the granule in the cylinder indicated by DIR + 22 which is the first granule of the file for that extent. This value is relative to zero ("0" denotes the first gran, "1" denotes the second, and so on).

DIR + 23, Bits 0-4 — Contain the number of contiguous granules, relative to 0 ("0" denotes one gran, "1" denotes two, and so on). Since the field is five bits, it contains a maximum of X'1F' or 31, which represents 32 contiguous grans.

DIR + 24 and DIR + 25

Contain the fields for the second extent. The format is identical to that for Extent 1.

DIR + 26 and DIR + 27

Contain the fields for the third extent. The format is identical to that for Extent 1.

DIR + 28 and DIR + 29

Contain the fields for the fourth extent. The format is identical to that for Extent 1.

DIR + 30

This is a flag noting whether or not a link exists to an extended directory record. If no further directory records are linked, the byte contains X'FF'. A value of X'FE' in this byte establishes a link to an extended directory entry. (See "Extended Directory Records" below.)

DIR + 31

This is the link to the extended directory entry noted by the previous byte. The link code is the Directory Entry Code (DEC) of the extended directory record. The DEC is actually the position of the Hash Index Table byte mapped to the directory record. For more information, see the section "Hash Index Table."

Extended Directory Records

Extended directory records (FXDE) have the same format as primary directory records, except that only Bytes 0, 1, and 21-31 are utilized. Within Byte 0, only Bits 4 and 7 are significant. Byte 1 contains the DEC of the directory record of which this is an extension. An extended directory record may point to yet another directory record, so a file may contain an "unlimited" number of extents (limited only by the total number of directory records available).

Granule Allocation Table (GAT)

The Granule Allocation Table (GAT) contains information on the free and assigned space on the disk. The GAT also contains data about the formatting used on the disk.

A disk is divided into cylinders (tracks) and sectors. Each cylinder has a specified number of sectors. A group of sectors is allocated whenever additional space is needed. This group is called a granule. The number of sectors per granule depends on the total number of sectors available on a logical drive. The GAT provides for a maximum of eight granules per cylinder.

In the GAT bytes, each bit set to "1" indicates a corresponding granule in use (or locked out). Each bit reset to "0" indicates a granule free to be used. In a GAT byte, bit 0 corresponds to the first relative granule, bit 1 to the second relative granule, bit 2 the third, and so on. A 5¼" single density diskette is formatted at 10 sectors per cylinder, 5 sectors per granule, 2 granules per cylinder. Thus, that configuration uses only bits 0 and 1 of the GAT byte. The remainder of the GAT byte contains all 1's, denoting unavailable granules. Other formatting conventions are as follows:

	Sectors per Cylinder	Sectors per Granule	Granules per Cylinder	Maximum No. of Cylinders
5" SDEN	10	5	2	80
5" DDEN	18	6	3	80
8" SDEN	16	8	2	77
8" DDEN	30	10	3	77
Hard Disk	32	16	8	153

*Hard drive format depends on the drive size and type, as well as the user's division of the drive into logical drives. These values assume that one physical hard disk is treated as one logical drive.

The above table is valid for single-sided disks. TRSDOS supports double-sided operation if the hardware interfacing the physical drives to the CPU allows it. A two-headed drive functions as a single logical drive, with the second side as a cylinder-for-cylinder extension of the first side. A bit in the Drive Code Table (DCT + 4, Bit 5) indicates one-sided or two-sided drive configuration.

A Winchester-type hard disk can be divided by heads into multiple logical drives. Details are supplied with Radio Shack drives.

The Granule Allocation Table is the first relative sector of the directory cylinder. The following information describes the layout and contents of the GAT.

GAT + X'00' through GAT + X'5F'

Contains the free/assigned table information. GAT + 0 corresponds to cylinder 0, GAT + 1 corresponds to cylinder 1, GAT + 2 corresponds to cylinder 2, and so on. As noted above, bit 0 of each byte corresponds to the first granule on the cylinder, bit 1 to the second granule, and so on. A value of "1" indicates the granule is not available for use.

GAT + X'60' through GAT + X'BF'

Contains the available/locked out table information. It corresponds cylinder for cylinder in the same way as the free/assigned table. It is used during mirror-image backups to determine if the destination diskette has the proper capacity to effect a backup of the source diskette. This table does not exist for hard disks; for this reason, mirror-image backups cannot be performed on hard disk.

GAT + X'C0' through GAT + X'CA'

Used in hard drive configurations; extends the free/assigned table from X'00' through X'CA'. Hard drive capacity up to 203 (0-202) logical or 406 physical cylinders is supported.

GAT + X'CB'

Contains the operating system version that was used in formatting the disk. For example, disks formatted under TRSDOS 6.2 have a value of X'62' contained in this byte. It is used to determine whether or not the disk contains all of the parameters needed for TRSDOS operation.

GAT + X'CC'

Contains the number of cylinders in excess of 35. It is used to minimize the time required to compute the highest numbered cylinder formatted on the disk. It is excess 35 to provide compatibility with alien systems not maintaining this byte. If you have a disk that was formatted on an alien system for other than 35 cylinders, this byte can be automatically configured by using the REPAIR utility. (See the section on the REPAIR utility in the *Disk System Owner's Manual*.)

GAT + X'CD'

Contains data about the formatting of the disk.

Bit 7 — If set to "1," the disk is a data disk. If "0," the disk is a system disk.

Bit 6 — If set to "1," indicates double-density formatting. If "0," indicates single-density formatting.

Bit 5 — If set to "1," indicates 2-sided disk. If "0," indicates 1-sided disk.

Bits 3-4 — Reserved.

Bits 0-2 — Contain the number of granules per cylinder minus 1.

GAT + X'CE' and GAT + X'CF'

Contain the 16-bit hash code of the disk master password. The code is stored in standard low-order, high-order format.

GAT + X'D0' through GAT + X'D7'

Contain the disk name. This is the name displayed during a FREE or DIR operation. The disk name is assigned during formatting or during an ATTRIB disk renaming operation. The name is left justified and padded with blanks.

GAT + X'D8' through GAT + X'DF'

Contain the date that the diskette was formatted or the date that it was used as the destination in a mirror image backup operation in the format mm/dd/yy.

GAT + X'E0' through GAT + X'FF'

Reserved for system use.

In Version 6.2:

GAT + X'E0' through GAT + X'F4'

Reserved for system use.

GAT + X'F5' through GAT + X'FF'

Contain the Media Data Block (MDB).

GAT + X'F5' through GAT + X'F8' — the identifying header. These four bytes contain a 3 (X'03'), followed by the letters LSI (X'4C', X'53', X'49').

GAT + X'F8' through GAT + X'FF' — the last seven bytes of the DCT in use when the media was formatted. FORMAT, MemDISK, and TRSFORM6 install this information. See Drive Control Table (DCT) for more information on these bytes.

Hash Index Table (HIT)

The Hash Index Table is the key to addressing any file in the directory. It pinpoints the location of a file's directory with a minimum of disk accesses, keeping overhead low and providing rapid file access.

The system's procedure is to construct an 11-byte filename/extension field. The filename is left-justified and padded with blanks. The file extension is then inserted and padded with blanks; it occupies the three least significant bytes of

the 11-byte field. This field is processed through a hashing algorithm which produces a single byte value in the range X'01' through X'FF. (A hash value of X'00' indicates a spare HIT position.)

The system then stores the hash code in the Hash Index Table (HIT) at a position corresponding to the directory record that contains the file's directory. Since more than one 11-byte string can hash to identical codes, the opportunity for "collisions" exists. For this reason, the search algorithm scans the HIT for a matching code entry, reads the directory record corresponding to the matching HIT position, and compares the filename/extension stored in the directory with that provided in the file specification. If both match, the directory has been found. If the two fields do not match, the HIT entry was a collision and the algorithm continues its search from the next HIT entry.

The position of the HIT entry in the hash table is called the Directory Entry Code (DEC) of the file. All files have at least one DEC. Files that are extended beyond four extents have a DEC for each extended directory entry and use more than one filename slot. To maximize the number of file slots available, you should keep your files below five extents where possible.

Each HIT entry is mapped to the directory sectors by the DEC's position in the HIT. Think of the HIT as eight rows of 32-byte fields. Each row is mapped to one of the directory records in a directory sector: The first HIT row is mapped to the first directory record, the second HIT row to the second directory record, and so on. Each column of the HIT field (0-31) is mapped to a directory sector. The first column is mapped to the first directory sector in the directory cylinder (not including the GAT and HIT). Therefore, the first column corresponds to sector 2, the second column to sector 3, and so on. The maximum number of HIT columns used depends on the disk formatting according to the formula: $N = \text{number of sectors per cylinder minus two, up to 32}$.

The following chart shows the correlation of the Hash Index Table to the directory records. Each byte value shown represents the position in the HIT. This position value is the DEC. The actual contents of each byte is either a X(00) indicating a spare slot, or the 1-byte hash code of the file that occupies the corresponding directory record.

	Columns															
Row 1	00 10	01 11	02 12	03 13	04 14	05 15	06 16	07 17	08 18	09 19	0A 1A	0B 1B	0C 1C	0D 1D	0E 1E	0F 1F
Row 2	20 30	21 31	22 32	23 33	24 34	25 35	26 36	27 37	28 38	29 39	2A 3A	2B 3B	2C 3C	2D 3D	2E 3E	2F 3F
Row 3	40 50	41 51	42 52	43 53	44 54	45 55	46 56	47 57	48 58	49 59	4A 5A	4B 5B	4C 5C	4D 5D	4E 5E	4F 5F
Row 4	60 70	61 71	62 72	63 73	64 74	65 75	66 76	67 77	68 78	69 79	6A 7A	6B 7B	6C 7C	6D 7D	6E 7E	6F 7F
Row 5	80 90	81 91	82 92	83 93	84 94	85 95	86 96	87 97	88 98	89 99	8A 9A	8B 9B	8C 9C	8D 9D	8E 9E	8F 9F
Row 6	A0 B0	A1 B1	A2 B2	A3 B3	A4 B4	A5 B5	A6 B6	A7 B7	A8 B8	A9 B9	AA BA	AB BB	AC BC	AD BD	AE BE	AF BF
Row 7	C0 D0	C1 D1	C2 D2	C3 D3	C4 D4	C5 D5	C6 D6	C7 D7	C8 D8	C9 D9	CA DA	CB DB	CC DC	CD DD	CE DE	CF DF
Row 8	E0 F0	E1 F1	E2 F2	E3 F3	E4 F4	E5 F5	E6 F6	E7 F7	E8 F8	E9 F9	EA FA	EB FB	EC FC	ED FD	EE FE	EF FF

A 5¼" single density disk has 10 sectors per cylinder, two of which are reserved for the GAT and HIT. Since only eight directory sectors are possible, only the first eight positions of each HIT row are used. Other formats use more columns of the HIT, depending on the number of sectors per cylinder in the formatting scheme.

The eight directory records for sector 2 of the directory cylinder correspond to assignments in HIT positions 00, 20, 40, 60, 80, A0, C0, and E0. On system

disks, the following positions are reserved for system overlays. On data disks, these positions (except for 00 and 01) are available to the user.

00 — BOOT/SYS	20 — SYS6/SYS
01 — DIR/SYS	21 — SYS7/SYS
02 — SYS0/SYS	22 — SYS8/SYS
03 — SYS1/SYS	23 — SYS9/SYS
04 — SYS2/SYS	24 — SYS10/SYS
05 — SYS3/SYS	25 — SYS11/SYS
06 — SYS4/SYS	26 — SYS12/SYS
07 — SYS5/SYS	27 — SYS13/SYS

These entry positions correspond to the first two rows of each directory sector for the first eight directory sectors. Since the operating system accesses these overlays by position in the HIT rather than by filename, these positions are reserved on system disks.

The design of the Hash Index Table limits the number of files on any one drive to a maximum of 256.

Locating a Directory Record

Because of the coding scheme used on the entries in the HIT table, you can locate a directory record with only a few instructions. The instructions are:

```
AND 1FH
ADD A,2
```

(calculates the sector)

and

```
AND 0E0H
```

(calculates the offset in that sector)

For example, if you have a Directory Entry Code (DEC) of X'84', the following occurs when these instructions are performed:

```
AND 1FH
ADD A,2
```

Value of accumulator
A = X'84'

A = X'04'

A = X'06'

The record is in the seventh
sector of the directory cylinder
(0-6)

Using the Directory Entry Code (DEC) again, you can find the offset into the sector that was found using the above instructions by executing one instruction:

```
AND 0E0H
```

Value of accumulator
A = X'84'

A = X'80'

The directory record is X'80' (128)
bytes from the beginning of
the sector

If the record containing the sector is loaded on a 256-byte boundary (LSB of the address is X'00') and HL points to the starting address of the sector, then you can use the above value to calculate the actual address of the directory record by executing the instruction:

```
LD L,A
```

When executed after the calculation of the offset, this causes HL to point to the record. For example:

A = X'80'

```
LD    HL ,4200H ;Where sector is loaded
LD    L ,A      ;Replace LSB with offset
```

HL now contains 4280H, which is the address of the directory record you wanted.

If you cannot place the sector on a 256-byte boundary, then you can use the following instructions:

A = X'80'

```
LD    HL ,4256H ;Where sector is loaded
LD    E ,A      ;Put offset in E (LSB)

LD    D ,0      ;Put a zero in D (MSB)
ADD   HL ,DE    ;Add two values together
```

HL now contains 42D6H, which is the address of the directory record.

Note that the first DEC found with a matching hash code may be the file's extended directory entry (FXDE). Therefore, if you are going to write system code to deal with this directory scheme, you must properly deal with the FPDE/FXDE entries. See Directory Records for more information.



File Control Block (FCB)

The File Control Block (FCB) is a 32-byte memory area. Before the file is opened, this space holds the file's filespec. After an @OPEN or @INIT supervisor call is performed, the system uses this area to interface with the file, and replaces the filespec with other information. When the file is closed, the filespec (without any specified password) is returned to the FCB.

While a file is open, the contents of the FCB are dynamic. As records are written to or read from the disk file, specific fields in the FCB are modified. Avoid changing the contents of the FCB during the time a file is open, unless you are sure that the change will not affect the integrity of the file.

During most system access of the FCB, the IX index register is used to reference each field of data. Register pair DE is used mainly for the initial reference to the FCB address. The information contained in each field of the FCB is as follows:

FCB + 0

Contains the TYPE code of the control block.

Bit 7 — If set to "1," indicates that the file is in an open condition; if "0," the file is assumed closed. This bit can be tested to determine the "open" or "closed" status of an FCB.

Bit 6 — Is set to "1" if the file was opened with UPDATE access or higher.

Bit 5 — Indicates a Partition Data Set (PDS) type file.

Bits 4-3 — Reserved for future use.

Bit 2 — Is set to "1" if the system performed any WRITE operation on this file. It is used to update the MOD flag in the directory record when the file is closed.

Bits 1-0 — Reserved for future use.

FCB + 1

Contains status flag bits used in read/write operations by the system.

Bit 7 — If set to "1," indicates that I/O operations will be either full sector operations or byte operations of logical record length (LRL) less than 256. If "0," only sector operations will be performed. If you are going to use only full-sector I/O, you can reduce system overhead by specifying the LRL at open time as 0 (indicating 256). An LRL of other than 256 sets bit 7 to "1" on open.

Bit 6 — If set to "1," indicates that the end of file (EOF) is to be set to ending record number (ERN) only if next record number (NRN) exceeds the current value of EOF. This is the case if random access is to be used. During random access, the EOF is not disturbed unless you extend the file beyond the last record slot. Any time the position routine (@POSN) is called, bit 6 is automatically set. If bit 6 is "0," then EOF will be updated on every WRITE operation.

Bit 5 — If "0," then the disk I/O buffer contains the current sector denoted by NRN. If set to "1," then the buffer does not contain the current sector. During byte I/O, bit 5 is set when the last byte of the sector is read. A sector read resets the bit, showing the buffer to be current.

Bit 4 — If set to "1," indicates that the buffer contents have been changed since the buffer was read from the file. It is used by the system to determine whether the buffer must be written back to the file before reading another record. If "0," then the buffer contents were not changed.

Bit 3 — Used to specify that the directory record is to be updated each time the NRN exceeds the EOF. (The normal operation is to update the directory only when an FCB is closed.) Some unattended operations may use this extra measure of file protection. It is specified by adding an exclamation mark ("!") to the end of a filespec when the filespec is requested at open time.

Bits 2-0 — Contain the user (access) protection level as retrieved from the directory of the file. The 3-bit binary value is one of the following:

0 = FULL	2 = RENAME	4 = UPDATE	6 = EXECUTE
1 = REMOVE	3 = WRITE	5 = READ	7 = NO ACCESS

FCB + 2

Used by Partition Data Set (PDS) files.

FCB + 3 and FCB + 4

Contain the buffer address in low-order, high-order format. This is the buffer address specified in register pair HL when the @INIT or @OPEN SVC is performed.

FCB + 5

Contains the relative byte offset within the current buffer for the next I/O operation. If this byte has a zero value, then FCB + 1, Bit 5 must be examined to see if the first byte in the current buffer is the target position or if it is the first byte of the next record. If you are performing sector I/O of byte data (that is, maintaining your own buffering), then it is important to maintain this byte when you close the file if the true end of file is not at a sector boundary.

FCB + 6

Bits 3-7 — Reserved for system use.

Bits 0-2 — Contain the logical drive number in binary of the drive containing the file. Do not modify this byte; altering this value may damage other files. This byte and FCB + 7 are the only links to the file's directory information.

FCB + 7

Contains the directory entry code (DEC) for the file. This code is the offset in the Hash Index Table where the hash code for the file appears. Do not modify this byte; altering this value may damage other files. This byte and FCB + 6 are the only links to the directory information for the file.

FCB + 8

Contains the end-of-file byte offset. This byte is similar to FCB + 5 except that it pertains to the end of file rather than to the next record number.

FCB + 9

Contains the logical record length that was in effect when the file was opened. This may not be the same LRL that exists in the directory. The directory LRL is generated at the file creation and never changes unless the file is overwritten.

FCB + 10 and FCB + 11

Contain the next record number (NRN), which is a pointer for the next I/O operation. When a file is opened, NRN is zero, indicating a pointer to the beginning. Each sequential sector I/O advances NRN by one.

FCB + 12 and FCB + 13

Contain the ending record number (ERN) of the file. This is a pointer to the sector that contains the end-of-file indicator. In a null file (one with no records), ERN equals 0. If one sector has been written, ERN equals 1.

FCB + 14 and FCB + 15

Contain the same information as the first extent of the directory. This represents the starting cylinder of the file (FCB + 14) and the starting relative granule within the starting cylinder (FCB + 15). FCB + 15 also contains the number of contiguous granules allocated in the extent. These bytes are used as a pointer to the beginning of the file referenced by the FCB.

FCB + 16 through FCB + 19

This 4-byte entry contains granule allocation information for an extent of the file. Relative bytes 0 and 1 contain the total number of granules allocated to the file up to but not including the extent referenced by this field. Relative byte 2 contains the starting cylinder of this extent. Relative byte 3 contains the starting relative granule for the extent and the number of contiguous granules.

FCB + 20 through FCB + 23

Contain information similar to the above but for a second extent of the file.

FCB + 24 through FCB + 27

Contain information similar to the above but for a third extent of the file.

FCB + 28 through FCB + 31

Contain information similar to the above but for a fourth extent of the file.

The file control block contains information on only four extents at one time. If the file has more than four extents, additional directory accessing is done to shift the 4-byte entries in order to make space for the new extent information.

Although the system can handle a file of any number of extents, you should keep the number of extents small. The most efficient file is one with a single extent. The number of extents can be reduced by copying the file to a disk that contains a large amount of free space.



7/TRSDOS Version 6 Programming Guidelines

Converting to TRSDOS Version 6

This section provides suggestions on writing programs effectively with TRSDOS Version 6, and on converting programs created with TRSDOS 1.3 and LDOS 5.1 operating systems for use with TRSDOS Version 6. This information is by no means complete, but presents some important concepts to keep in mind when using TRSDOS Version 6.

When programming in assembly language, you can use TRSDOS Version 6 routines for commonly used operations. These are accessed through the supervisor calls (SVCs) instead of absolute call addresses. Nothing in the system can be accessed via any absolute address reference (except Z-80 RST and NMI jump vectors).

IMPORTANT NOTE: TRSDOS provides all functions and storage through supervisor calls. No address or entry point below 3000H is documented or supported by Radio Shack.

The keyboard is not accessible via "peeking," and the video RAM cannot be "poked." The keyboard and video are accessible only through the appropriate SVCs.

Another distinction is that TRSDOS Version 6 handling of logical byte I/O devices (keyboard, video, printer, communications line) completely supports error status feedback. A FLAG convention is uniform throughout these device drivers as well as physical byte I/O associated with files. The device handling in TRSDOS Version 6 is completely independent. That means that byte I/O, both logical and physical, can be routed, filtered, and linked. Therefore, it is important to test status return codes in all applications using byte I/O regardless of the device that the application expects to be used, since re-direction to some other device is possible at the TRSDOS level. Appropriate action must be taken when errors are detected.

Modules loaded into memory and protected by lowering HIGH\$ must include the standard header, as described earlier under "Memory Header." The @GTMOD supervisor call requires that this header be present in every resident module for proper operation.

The file password protection terms of UPDATE and ACCESS have been changed in TRSDOS Version 6 to OWNER and USER, respectively. The additional file protection level of UPDATE has been added. A file with UPDATE protection level can be read or written to, but its end of file cannot be extended. This protection can be useful in a random access fixed-size file or in a file where shared access is to take place.

Files opened with UPDATE or greater access are indicated as open in their directory. Attempting to open the file again forces a change to READ access protection and a "File already open" error code. It is therefore important for applications to CLOSE files that are opened.

For the convenience of applications that access files only for reading, you can inhibit the "file open bit." If you set bit 0 of the system flag SFLAG\$ (see the @FLAGS supervisor call), the file open bit is not set in the file's directory. Once set, the next @OPEN or @INIT SVC automatically resets bit 0 of SFLAG\$. Note that you cannot use this procedure for files being written to, since it inhibits the CLOSE process.

Some application programs need access to certain system parameters and variables. A number of flags, variables, and port images can be accessed relative to a flag pointer obtained via the @FLAGS supervisor call. These parameters are only accessible relative to this pointer, as the pointer's location may change. (See the explanation of the @FLAGS SVC.)

All applications must honor the contents of HIGH\$. This pointer contains the highest RAM address usable by any program. You can retrieve and change HIGH\$ by using the @HIGH\$ SVC.

TRSDOS Version 6 library commands and utilities supply a return code (RC) at completion. The RC is returned in register pair HL. The value returned is either zero (indicating no error), a number from one through 62 (indicating an error as noted in Appendix A, TRSDOS Error Messages), or X'FFFF' (indicating an extended error which is currently not assigned an error number). TRSDOS Version 6 Job Control Language (JCL) aborts on any program terminating with a non-zero RC value. Applications should therefore properly set the return code register pair HL before exiting.

TRSDOS Version 6 library commands are also invokable via the @CMNDR SVC which executes the command. Library commands properly maintain the Stack Pointer (SP) and exit via a RET instruction. In this manner, control is returned to the invoking program with the RC present for testing. For commands invoked with the @CMNDI SVC or prompted for via the @EXIT SVC, the SP is restored to the system stack. The top of the stack will contain an address suitable for simulating an @EXIT SVC; thus, if your application program properly maintains the integrity of the stack pointer, it can exit after setting the RC via a RET instruction instead of an @EXIT SVC.

TRSDOS Version 6 diskette and file structure is identical to that used in LDOS 5.1. This includes formatting, directory structure, and data address mark conventions. TRSDOS Version 6 system diskettes, however, use the entire BOOT track (track 0). This compatibility means that data files may be used interchangeably between LDOS 5.1 equipped machines and TRSDOS Version 6 equipped machines; the diskettes themselves are readable and writable across both operating systems.

The methods of internal handling of device linking and filtering have been changed from LDOS 5.1. (It is beyond the scope of this manual to explain the internal functioning of TRSDOS Version 6.) Device filters must adhere to a strict protocol of linkage in order to function properly. See the section on "Device Driver and Filter Templates" for information on device driver and filter protocol.

Stack Handling Restrictions*

Interrupt tasks and filters that deal with the keyboard or video must not place the stack pointer above X'F3FF'. This is because any operation that requires the keyboard or video RAM switches in the 3K bank at X'F400' and suppresses the stack until it is switched out again. If the system accesses the stack at any time during this period, the integrity of the stack is destroyed.

*In TRSDOS 6.0.0, the stack cannot be placed above X'F3FF' for any reason.

Programming With Restart Vectors

The Restart instruction (RST) provides the assembly language programmer with the ability to call a subroutine with a one-byte call. If a routine is called many times by a program, the amount of space that is saved by using the RST instruction (instead of a three-byte CALL) can be significant.

In TRSDOS a RST instruction is also used to interface to the operating system. The system uses RST 28H for supervisor calls. RSTs 00H, 30H, and 38H are for the system's internal use.

RSTs 08H, 10H, 18H, and 20H are available for your use. Caution: Some programs, such as BASIC, may use some of these RSTs.

Each RST instruction calls the address given in the operand field of the instruction. For example, RST 18H causes the system to push the current program counter address onto the stack and then set the program counter to address 0018H. RST 20H causes a jump to location 0020H, and so on.

Each RST has three bytes reserved for the subroutine to use. If the subroutine will not fit in three bytes, then you should code a jump instruction (JP) to where the subroutine is located. At the end of the subroutine, code a return instruction (RET). Control is then transferred to the instruction that follows the RST.

For example, suppose you want to use RST 18H to call a subroutine named "ROUTINE." The following routine loads the restart vector with a jump instruction and saves the old contents of the restart vector for later use.

```
SETRST: LD      IX,0018H    ;Restart area address
        LD      IY,RDATA    ;Data area address
        LD      B,3         ;Number of bytes to move
LOOP:   LD      A,(IX)       ;Read a byte from
                                ;restart area
        LD      C,(IY)       ;Read a byte from data
                                ;area
        LD      (IX),C       ;Store this byte in
                                ;restart area
        LD      (IY),A       ;Store this byte in data
                                ;area
        INC     IX           ;Increment restart area
                                ;pointer
        INC     IY           ;Increment data area
                                ;pointer
        DJNZ    LOOP        ;Loop till 3 bytes moved
        RET                ;Return when done
RDATA:  DEFB     0C3H        ;Jump instruction (JP)
        DEFW    ROUTINE     ;Operand (name of
                                ;subroutine)
```

Before exiting the program, calling the above routine again puts the original contents of the restart vector back in place.

KFLAG\$ (BREAK), (PAUSE), and (ENTER) Interfacing

KFLAG\$ contains three bits associated with the keyboard functions of BREAK, PAUSE (SHIFT @), and ENTER. A task processor interrupt routine (called the KFLAG\$ scanner) examines the physical keyboard and sets the appropriate KFLAG\$ bit if any of the conditions are observed. Similarly, the RS-232C driver routine also sets the KFLAG\$ bits if it detects the matching conditions being received.

Many applications need to detect a PAUSE or BREAK while they are running. BASIC checks for these conditions after each logical statement is executed (that is, at the end of a line or at a ":"). That is how, in BASIC, you can stop a program with the **(BREAK)** key or pause a listing.

One method of detecting the condition in previous TRSDOS operating systems was to issue the @KBD supervisor call to check for BREAK or PAUSE (**(SHIFT)@**), ignoring all other keys. Unfortunately, this caused keyboard type-ahead to be ineffective; the @KBD SVC flushed out the type-ahead buffer if any other keystrokes were stacked up.

Another method was to scan the keyboard, physically examining the keyboard matrix. An undesirable side effect of this method was that type-ahead stored up the keyboard depression for some future unexpected input request. Examining the keyboard directly also inhibits remote terminals from passing the BREAK or PAUSE condition.

In TRSDOS Version 6, the KFLAG\$ scanner examines the keyboard for the BREAK, PAUSE, and ENTER functions. If any of these conditions are detected, appropriate bits in the KFLAG\$ are set (bits 0, 1, and 2 respectively).

Note that the KFLAG\$ scanner only sets the bits. It does not reset them because the "events" would occur too fast for your program to detect. Think of the KFLAG\$ bits as a latch. Once a condition is detected (latched), it remains latched until something examines the latch and resets it—a function to be performed by your KFLAG\$ detection routine.

Under Version 6.2, you can use the @CKBRKC SVC, SVC 106, to see if the BREAK key has been pressed. If a BREAK condition exists, @CKBRKC resets the break bit of KFLAG\$.

For illustration, the following example routine uses the BREAK and PAUSE conditions:

```

KFLAG$ EQU 10
@FLAGS EQU 101
@KBD EQU 8
@KEY EQU 1
@PAUSE EQU 16
CKPAWS LD A,@FLAGS ;Get Flags pointer
RST 28H ;into register IY
LD A,(IY+KFLAG$) ;Get the KFLAG$
RRCA ;Bit 0 to carry
JP C,GOTBRK ;Go on BREAK
RRCA ;Bit 1 to carry
RET NC ;Return if no Pause
CALL RESKFL ;Reset the flag
PUSH DE
FLUSH LD A,@KBD ;Flush type-ahead
RST 28H ;buffer while
JR Z,FLUSH ;ignoring errors
POP DE
PROMPT PUSH DE
LD A,@KEY ;Wait on key entry
RST 28H
POP DE
CP 80H ;Abort on (BREAK)
JP Z,GOTBRK
CP 60H ;Ignore PAUSE!
JR Z,PROMPT ;else . . .
RESKFL PUSH HL ;reset KFLAG$
PUSH AF
LD A,@FLAGS ;Get flags pointer
RST 28H ;into register IY
RESKFL1 LD A,(IY+KFLAG$) ;Get the flag
AND 0F8H ;Strip ENTER,

```

```

LD      (IY+KFLAG$),A ;PAUSE, BREAK
PUSH    BC
LD      B,16
LD      A,@PAUSE      ;Pause a while
RST     28H
POP     BC
LD      A,(IY+KFLAG$) ;Check if finger is
AND     3              ;still on key
JR      NZ,RESKFL1    ;Reset it again
POP     AF             ;Restore registers
POP     HL             ;and exit
RET

```

The best way to explain this KFLAG\$ detection routine is to take it apart and discuss each subroutine. The first piece reads the KFLAG\$ contents:

```

KFLAG$  EQU    10
CKPAWS  LD      A,@FLAGS      ;Get Flags pointer
RST     28H                  ;into register IY
LD      A,(IY+KFLAG$) ;Get the KFLAG$
RRCA                    ;Bit 0 to carry
JP      C,GOTBRK          ;Go on BREAK
RRCA                    ;Bit 1 to carry
RET     NC                ;Return if no pause

```

The @FLAGS SVC obtains the flags pointer from TRSDOS. Note that if your application uses the IY index register, you should save and restore it within the CKPAWS routine. (Alternatively, you could use @FLAGS to calculate the location of KFLAG\$, use register HL instead of IY, and place the address into the LD instructions of CKPAWS at the beginning of your application.)

The first rotate instruction places the BREAK bit into the carry flag. Thus, if a BREAK condition is in effect, the subroutine branches to "GOTBRK," which is your BREAK handling routine.

If there is no BREAK condition, the second rotate places what was originally in the PAUSE bit into the carry flag. If no PAUSE condition is in effect, the routine returns to the caller.

This sequence of code gives a higher priority to BREAK (that is, if both BREAK and PAUSE conditions are pending, the BREAK condition has precedence). Note that the GOTBRK routine needs to clear the KFLAG\$ bits after it services the BREAK condition. This is easily done via a call to RESKFL.

The next part of the routine is executed on a PAUSE condition:

```

CALL    RESKFL            ;Reset the flag
PUSH    DE
FLUSH   LD      A,@KBD     ;Flush type-ahead
RST     28H              ;buffer while
JR      Z,FLUSH          ;ignoring errors
POP     DE

```

First the KFLAG\$ bits are reset via the call to RESKFL. Next, the routine takes care of the possibility that type-ahead is active. If it is, the PAUSE key was probably detected by the type-ahead routine and so is stacked in the type-ahead buffer also. To flush out (remove all stored characters from) the type-ahead buffer, @KBD is called until no characters remain (an NZ is returned).

Now that a PAUSEd state exists and the type-ahead buffer is cleared, the routine waits for a key input:

```

PROMPT  PUSH    DE
LD      A,@KEY           ;Wait on key entry
RST     28H
POP     DE
CP      80H              ;Abort on (BREAK)
JP      Z,GOTBRK

```

```

CP      60H      ;Ignore PAUSE;
JR      Z,PROMPT ;else . . .

```

The PROMPT routine accepts a BREAK and branches to your BREAK handling routine. It ignores repeated PAUSE (the 60H). Any other character causes it to fall through to the following routine which clears the KFLAG\$:

```

RESKFL  PUSH  HL      ;reset KFLAG$
        PUSH  AF
        LD    A,@FLAGS ;Get flags pointer
        RST   28H      ;into register IY
RESKFL1 LD    A,(IY+KFLAG$) ;Get the flag
        AND   0FBH      ;Strip ENTER,
        LD    (IY+KFLAG$),A ;PAUSE, BREAK
        PUSH  BC
        LD    B,16
        LD    A,@PAUSE   ;Pause a while
        RST   28H
        POP   BC
        LD    A,(IY+KFLAG$) ;Check if finger is
        AND   3          ;still on key
        JR    NZ,RESKFL1 ;Reset it again
        POP   AF          ;Restore registers
        POP   HL          ;and exit
        RET

```

The RESKFL subroutine should be called when you first enter your application. This is necessary to clear the flag bits that were probably in a "set" condition. This "primes" the detection. The routine should also be called once a BREAK, PAUSE, or ENTER condition is detected and handled. (You need to deal with the flag bits for only the conditions you are using.)

Interfacing to @ICNFG

With the TRSDOS library command SYSGEN, many users may wish to SYSGEN the RS-232C driver. Before doing that, the RS-232C hardware (UART, Baud Rate Generator, etc.) must be initialized. Simply using the SYSGEN command with the RS-232C driver resident is not enough; some initialization routine is necessary. The @ICNFG (Initialization CoNFiGuration) vector is included in TRSDOS to provide a way to invoke a routine to initialize the RS-232C driver when the system is booted. It also provides a way to initialize the hard disk controller at power-up (required by the Radio Shack hard disk system).

The final stages of the booting process loads the configuration file CONFIG/SYS if it exists. After the configuration file is loaded, an initialization subroutine CALLs the @ICNFG vector. Thus, any initialization routine that is part of a memory configuration can be invoked by chaining into @ICNFG.

If you need to configure your own routine that requires initialization at power-up, you can chain into @ICNFG. The following procedure illustrates this link. The first thing to do is to move the contents of the @ICNFG vector into your initialization routine:

```

LD      A,@FLAGS      ;Get flags pointer
RST     28H            ;into register IY
LD      A,(IY+28)      ;Get opcode
LD      (LINK),A
LD      L,(IY+29)      ;Get address LOW
LD      H,(IY+30)      ;Get address HIGH
LD      (LINK+1),HL

```

This subroutine does this by transferring the 3-byte vector to your routine. You then need to relocate your routine to its execution memory address. Once this

is done, transfer the relocated initialization entry point to the @ICNFG vector as a jump instruction:

```
LD    HL,INIT        ;Get (relocated)
LD    (IY+29),L      ;init address
LD    (IY+30),H
LD    A,0C3H         ;Set JP instruction
LD    (IY+28),A
```

If you need to invoke the initialization routine at this point, then you can use:

```
CALL ROUTINE        ;Invoke your routine
```

Your initialization routine would be unique to the function it was to perform, but an overall design would look like this:

```
INIT    CALL ROUTINE    ;Start of init
LINK    DEFS 3          ;Continue on
ROUTINE .
        your initialization routine

RET
```

After linking in your routine, perform the SYSGEN. If you have followed these procedures, your routine will be invoked every time you start up TRSDOS.

Interfacing to @KITSK

Background tasks can be invoked in one of two ways. For tasks that do not require disk I/O, you can use the RTC (Real Time Clock) interrupt and one of the 12 task slots (or other external interrupt). For tasks that require disk I/O, you can use the keyboard task process.

At the beginning of the TRSDOS keyboard driver is a call to @KITSK. This means that any time that @KBD is called, the @KITSK vector is also called. (The type-ahead task, however, bypasses this entry so that @KITSK is not called from the type-ahead routine.) Therefore, if you want to interface a background routine that does disk I/O, you must chain into @KITSK.

The interfacing procedure to @KITSK is identical to that shown in the section "Interfacing to @ICNFG," except that IY+31 through IY+33 is used to reference the @KITSK vector. You may want to start your background routine with:

```
START    CALL ROUTINE    ;Invoke task
LINK     DEFS 3          ;For @KITSK hook
ROUTINE  EQU $           ;Start of the task
```

Be aware of one major pitfall. The @KBD routine is invoked from @CMNDI and @CMNDR (which is in SYS1/SYS). This invocation is from the @KEYIN call, which fetches the next command line after issuing the "TRSDOS Ready" message. If your background task executes and opens or closes a file (or does anything to cause the execution of a system overlay other than SYS1), then SYS1 is overwritten by SYS2 or SYS3. When your routine finishes, the @KEYIN handler tries to return to what called it—SYS1, which is no longer resident. Therefore, any task chained to @KITSK which causes a resident SYS1 to be overwritten must reload SYS1 before returning.

You can use the following code to reload SYS1 if SYS1 was resident prior to your task's execution:

```
ROUTINE LD    A,@FLAGS    ;Get flags pointer
        RST   28H         ;into register IY
        LD    A,(IY-1)    ;Get resident over-
        AND   8FH         ;lay and remove
        LD    (OLDSYS+1),A ;the entry code
        .
```

```

                                rest of your task
                                .
EXIT      EQU      $
OLDSYS    LD        A,0          ;Get old overlay #
          CP        83H         ;Was it SYS1?
          RET       NZ          ;Return if not; else
          RST       28H         ;Get SYS1 per reg. A
                                ;(no RET needed)

```

Interfacing to the Task Processor

This section explains how to integrate interrupt tasks into your applications.

One of the hardware interrupts in the TRS-80 is the real time clock (RTC). The RTC is synchronized to the AC line frequency and pulses at 60 pulses per second, or once every 16.67 milliseconds. (Computers operating with 50 Hz AC use a 50 pulses per second RTC interrupt. In this case, all time relationships discussed in this section should be adjusted to the 50 Hz base.)

A software task processor manages the RTC interrupt in performing background tasks necessary to specific functions of TRSDOS (such as the time clock, blinking cursor, and so on). The task processor allows up to 12 individual tasks to be performed on a "time-sharing" basis.

These tasks are assigned to "task slots" numbered from 0 to 11. Slots 0-7 are considered "low priority" tasks (executing every 266.67 milliseconds). Slots 8-10 are medium priority tasks (executing every 33.33 milliseconds). Slot 11 is a high priority task (executing every 16.66 milliseconds SYSTEM (FAST) or 33.33 milliseconds SYSTEM (SLOW)). Task slots 3, 7, 9, and 10 are reserved by the system for the ALIVE, TRACE, SPOOL, and TYPE-AHEAD functions, respectively.

TRSDOS maintains a Task Control Block Vector Table (TCBVT) which contains 12 vectors, one for each of the 12 task slots. TRSDOS contains five supervisor calls that manage the task vectors. The five SVCs and their functions are:

@CKTSK	Checks to see whether a task slot is unused or active
@ADTSK	Adds a task to the TCBVT
@RMTSK	Removes a task from the TCBVT
@KLTSK	Removes the currently executing task
@RPTSK	Replaces the TCB address for the current task

The TRSDOS Task Control Block Vector Table contains vector pointers. Each TCBVT vector points to an address in memory, which in turn contains the address of the task. Thus, the tasks themselves are indirectly addressed.

When you are programming a task to be called by the task processor, the entry point of the routine needs to be stored in memory. If you make this storage location the beginning of a Task Control Block (TCB), the reason for indirect vectoring of interrupt tasks will become more clear. Consider an example TCB:

```

MYTCB     DEFW     MYTASK
COUNTER   DEFB     15
TEMPY     DEFS     1
MYTASK    RET

```

This is a useless task, since the only thing it does is return from the interrupt. However, note that a TCB location has been defined as "MYTCB" and that this location contains the address of the task. A few more data bytes immediately following the task address storage have also been defined.

Upon entry to a service routine, index register IX contains the address of the TCB. You can therefore address any TCB data using index instructions. For example, you could use the instruction "DEC (IX+2)" to decrement the value contained in COUNTER in the above routine.

Here is the routine expanded slightly:

```

MYTCB   DEFW   MYTASK
COUNTER DEFB   15
TEMPY   DEFB   0
MYTASK  DEC    (IX+2)
        RET    NZ
        LD     (IX+2),15
        RET

```

This version makes use of the counter. Each time the task executes, the counter is decremented. When the count reaches zero, the counter is restored to its original value.

In order to be executed, all tasks must be added to the TCBVT. The @ADTSK supervisor call does this. For the above routine, assume the task slot chosen is low-priority slot 2. You can ascertain that slot 2 is available for use by using the @CKTSK SVC as follows:

```

LD      C,2           ;Reference slot 2
LD      A,28          ;Set for @CKTSK SVC
RST     28H           ;An "NZ" indication
JP      NZ,INUSE      ;says that the slot is
                     ;being used.

```

Once you determine that the slot is available (that is, not being used by some other task), you can add your task routine. The following code adds this task to the TCBVT:

```

LD      DE,MYTCB      ;Point to the TCB
LD      C,2           ;Reference slot 2
LD      A,29          ;Set for @ADTSK SVC
RST     28H           ;Issue the SVC

```

The above program lines point register DE to the TCB, load the task slot number into register C, and then issue the @ADTSK supervisor call. If you want this task to run regardless of what is in memory, you can place it in high memory (of bank 0) and protect it by moving HIGH\$ below it via the @HIGH\$ supervisor call.

Once a task has been activated, it is sometimes necessary to deactivate it. You can do this in two ways. The most common way is to use the @RMTSK supervisor call:

```

LD      C,2           ;Designate the task
                     ;slot
LD      A,30          ;Set for @RMTSK SVC
RST     28H           ;Issue the SVC

```

You identify the task slot to remove by placing a value in register C, and then you issue the supervisor call.

You can use another method if you want to remove the task while it is being executed. Examine the routine modified as follows:

```

MYTCB   DEFW   MYTASK
COUNTER DEFB   10
TEMPY   DEFB   0
MYTASK  DEC    (IX+2)
        RET    NZ
        LD     A,32          ;Set for @KLTSK SVC
        RST    28H          ;Issue the SVC

```

The @KLTSK supervisor call removes the currently executing task from the TCBVT. The system does not return to your routine, but continues as if you had executed a RET instruction. For this reason, the @KLTSK SVC should be the last instruction you want executed. In this example, MYTASK decrements the counter by one on each entry to the task. When the counter reaches zero, the task is removed from slot 2.

The last task processor supervisor call is @RPTSK. The @RPTSK function updates the TCB storage vector (the vector address in your Task Control Block) to be the address immediately following the @RPTSK SVC instruction. As with @KLTSK, the system does not return to your service routine after the SVC is made, but continues on with the task processor. The following example illustrates how @RPTSK can be used in a program:

```

@ADTSK      ORG      9000H
@RPTSK      EQU      31
@RMTSK      EQU      30
@EXIT       EQU      22
@VDCTL      EQU      15
BEGIN       LD        DE,TCB          ;Point to TCB
            LD        C,0             ;and add the task
            LD        A,@ADTSK        ;to slot 0
            RST        28H
            LD        A,@EXIT         ;Exit to TRSDOS
            RST        28H
TCB         DEFW      TASK
COUNTER     DEFB      15
TASKA       LD        A,@RPTSK        ;Replace current
            RST        28H            ;task with TASKA
TASK        LD        BC,027CH        ;Put a character
            LD         HL,004FH        ;at Row 0, Col. 79
            LD        A,@VDCTL
            RST        28H
            DEC        (IX+2)         ;Decrement the counter
            RET        NZ             ;and return if not
            LD        (IX+2),15       ;expired; else reset
            LD        A,@RPTSK        ;Replace the previous
            RST        28H            ;task with TASKB
TASKB       LD        BC,022DH        ;Put a character
            LD         HL,004FH        ;at Row 0, Col. 79
            LD        A,@VDCTL
            RST        28H
            DEC        (IX+2)
            RET        NZ
            LD        (IX+2),15
            JR         TASKA
END         BEGIN

```

This task routine contains no method of relocating it to protected RAM. The statements starting at the label BEGIN add the task to TCBVT slot 0 and return to TRSDOS Ready. The task contains a four-second down counter and a routine to put a character in video RAM (80th character of Row 0). At four-second intervals, the character toggles between '|' and '-'. This is done by using the @RPTSK SVC to toggle the execution of two separate routines which perform the character display.

TRSDOS uses bank-switched memory. In order to properly control and manage this additional memory, certain restrictions are placed on tasks. All tasks must be placed either in low memory (addresses X'0000' through X'7FFF') or in bank zero of high memory (addresses X'8000' through X'FFFF'). The task processor always enables bank zero when performing background tasks. The assembly language programmer must ensure that tasks are placed in the correct memory area.

Interfacing RAM Banks 1 and 2

The proper use of the RAM bank transfer techniques described here requires a high degree of skill in assembly language programming. This section on bank switching is intended for the professional.

The TRS-80 Model 4 can optionally support a second set of 64K RAM, bringing the total RAM to 128K. TRSDOS designates this extra 64K RAM as two banks of 32K RAM each, which are banks 1 and 2 of bank-switched RAM. The upper 32K of standard RAM is designated bank 0. At any one time, only one of the banks is resident. The resident bank is always addressed at X'8000' through X'FFFF'. When a bank transfer is performed, the specified bank becomes addressable and the previous bank is no longer available. Since memory refresh is performed on all banks at all times, nothing in the previously resident bank is altered during whatever time it is not addressable (that is, not resident).

You can access this additional RAM by means of the @BANK supervisor call (SVC 102). When you power up your computer or press reset, TRSDOS looks to see which banks of RAM are installed in your machine. TRSDOS maintains a bit map in one byte of storage, with each bit representing one of the banks of RAM. This byte is called "Bank Available RAM" (BAR), and its information is set when you boot TRSDOS. Bit 0 corresponds to bank 0, bit 1 corresponds to bank 1, and so on up to bit 7. From a hardware standpoint, the Model 4 has a maximum of three banks. You have either bank 0 only (a 64K machine), or banks 0-2 (a 128K machine).

Another bit map is used to indicate whether a bank is reserved or available for use. This byte is called the "Bank Used RAM" (BUR). Again, bit 0 corresponds to bank 0, bit 1 to bank 1, and so on. TRSDOS design supports the use of banks 1 and 2 primarily for data storage (for example, a spool buffer, Memdisk, etc.). The management of any memory space within a particular bank of RAM (excluding bank 0) is the responsibility of the application program "reserving" a particular bank.

TRSDOS requires that any device driver or filter that is relocated to high memory (X'8000' through X'FFFF') reside in bank 0. The TRSDOS device handler always invokes bank 0 upon execution of any byte I/O service request (@PUT, @GET, @CTL, as well as other byte I/O SVCs that use @PUT/@GET/@CTL). This ensures that any filter or driver attached to the device in question will be available. If a RAM bank other than 0 was resident, it is restored upon return from the device handler. This ensures that device I/O is never impacted by bank switching.

TRSDOS also requires that all interrupt tasks reside in bank 0 or low memory (X'0000' through X'7FFF'). The interrupt task processor always enables bank 0 and restores whatever bank was previously resident. An interrupt task may perform a bank transfer from 0 to another bank provided the necessary linkage and stack area is used. This is discussed in more detail later.

All bank transfer requests must be performed using the @BANK SVC. This SVC provides four functions, three of which are interrogatory and one of which performs the actual bank switching.

As mentioned previously, the contents of banks other than 0 are managed by the application, not by TRSDOS. Therefore, the application needs a way of finding out if any given bank is available. For example, if an application wants to reserve use of bank 1, it must first check to see if bank 1 is free to use. This is done by using function 2 as follows:

```
LD      C,1           ;Specify bank 1
LD      B,2           ;Check BUR if bank in use
LD      A,@BANK       ;Set @BANK SVC (102)
RST     2BH
JR      NZ,INUSE      ;NZ if bank already in use
```

Note that the return condition (NZ or Z) shows whether or not you can use the specified bank (it may not even be installed).

If the specified bank is available, you then need to reserve it. Do this by using function 3 as follows:

```
LD      C,1           ;Specify bank 1
LD      B,3           ;Set BUR to show "in use"
```

```
LD      A,@BANK      ;Set @BANK SVC (102)
RST     28H
JR      NZ,ERROR
```

You must check for an error by examining the Z flag. In general (discounting a system error), an NZ condition returned means that the specified bank is already in use. If you had performed a function 2 (testing to see if the bank was available) and got a not-in-use indication, but got an NZ condition on function 3, then the @BANK SVC routine has been altered and is probably unusable.

When an application no longer requires a memory bank, it can return the bank to a "free" state by using function 1 as follows:

```
LD      C,1          ;Specify bank 1
LD      B,1          ;Set BUR to show free
LD      A,@BANK      ;Set @BANK SVC (102)
RST     28H
```

No error condition is checked, as none is returned by TRSDOS. If you should mistakenly use function 1 with a bank that is nonexistent, an error is returned if you try to invoke the nonexistent bank.

To find out which bank is resident at any time, use function 4 as follows:

```
LD      B,4          ;Which bank is resident?
LD      A,@BANK      ;Set @BANK SVC (102)
RST     28H
```

The current bank number is returned in register A.

To exchange the current bank with the specified bank, use function 0. Since a memory transfer takes place in the address range X'8000' through X'FFFF', the transfer cannot proceed correctly if the stack pointer (SP) contains a value that places the stack in that range. @BANK inhibits function 0 and returns an SVC error if the stack pointer violates this condition.

A bank can be used purely as a data storage buffer. The application's routines for invoking and indexing the bank switching probably reside in the user range X'3000' through X'7FFF'. As an example, the following code invokes a previously tested and reserved bank (via functions 2 and 3), accesses the buffer, and then restores the previous bank:

```
LD      C,1          ;Specify bank 1
LD      B,0          ;Bring up bank
LD      A,@BANK      ;Set @BANK SVC (102)
RST     28H
JR      NZ,ERROR     ;Error trap
PUSH    BC           ;Save old bank data
;
; your code to access the buffer region
;
POP     BC           ;Recover old bank data
LD      A,@BANK      ;Set @BANK SVC (102)
RST     28H
JR      NZ,ERROR     ;Error trap
```

Note that the @BANK function 0 conveniently returns a zero in register B to effect a function 0 later, as well as provides the old bank number in register C. This means that you only have to save register pair BC, pop it when you want to restore the previous bank, and then issue the @BANK SVC.

Suppose you want to transfer to another bank from a routine that is executing in high memory. (Recall that the only limitation is that the stack must not be in high memory.) The @BANK SVC function 0 provides a technique for automatically transferring to an address in the new bank. This technique is called the transfer function. It relies on the assumption that since you are managing the entire 32K bank 1 or 2, your application should know exactly where it needs to transfer (that is, where the application originally placed the code to execute).

The code to perform a bank transfer is similar to the above example. Register pair HL is loaded with the transfer address. Register C, which contains the number of the bank to invoke, must have its high order bit (bit 7) set. After the specified bank is enabled, control is passed to the transfer address that is in HL. Upon entry to your routine in the new bank (referred to here as "PROGB"), register HL will contain the old return address so that PROGB will know where to return transfer. Register C will also contain the old bank number with bit 7 set and register B will contain a zero. This register set-up provides for an easy return to the routine in the old bank that invoked the bank transfer. An illustration of the transfer code follows:

```

LD      C,1           ;Specify bank 1
LD      B,0           ;Bring up bank 0
LD      HL,(TRAADR)   ;Set the transfer
                        ;address
SET      7,C          ;and denote a
                        ;transfer
LD      A,@BANK       ;Set @BANK SVC (102)
RST      28H
RETADR   JR      NZ,ERROR

```

Control is returned to "RETADR" under either of two conditions. If there was an error in executing the bank transfer (for example, if an invalid bank number was specified or the stack pointer is in high memory), the returned condition is NZ. If the transfer took place and PROGB transferred back, the returned condition is Z. Thus, the Z flag shows whether or not there was a problem with the transfer.

If PROGB needs to provide a return code, it must be done by using register pair DE, IX, or IY, as registers AF, BC, and HL are used to perform the transfer. (Or, some other technique can be used, such as altering the return transfer address to a known error trapping routine.)

PROGB should contain code that is similar to that shown earlier. For example, PROGB could be:

```

PROGB    PUSH      BC          ;Save old bank data
          PUSH      HL          ;Save the RET
                                   ;address
          ;
          ; your PROGB routines
          ;
          POP       HL          ;Recover transfer
                                   ;address
          POP       BC          ;Get bank transfer
                                   ;data
          LD        A,102       ;Set @BANK SVC
          RST       28H
          JR        NZ,ERROR    ;Error trap

```

PROGB saves the bank data (register BC). Don't forget that a transfer was effected and register C has bit 7 already set when PROGB is entered. PROGB also saves the address it needs to transfer back (which is in HL). It then performs whatever routines it has been coded for, recovers the transfer data, and issues the bank transfer request. As explained earlier, an NZ return condition from the @BANK SVC indicates that the bank transfer was not performed. You should verify that your application has not violated the integrity of the stack where the transfer data was stored.

Never place disk drivers, device drivers, device filters, or interrupt service routines in banks other than bank 0. It is possible to segment one of the above modules and place segments in bank 1 or 2, provided the segment containing the primary entry is placed in bank 0. You can transfer between segments by using the bank transfer techniques discussed above.

Device Driver and Filter Templates

Device independence has its roots in "byte I/O." Byte I/O is any I/O passed through a device channel one byte at a time.

Three primitive routines are available at the assembly language level for byte I/O. These byte I/O primitives can be used to build larger routines. The three primitives are the TRSDOS supervisor calls @GET, @PUT, and @CTL. @GET is used to input a byte from a device or file. @PUT is used to output a byte to a device or file. @CTL is used to communicate with the driver routine servicing the device or file.

Other supervisor calls perform byte I/O, such as @KBD (scan the keyboard and return the key code if a key is down), @DSP (display a character on the video screen), and @PRT (output a character to the line printer). These functions operate by first loading register pair DE with a pointer to a specific Device Control Block (DCB) assigned for use by the device, then issuing a @GET or @PUT SVC for input or output requests.

When TRSDOS passes control over to the device driver routine, the Z-80 flag conditions are unique for each different primitive. This enables the driver to establish which primitive was used to access the routine, so it can turn over the I/O request to the proper driver or filter subroutine according to the type of request — input, output, or control.

The following table shows the FLAG register conditions upon entry to a driver or filter:

C,NZ	= @GET primitive
Z,NC	= @PUT primitive
NZ,NC	= @CTL primitive

Register B contains the I/O direction code: 1 = @GET, 2 = @PUT, 4 = @CTL. Register C contains the character code that was passed in the @PUT or @CTL supervisor call. Register IX points to the TYPE byte (DCB + 0) of the Device Control Block. Registers BC, DE, HL, and IX have been saved on the stack and are available for use. Register AF is not saved; if you want it preserved, your program must do so.

Your driver must start with a standard front-end header (see "Memory Header"):

BEGIN	JR	START	iGo to actual code
			ibeginning
	DEFW	MODEND-1	iLast byte used by
			i module
	DEFB	7	iLength of name
	DEFM	'MODNAME'	iName
MODDCB	DEFW	\$\$	iDCB ptr. for this
			i module
	DEFW	0	iReserved by TRSDOS

At the start of the actual module code, test the condition of the F register flags for @GET, @PUT, and @CTL:

START	EQU	\$	
i		Actual module code start	
	JR	C,WASGET	iGo if @GET request
	JR	Z,WASPUT	iGo if @PUT request
	.		iWas @CTL request

At the label START, a test is made on the carry flag. If the carry was set, then the disk primitive must have been an input request (@GET). An input request could be directed to a part of the driver which only handles input from the device.

If the request was not from the @GET primitive, the carry will not be set. The next test checks to see if the zero flag is set. The zero condition is preset when a @PUT primitive was the initial request. The jump to WASPUT can go to a part of the driver that deals specifically with output to the device.

If neither the zero nor carry flags are set, the routine falls through to the next instruction (not shown), which would begin the part of the driver that handles @CTL calls. For example, you may want to have an RS-232C driver handle a BREAK by issuing a @CTL call so that the RS-232C driver emits a true modem break, but a CONTROL C would @PUT a X'03'.

Some drivers are written to assume that @CTL requests are to be handled exactly like @PUT requests. This is entirely up to the author and the function of the driver.

Note that when a device is routed to a disk file, TRSDOS ignores @CTL requests. That is, the @CTL codes are not written to the disk file.

On @GET requests, the character input should be placed in the accumulator. On output requests (either @PUT or @CTL), the character is obtained from register C. It is important for drivers and filters to observe return codes. Specifically, if the request is @GET and no byte is available, the driver returns an NZ condition and the accumulator contains a zero (that is, OR 1 : LD A,0 : RET). If a byte is available, the byte is placed in the accumulator and the Z flag is set (that is, LD A,CHAR : CP A : RET). If there is an input error, the error code is returned in the accumulator and the Z flag is reset (that is, LD A,ERRNUM : OR A : RET). On output requests, the accumulator will contain the byte output with the Z flag set if no error occurred. In the case of an output error, the accumulator must be loaded with the error code and the Z flag reset as shown above.

A filter module is inserted between the DCB and driver routine (or between the DCB and the current filter when it is applied to a DCB already filtered). The insertion is performed by the TRSDOS FILTER command once the filter module is resident and attached to a phantom DCB. The usual linkage for a filter is to access the chained module by calling the @CHNIO supervisor call with specific linkage data in registers IX and BC. Register IX is loaded with the filter's DCB pointer obtained from the memory header MODDCB pointer. Register B must contain the I/O direction code (1=@GET, 2=@PUT, 4=@CTL). This code is already in register B when the filter is entered. You can either keep register B undisturbed or load it with the proper direction code. Also, output requests expect the output byte to be in register C.

The DCB pointer obtained from MODDCB is passed in register DE by the SET command and is loaded into MODDCB by your filter initialization routine. The initialization routine needs to relocate the filter to high memory and attach itself to the DCB assigned by the SET command. If the initialization front end had transferred the DCB pointer from DE to IX, then the following code could be used to establish the TYPE byte and vector for the filter:

```
LD      (IX),47H      ;Init DCB type to
LD      (IX+1),E      ;FILTER, G/P/C I/O,
LD      (IX+2),D      ;& stuff vector
```

A filter module can operate on input, output, control, or any combination based on the author's design. The memory header provides a region for user data storage conveniently indexed by the module.

An illustration of a filter follows. The purpose of this filter is to add a linefeed on output whenever a carriage return is to be sent. Although the filter requires no data storage, the technique for accessing data storage is shown.

```

BEGIN      JR      START          ;Branch to start
           DEFW    FLTEND-1        ;Last byte used
           DEFB     6              ;Name length
           DEFM     'SAMPLE'        ;Name
MODDCB     DEFW     0              ;Link to DCB
           DEFW     0              ;Reserved
;          Data storage area for your filter
CR         EQU     0DH
LF         EQU     0AH
DATA$      EQU     $
DATA1      EQU     $-DATA$
           DEFB     0              ;Data storage
DATA2      EQU     $-DATA$
           DEFB     0              ;Data storage
;          Start of filter
START      JR      Z,GOTPUT        ;Go if @PUT
;          @GET and @CTL requests are chained to
;          the next module attached to the device.
;          This is accomplished by falling through
;          to the @CHNIO call. Note that the sample
;          filter does not affect the B register,
;          so the filter does not have to load it
;          with the direction code.
FLTPUT     PUSH     IX              ;Save your data
                                           ;pointer
RX01        LD      IX,(MODDCB)
           EQU     $-2              ;Grab the DCB vector
           LD      A,@CHNIO        ;and chain to it
           RST      2BH
           POP      IX
           RET
;          Filter code
GOTPUT     LD      IX,PFDATA$      ;Base register is
RX02        EQU     $-2              ;used to index data
           LD      A,C              ;Get character to
                                           ;test
           CP      CR              ;If not CR, put it
           JR      NZ,FLTPUT
           CALL     FLTPUT          ;else put it
RX03        EQU     $-2
           RET      NZ              ;Back on error
           LD      C,LF              ;Add linefeed
           JR      FLTPUT
FLTEND     EQU     $
;          Relocation table
RELTAB     DEFW     RX01,RX02,RX03
TABLEN     EQU     $-RELTAB/2

```

The relocation table, RELTAB, would be used by the filter initialization relocation routine.

@CTL Interfacing to Device Drivers

This section discusses the @CTL functions supported by the system device drivers. To invoke a @CTL function, point register pair DE to the Device Control Block (DCB), load the function code into register C, and issue the @CTL supervisor call. You can locate the DCB address by either 1) using the @GTDCB SVC, or 2) using the @OPEN SVC to open a File Control Block containing the device specification and using the FCB address. See the @CTL supervisor call for a list of the function codes and their meanings.

The @CTL functions are listed below for each driver.

Keyboard Driver (resident driver assigned to *KI)

A function value of X'03' clears the type-ahead buffer. This serves the same purpose as repeated calls to @KBD until no character is available.

A function value of X'FF' is reserved for system use.

All other function values are treated as @GET requests.

The module name assigned to this driver is "\$KI".

Video Driver (resident driver assigned to *DO)

All @CTL requests are treated as if they were @PUT requests.

The module name assigned to this driver is "\$DO".

Printer Driver (resident driver assigned to *PR)

The printer driver is transparent to all code values when requested by the @PUT SVC. That means that all values from X'00' through X'FF' (0-255) can be sent to the printer. If the FORMS filter is attached to the *PR device, then various codes are trapped and used by the filter according to parameters specified with the FORMS library command, as follows:

- X'0D' — Generates a carriage return and optionally a linefeed (ADDLF).
Generates form feeds as required.
- X'0A' — Treated the same way as X'0D'.
- X'0C' — Generates form feeds (via repeated line feeds if soft form feed).
(FFHARD = OFF)
- X'09' — Advances to next tab column.
- X'06' — Sets top-of-form by resetting the internal line counter to zero.

Other character codes may be altered if the user translation option of the FORMS command (XLATE) is set.

The printer driver accepts a function value of X'00' via the @CTL request to return the printer status. If the printer is available, the Z flag will be set and register A will contain X'30'. If the Z flag is reset, register A will contain the four high-order bits of the parallel printer port (bits 4-7).

The module name assigned to the printer driver is "\$PR". The module name of the FORMS filter is "\$FF".

COM Driver (non-resident driver for the RS-232C)

This driver handles the interfacing between the RS-232C hardware and byte I/O (usually the *CL device).

A @CTL function value of X'00' returns an image of the RS-232 status register in the accumulator. The Z flag will be set if the RS-232 is available for "sending" (that is, if the transmit holding register is empty and the flag conditions match as specified by SETCOM).

A function value of X'01' transmits a "modem break" until the next character is @PUT to the driver.

A function value of X'02' re-initializes the UART to the values last established by SETCOM.

A function value of X'04' enables or disables the WAKEUP feature.

All other function values are ignored and the driver returns with register A containing a zero value and the Z flag set.

The WAKEUP feature is useful for application software specializing in communications. The RS-232 hardware can generate a machine interrupt under any of three conditions: when the transmit holding register is empty, when a received character is available, or when an error condition has been detected (framing error, parity error, and so on). The COM driver makes use of the

"received character available" interrupt to take control when a fully formed character is in the holding register. The COM driver services the interrupt by reading the character and storing it in a one-character buffer. COM then normally returns from the interrupt.

An application can request that, instead of returning, control be passed to the application for immediate attention. Note that this action would occur during interrupt handling, and any processing by the application must be kept to a minimum before control is returned to COM via a RET instruction.

If you use a @CTL function value of X'04', then register IY must contain the address of the handling routine in your application. Upon return from the @CTL request, register IY contains the address of the previous WAKEUP vector. This should be restored when your application is finished with the WAKEUP feature.

When control is passed to your WAKEUP vector upon detection of a "receive character available" interrupt, certain information is immediately available. Register A contains an image of the UART status register. The Z flag is set if a valid character is actually available. The character, if any, is in the C register.

Since system overhead takes a small amount of time in the @GET supervisor call, you may need to @GET the character via standard device interfacing. This ensures that any filtering or linking in the *CL device chain will be honored. If, on the other hand, your application is attempting to transfer data at a very high rate (9600 baud or higher), you may need to bypass the @GET SVC and use the character immediately available in the C register. Note that this procedure bypasses the normal device chain (device routing and linking).

The module name of the COM driver is "\$CL."

8/Using the Supervisor Calls

Supervisor Calls (SVCs) are operating system routines that are available to assembly language programs. These routines alter certain system functions and conditions, provide file access, and perform various computations. They also perform I/O to the keyboard, video display, and printer.

Each SVC has a number which you specify to invoke it. These numbers range from 0 to 104.

In addition, under Version 6.2, you can write your own operating system routines using the numbers 124 through 127 to install your own SVC's. See Appendix E, "Programmable SVCs" for more information.

Calling Procedure

To call a TRSDOS SVC:

1. Load the SVC number for the desired SVC into register A. Also load any other registers which are needed by the SVC, as detailed under Supervisor Calls.
2. Execute a RST 28H instruction.

Note: If the SVC number supplied in register A is invalid, the system prints the message "System Error xx", where xx is usually 2B. It then returns you to TRSDOS Ready (*not* to the program that made the invalid SVC call).

The alternate register set (AF, BC, DE, HL) is not used by the operating system.

Program Entry and Return Conditions

When a program executed from the @CMNDI SVC is entered, the system return address is placed on the top of the stack. Register HL will point to the first non-blank character following the command name. Register BC will point to the first byte of the command line buffer.

Three methods of return from a program back to the system are available: the @ABORT SVC, the @EXIT SVC, and the RET instruction. For application programs and utilities, the normal return method is the @EXIT SVC. If no error condition is to be passed back, the HL register pair must contain a zero value. Any non-zero value in HL causes an active JCL to abort.

The @ABORT SVC can be used as an error return back to the system; it automatically aborts any active JCL processing. This is done by loading the value X'FFFF' into the HL register pair and internally executing an @EXIT SVC.

If stack integrity is maintained, a RET instruction can be used since the system return address is put on the stack by @CMNDI. This allows a return if the program was called with @CMNDR.

Most of the SVCs in TRSDOS Version 6 set the Z flag when the operation specified was successful. When an operation fails or encounters an error, the Z flag is reset (also known as NZ flag set) and a TRSDOS error code is placed in the A register. The remaining SVCs use the Z/NZ flag in differing ways, so you should refer to the description of the SVCs you are using to determine the exit conditions.

Supervisor Calls

The TRSDOS Supervisor Calls are:

Keyboard SVCs

@CKBRKC
@KBD
@KEY
@KEYIN

Printer and Video SVCs

@CLS
@DSP
@DSPLY
@LOGGER
@LOGOT
@MSG
@PRT
@PRINT
@VDCTL

Disk SVCs

@DCINIT
@DCRES
@DCSTAT
@RDSEC
@RDSSC
@RSLCT
@RSTOR
@SEEK
@SLCT
@STEPI
@VRSEC
@WRSEC
@WRSSC
@WRTRK

System Control SVCs

@ABORT
@BREAK
@CMNDI
@CMNDR
@EXIT
@FLAGS
@HIGH\$
@IPL
@LOAD
@RUN

Special Purpose Disk SVCs

@DIRRD
@DIRWR
@GTDCT
@HDFMT
@RDHDR
@RDTRK

Byte I/O SVCs

@CTL
@GET
@PUT

File Control SVCs

@CLOSE
@FEXT
@FNAME
@FSPEC
@INIT
@REMOV
@OPEN
@RENAM

Disk File Handler SVCs

@BKSP
@CKEOF
@LOC
@LOF
@PEOF
@POSN
@READ
@REW
@RREAD
@RWRIT
@SEEKSC
@SKIP
@VER
@WEOF
@WRITE

TRSDOS Task Control SVCs

@ADTSK
@CKTSK
@KLTSK
@RMTSK
@RPTSK

Special Overlay SVCs

@CKDRV
@DEBUG
@DODIR
@ERROR
@PARAM
@RAMDIR

Miscellaneous SVCs

@BANK
@DATE
@DECHEX
@DIV8
@DIV16
@HEXDEC
@HEX8
@HEX16
@MUL8
@MUL16
@PAUSE
@SOUND
@TIME
@WHERE

Special Purpose SVCs

@CHNIO
@GTDCB
@GTMOD

See the pages that follow for a detailed description of each supervisor call.

Abort Program

Loads HL with an X'FFFF' error code and exits through the @EXIT supervisor call. Any active JCL processing is aborted.

Entry Conditions:

A = 21 (X'15')

General:

This SVC does not return.

Example:

See the example for @EXIT in Sample Program B, lines 206-207.

Add an Interrupt Level Task

Adds an interrupt level task to the real time clock task table. The task slot number can be 0-11; however, some slots are already assigned to certain functions in TRSDOS. Slot assignments 0-7 are low priority tasks executing every 266.67 milliseconds. Slots 8-10 are medium priority tasks executing every 33.33 milliseconds. Slot 11 is a high priority task, executing every 16.66 milliseconds High Speed or 33.33 milliseconds Low Speed. The system uses task slots 3, 7, 9, and 10 for the ALIVE, TRACE, SPOOL, and TYPE-AHEAD functions, respectively.

It is a good practice to remove an existing task (using the @RMTSK or @KLTSK SVC) before installing a new task in the same task slot.

Entry Conditions:

A = 29 (X'1D')
DE = *pointer to Task Control Block (TCB)*
C = *task slot assignment (0-11)*

Exit Conditions:

Success always.
HL and AF are altered by this SVC.

The Task Control Block, or TCB, is a 2-byte block of RAM which contains the address of the task driver entry point. If your task is prefixed with the memory header described earlier under "Device Access," then the TCB can be stored in the memory header data storage area. If the task is not a driver or filter, the TCB can be stored in the memory header location MODDCB. Upon entry to your task routine, the IX register contains the TCB address.

Example:

See Sample Program F, lines 109-120.

Memory Bank Use

Controls 32K memory bank operation. The top half of the main 64K block is bank 0, and the alternate 64K block is divided into banks 1 and 2. The system maintains two locations to perform bank management. These areas are known as "bank available RAM" (BAR) and "bank in use RAM" (BUR).

If the Stack Pointer is not X'7FFE' or lower, the SVC aborts with an Error 43 only if B = 0.

Entry Conditions:

A = 102 (X'66')

B selects one of the following functions:

If B = 0, the specified bank is selected and is made addressable. The 32K bank starts at X'8000' and ends at X'FFFF'.

C = bank number to be selected (0-2)

If bit 7 is set, then execution will resume in the newly loaded bank at the address specified.

HL = address to start execution in the new bank

If B = 1, reset BUR and show the bank not in use.

C = bank number to be selected (0-2)

If B = 2, test BUR if bank is in use.

C = bank number to be selected (0-2)

If B = 3, set BUR to show bank in use.

C = bank number to be selected (0-2)

If B = 4, return number of bank currently selected.

Exit Conditions:

If B = 0:

Success, Z flag set.

C = the bank number that was replaced. If bit 7 was set in register C on entry, it is also set on exit.

HL = SVC return address. By keeping the contents of C and HL, you can later return to the instruction following the first @BANK SVC. See "Interfacing RAM Banks 1 and 2" for more information.

Failure, NZ flag set. Bank not present or parameter error.

A = error number

If B = 1:

Success, Z flag set. Bank available for use.

Failure, NZ flag set. Bank not present.

If B = 2:

Success always.

If Z flag is set, then the bank is available for use.

If NZ flag is set, then test register A:

If A ≠ X'2B' then the bank is either in use or it does not exist on your machine. Banks 1 and 2 produce this error on a 64K machine.

If A = X'2B' then an entry parameter is out of range.

If B = 3:

Success, Z flag set. Bank is now reserved for your use.

Failure, NZ flag set. Test register A:

If A ≠ X'2B' then the bank is already in use or does not exist. Banks 1 and 2 produce this error on a 64K machine.

If A = X'2B' then an entry parameter is out of range.

If B = 4:

Success always.

A = number of the bank which is currently resident

General:

AF is altered for all functions.

BC is altered if the SVC is successful.

Example:

See the section "Interfacing RAM Banks 1 and 2"

Backspace One Logical Record

Performs a backspace of one logical record.

Entry Conditions:

A = 61 (X'3D')

DE = *pointer to FCB of the file to backspace*

Exit Conditions:

If the Z flag is set or if A = X'1C' or X'1D', then the operation was successful.

The LOC pointer to the file was backspaced one record. Otherwise,
A = *error number*.

If A = X'1C' is returned, the file pointer is positioned at the end of the file.

Any Appending operations would be performed here.

If A = X'1D' is returned, the file pointer is positioned beyond the end of the file.

General:

Only AF is altered by this SVC.

If the LOC pointer was at record 0 when the call was executed, the results are indeterminate.

Example:

See the example for @LOC in Sample Program C, lines 305-311.

Set Break Vector

Sets a user or system break vector. The BREAK vector is an abort mechanism; there is no return.

The BREAK vector executes whenever the following conditions occur at the same time: 1) the Program Counter is greater than X'2400; 2) the BREAK key is pressed, and 3) a real time clock interrupt which executes 30 times per second occurs.

After executing this SVC, you must reset bit 4 of SFLAG\$. The BREAK flag in KFLAG\$ (bit 0) requires the setting of SFLAG\$ bit 4 and a delay of 0.1 to 0.5 second to clear any other interrupts that may be pending. Then you can enter your BREAK key handler (in which the BREAK key bit in SFLAG\$ is reset). See KFLAG\$ and SFLAG\$ in the section about the @FLAGS SVC for more information.

Entry Conditions:

A = 103 (X'67')

HL = *user break vector*

HL = 0 (sets system break vector)

Exit Conditions:

Success always.

HL = *existing break vector* (if user break vector was set)

Note: @EXIT and @CMNDI automatically restore BREAK to the system handler. @CMNDR does not do this.

Pass Control to Next Module in Device Chain

Passes control to the next module in the device chain.

Entry Conditions:

A = 20 (X'14')

IX = *contents of DCB in the header block*

B = *GET/PUT/CTL direction code (1/2/4)*

C = *character (if output request)*

General:

IX is not checked for validity.

Example:

See the section "Device Driver and Filter Templates."

Check BREAK bit and clear it**Version 6.2 only**

Checks to see if the BREAK key has been pressed. If a BREAK condition exists, @CKBRKC resets the break bit, Bit 0 of KFLAG\$.

Entry Conditions:

A = 106(X'6A')

Exit Conditions:

Success always.

If Z flag is set, the break bit was not detected. If NZ flag is set, the break bit was detected and is cleared. If the BREAK key is being depressed, the SVC will not return until the key is released.

General:

Only AF is altered by this SVC.



Check Drive

Checks a drive reference to ensure that the drive is in the system and a TRSDOS Version 6 or LDOS 5.1.3 (Model III Hard Disk Operating System) formatted disk is in place.

Entry Conditions:

A = 33 (X'21')

C = *logical drive number (0-7)*

Exit Conditions:

Success always.

If Z flag is set, the drive is ready.

If CF is set, the disk is write protected.

If NZ flag is set, the drive is not ready. The user may examine DCT + 0 to see if the drive is disabled.

Example:

See Sample Program D, lines 35-55.

Check for End-Of-File

Checks for the end of file at the current logical record number.

Entry Conditions:

A = 62 (X'3E')

DE = *pointer to the FCB of the file to check*

Exit Conditions:

Success always.

If Z flag is set, LOC does not point at the end of file (LOC < LOF).

If NZ flag is set, test A for error number:

If A = X'1C', LOC points at the end of the file (LOC = LOF).

If A = X'1D', LOC points beyond the end of the file (LOC > LOF).

If A ≠ X'1C' or X'1D', then A = *error number*.

General:

Only AF is altered by this SVC.

Example:

See Sample Program C, lines 352-353.

Check if Task Slot in Use

Checks to see if the specified task slot is in use.

Entry Conditions:

A = 28 (X'1C')

C = *task slot to check* (0-11)

Exit Conditions:

Success always.

If Z flag is set, the task slot is available for use.

If NZ flag is set, the task slot is already in use.

General:

AF and HL are altered by this SVC.

Example:

See Sample Program F, lines 70-73.

Close a File or Device

Terminates output to a file or device. Any unsaved data in the buffer area is saved to disk and the directory is updated. All files that have been written to must be closed, as well as all files opened with UPDATE or higher access.

If you remove a diskette containing an open file, any attempt to close the file results in the message:

**** CLOSE FAULT **** *error message*, <ENTER> to retry, <BREAK> to abort

where *error message* is usually "Drive not ready" You may put the diskette back in the drive and:

1. Press **(ENTER)** to close the file.
2. Press **(BREAK)** to abort the close.

If you press **(BREAK)**, the NZ flag is set and Register A contains X'20', the error code for an illegal drive number error.

Entry Conditions:

A = 60 (X'3C')

DE = *pointer to FCB or DCB to close*

Exit Conditions:

Success, Z flag set. The file or device was closed. The filespec (excluding the password) or the devspec is returned to the FCB or DCB.

Failure, NZ flag set.

A = *error number*

General:

Only AF is altered by this SVC.

Example:

See Sample Program C, lines 360-368.

Clear Video Screen**Version 6.2 only**

Clears the video screen by sending a Home Cursor (X'1C') and Clear to End of Frame (X'1F') sequence to the video driver.

Entry Conditions:

A = 105(X'69')

Exit Conditions:

Success, Z flag is set.

Failure, NZ is set.

A = *error number*

General:

Only AF is altered by this SVC.

Execute Command with Return to System

Passes a command string to TRSDOS for execution. After execution is complete, control returns to TRSDOS Ready. If the command gets an error, it still returns to TRSDOS Ready.

Entry Conditions:

A = 24 (X'18')

HL = *pointer to buffer containing command string terminated with X'0D'*
(up to 80 bytes, including the X'0D')

General:

This SVC does not return.

Example:

See Sample Program E, lines 43-58.

Execute Command

Executes a command or program and returns to the calling program. The executed program should maintain the Stack Pointer and exit via a RET instruction. All TRSDOS library commands comply with this requirement.

If bit 4 of CFLAG\$ is set (see the @FLAGS SVC), then @CMNDR executes only system library commands.

Entry Conditions:

A = 25 (X'19')

HL = *pointer to buffer containing command string terminated with X'0D'*
(up to 80 bytes, including the X'0D')

Exit Conditions:

Success always.

HL = *return code* (See the section "Converting to TRSDOS Version 6" for information on return codes.)

Registers AF, BC, DE, IX, and IY are altered by the command or program executed by this SVC.

If the command invokes a user program which uses the alternate registers, they are modified also.

Example:

See Sample Program E, lines 18-29.

Output a Control Byte

Outputs a control byte to a logical device. The DCB TYPE byte (DCB + 0, Bit 2) must permit CTL operation. See the section "@CTL Interfacing to Device Drivers" for information on which of the functions listed below are supported by the system device drivers.

Entry Conditions:

A = 5 (X'05')

DE = *pointer to DCB to control output*

C selects one of the following functions:

If C = 0, the status of the specified device will be returned.

If C = 1, the driver is requested to send a BREAK or force an interrupt.

If C = 2, the initialization code of the driver is to be executed.

If C = 3, all buffers in the driver are to be reset. This causes all pending I/O to be cleared.

If C = 4, the wakeup vector for an interrupt-driven driver is specified by the caller.

IY = address to vector when leaving driver. If IY = 0, then the wakeup vector function is disabled. The RS-232C driver COM/DVR (\$CL), is the only system driver that provides wakeup vectoring.

If C = 8, the next character to be read will be returned. This allows data to be "previewed" before the actual @GET returns the character.

Exit Conditions:

If C = 0,

Z flag set, device is ready

NZ flag set, device is busy

A = status image, if applicable

Note: This is a hardware dependent image.

If C = 1,

Success, Z flag set. BREAK or interrupt generated.

Failure, NZ flag set

A = *error number*

If C = 2,

Success, Z flag set. Driver initialized.

Failure, NZ flag set

A = *error number*

If C = 3,

Success, Z flag set. Buffers cleared.

Failure, NZ flag set.

A = *error number*

If C = 4,

Success always.

IY = *previous vector address*

This function is ignored if the driver does not support wakeup vectoring.

If C = 8,

Success, Z flag set. Next character returned.

A = *next character in buffer*

Failure, NZ flag set. Test register A:

If A = 0, no pending character is in buffer

If A ≠ 0, A contains *error number*. (TRSDOS driver returns Error 43.)

General:

BC, DE, HL, and IX are saved.

Function codes 5 to 7, 9 to 31, and 255 are reserved for the system. Function codes 32 to 254 are available for user definition.

Entry and exit conditions for user-defined functions are up to the design of the user-supplied driver.

Example:

See the section "Device Driver and Filter Templates."

Get Date

Returns today's date in display format (MM/DD/YY).

Entry Conditions:

A = 18 (X'12')

HL = *pointer to 8-byte buffer to receive date string*

Exit Conditions:

Success always.

HL = *pointer to the end of the buffer supplied + 1*

DE = *pointer to start of DATE\$ storage area in TRSDOS*

BC is altered by this SVC.

Example:

See Sample Program F, lines 252-253.

Initialize the FDC

Issues a disk controller initialization command. The floppy disk driver treats this the same as @RSTOR (SVC 44).

Entry Conditions:

A = 42 (X'2A')

C = *logical drive number* (0-7)

Exit Conditions:

Success, Z flag set.

Failure, NZ flag set.

A = *error number*

Example:

See the example for @CKDRV in Sample Program D, lines 38-39.

Reset the FDC

Issues a disk controller reset command. The floppy disk driver treats this the same as @RSTOR (SVC 44).

Entry Conditions:

A = 43 (X'2B')

C = *logical drive number (0-7)*

Exit Conditions:

Success, Z flag set.

Failure, NZ flag set.

A = *error number*

Example:

See the example for @CKDRV in Sample Program D, lines 38-39.

Test if Drive Assigned in DCT

Tests to determine whether a drive is defined in the Drive Code Table (DCT).

Entry Conditions:

A = 40 (X'28')

C = *logical drive number (0-7)*

Exit Conditions:

Success always.

If Z is set, the specified drive is already defined in the DCT.

If NZ is set, the specified drive is not defined in the DCT.

General:

Only AF is altered by this SVC.

Example:

See Sample Program D, lines 27-33.

Enter DEBUG

Forces the system to enter the DEBUG utility. Pressing **Ⓢ** (**ENTER**) from the DEBUG monitor causes program execution to continue with the next instruction. If you want to use the functions in the extended debugger when DEBUG is entered in this fashion, you must issue the DEBUG (E) command (optionally with the @CMNDR SVC) before this SVC is executed.

Entry Conditions:

A = 27 (X'1B')

General:

This SVC does not return unless **Ⓢ** is entered in DEBUG.

Example:

See Sample Program A, lines 54-60.

Convert Decimal ASCII to Binary

Converts a decimal ASCII string to a 16-bit binary number. Overflow is not trapped. Conversion stops on the first out-of-range character.

Entry Conditions:

A = 96 (X'60')

HL = *pointer to decimal string*

Exit Conditions:

Success always.

BC = *binary conversion of ASCII string*

HL = *pointer to the terminating byte*

AF is altered by this SVC.

Example:

See Sample Program B, lines 88-95.

Directory Record Read

Reads a directory sector that contains the directory entry for a specified Directory Entry Code (DEC). The sector is placed in the system buffer and the register pair HL points to the first byte of the directory entry specified by the DEC.

Entry Conditions:

A = 87 (X'57')

B = *Directory Entry Code of the file*

C = *logical drive number (0-7)*

Exit Conditions:

Success, Z flag set.

HL = *pointer to directory entry specified by register B*

Failure, NZ flag set.

A = *error number*

HL is altered.

General:

AF is always altered.

If the drive does not contain a disk, this SVC may hang indefinitely waiting for formatted media to be placed in the drive. The programmer should perform a @CKDRV SVC before executing this call.

If the Directory Entry Code is invalid, the SVC may not return or it may return with the Z flag set and HL pointing to a random address. Care should be taken to avoid using the wrong value for the DEC in this call.

Example:

See Sample Program C, lines 152-174.

Directory Record Write

Writes the system buffer back to the disk directory sector that contains the directory entry of the specified DEC.

Entry Conditions:

A = 88 (X'58')
B = *Directory Entry Code of the file*
C = *logical drive number (0-7)*

Exit Conditions:

Success, Z flag set.
HL = *pointer to directory entry specified by register B*
Failure, NZ flag set.
A = *error number*
HL is altered.

General:

AF is always altered.
If the drive does not contain a disk, this SVC may hang indefinitely waiting for formatted media to be placed in the drive. The programmer should perform a @CKDRV SVC before executing this call.
If the Directory Entry Code is invalid, the SVC may not return or it may return with the Z flag set and HL pointing to a random address. Care should be taken to avoid using the wrong value for the DEC in this call.

Example:

See the example for @DIRRD in Sample Program C, lines 152-174.

8-Bit Divide

Performs an 8-bit unsigned integer divide.

Entry Conditions:

A = 93 (X'5D')

E = *dividend*

C = *divisor*

Exit Conditions:

Success always.

A = *quotient*

E = *remainder*

No other registers are altered.

Example:

See Sample Program B, lines 61-64.

16-Bit by 8-Bit Divide

Performs a division of a 16-bit unsigned integer by an 8-bit unsigned integer.

Entry Conditions:

A = 94 (X'5E')

HL = *dividend*

C = *divisor*

Exit Conditions:

Success always.

HL = *quotient*

A = *remainder*

No other registers are altered.

Example:

See Sample Program B, lines 105-109.

Do Directory Display / Buffer

Reads files from a disk directory or finds the free space on a disk. The directory information is either displayed on the screen (in five-across format) or sent to a buffer. The directory information buffer consists of 18 bytes per active, visible file: the first 16 bytes of the directory record, plus the ERN (ending record number). An X'FF' marks the buffer end.

Entry Conditions:

A = 34 (X'22')

C = *logical drive number (0-7)*

B selects one of the following functions:

If B = 0, the directory of the visible, non-system files on the disk in the specified drive is displayed on the screen. The filenames are displayed in columns, 5 filenames per line.

If B = 1, the directory is written to memory.

HL = *pointer to buffer to receive information*

If B = 2, a directory of the files on the specified drive is displayed for files that are visible, non-system, and match the extension partspec pointed to by HL.

HL = *partspec for the filename's extension*

This field must contain a valid 3-character extension, padded with dollar signs (\$). For example, to display all visible, non-system files that have the letter 'C' as the first character of the extension, HL should point to the string "C\$\$".

If B = 3, a directory of the files on the specified drive is written to the buffer that is specified by HL for files that match the extension partspec pointed to by HL.

HL = *pointer to the 3-byte partspec and to the buffer to receive the directory records (see general notes)*

Keep in mind that the area pointed to by HL is shared. If you are using this buffer more than once, you have to re-create the partspec in the buffer before each call because the previous call will have erased the partspec by writing the directory records.

If B = 4, the disk name, original free space, and current free space on the disk is read.

HL = *pointer to a 20-byte buffer to receive information*

Exit Conditions:

Success, Z flag set.

If B = 1 or 3, the directory records have been stored.

HL = *pointer to the beginning of the buffer*

If B = 0 or 2, the filenames or matching filenames are displayed with 5 filenames per line.

If B = 4, the disk name and free space information are stored in the format:

Bytes 0-7 = Disk name. Disk name is padded on the right with blanks (X'20').

Bytes 8-15 = Creation date (the date the disk was formatted or was the target disk in a mirror image backup). The date is in the format MM/DD/YY.

Bytes 16-17 = Total K originally available in binary LSB-MSB format.

Bytes 18-19 = Free K available now in binary LSB-MSB format.

HL = *pointer to the beginning of the data area*

Failure, NZ flag set.

A = *error number*

General:

AF is the only register altered by this SVC.

The size of the buffer to receive directory records must be large enough to hold directory entries for the maximum number of files allowed on the drive and disk you specify. For example, if the drive is a hard disk, you must be able to store 256 directory entries, and each entry requires 18 bytes of storage. For more information on calculating the amount of space needed for this buffer, see the tables under "Directory Records." They give the maximum number of entries allowed on a given type of disk. You must add 2 records to this value when B = 1 to store the directory entry for DIR/SYS and BOOT/SYS.

Example:

See Sample Program E, lines 32-40.

Display Character

Outputs a byte to the video display. The byte is displayed at the current cursor position.

Entry Conditions:

A = 2 (X'02')

C = *byte to display*

Exit Conditions:

Success, Z flag set.

A = *byte displayed*

Failure, NZ flag set.

A = *error number*

General:

DE is altered by this SVC.

Example:

See Sample Program C, lines 219-221.

Display Message Line

Displays a message line, starting at the current cursor position. The line must be terminated with either a carriage return (X'0D') or an ETX (X'03'). If an ETX terminates the line, the cursor is positioned immediately after the last character displayed.

Entry Conditions:

A = 10 (X'0A')

HL = *pointer to first byte of message*

Exit Conditions:

Success, Z flag set.

Failure, NZ flag set.

A = *error number*

General:

AF and DE are altered by this SVC.

Example:

See Sample Program C, lines 35-37.

Entry to Post an Error Message

Provides an entry to post an error message. If bit 7 of register C is set, the error message is displayed and return is made to the calling program. If bit 6 is not set, the extended error message is displayed. Under versions prior to 6.2 the error display is in the following format:

```
*** Errcod=xx, Error message string ***
    <filespec or devspec>
Referenced at X'dddd'
```

Under Version 6.2 the error display is in the following format:

```
**Error code = xx, Returns to X' dddd'
**Error message string
<filespec, devspec, or open FCB/DCB status>
Last SVC = nnn, Returned to X' rrrr'
```

dddd is the return address of the @ERROR SVC in the application program.

nnn is the last SVC executed before the @ERROR SVC request.

rrrr is the address the previous SVC returned to in the application program.

If bit 6 is set, then only the "Error message string" is displayed. This bit is ignored if bit 6 of SFLAG\$ (the extended error message bit) is set. If bit 6 of CFLAG\$ is set, then no error message is displayed. If bit 7 of CFLAG\$ is set, then the "Error message string" is placed in a user buffer pointed to by register pair DE. See @FLAGS (SVC 101) for more information on SFLAG\$ and CFLAG\$.

Entry Conditions:

A = 26 (X'1A')

C = error number with bits 6 and 7 optionally set

Exit Conditions:

Success always.

General:

To avoid a looping condition that could result from the display device generating an error, do not check for errors after returning from @ERROR.

If you do not set bit 6 of register C, then you should execute this SVC only after an error has actually occurred.

Example:

See Sample Program C, lines 379-389.

Exit to TRSDOS

This is the normal program exit and return to TRSDOS. An error exit can be done by placing a non-zero value in HL. Values 1 to 62 indicate a primary error as described in TRSDOS Error Codes (Appendix A). (A non-zero value in HL causes an active JCL to abort.)

Entry Conditions:

A = 22 (X'16')

HL = *Return Code*

If HL = 0, then no error on exit.

If HL ≠ 0, then the @ABORT SVC returns X'FFFF' in HL automatically.

General:

This SVC does not return.

Example:

See Sample Program B, lines 206-207.

Set Up Default File Extension

Inserts a default file extension into the File Control Block if the file specification entered contains no extension. @FEXT must be done before the file is opened.

Entry Conditions:

A = 79 (X'4F')

DE = *pointer to FCB*

HL = *pointer to default extension* (3 characters; alphabetic characters must be upper case and first character must be a letter)

Exit Conditions:

Success always.

AF and BC are altered by this SVC.

If the default extension is used, HL is also altered.

Example:

See Sample Program C, lines 111-132.

Point IY to System Flag Table

Points the IY register to the base of the system flag table. The status flags listed below can be referenced off IY. You can alter those bits marked with an asterisk (*). Bits without an asterisk are indicators of current conditions, or are unused or reserved.

Note: You may wish to save KFLAG\$ and SFLAG\$ if you intend to modify them in your program, and restore them on exit.

Entry Conditions:

A = 101 (X'65')

Exit Conditions:

Success always.

IY = *pointer to the following system information:*

IY - 1 Contains the overlay request number of the last system module resident in the system overlay region.

IY + 0 = AFLAG\$ (allocation flag under Version 6.2 only)

Contains the starting cylinder number to be used when searching for free space on a diskette. It is normally 1. If the starting cylinder number is larger than the number of cylinders for a particular drive, 1 is used for that drive.

IY + 2 = CFLAG\$

* bit 7 — If set, then @ERROR will transfer the "Error message string" to your buffer instead of displaying it. The message is terminated with X'0D.'

* bit 6 — If set, do not display system error messages 0-62. See @ERROR (SVC 26) for more information.

* bit 5 — If set, sysgen is not allowed.

* bit 4 — If set, then @CMNDR will execute only system library commands.

bit 3 — If set, @RUN is requested from either the SET or SYSTEM (DRIVER =) commands.

bit 2 — If set, @KEYIN is executing due to a request from SYS1.

bit 1 — If set, @CMNDR is executing. This bit is reset by @EXIT and @CMNDI.

* bit 0 — If set, HIGH\$ cannot be changed using @HIGH\$ (SVC 100). This bit is reset by @EXIT and @CMNDI.

IY + 3 = DFLAG\$ (device flag)

* bit 7 — "1" if GRAPHIC printer capability desired on screen print ((CONTROL) causes screen print. See the SYSTEM (GRAPHIC) command under "Technical Information on TRSDOS Commands and Utilities.")

bit 6 — "1" if KSM module is resident

bit 5 — Currently unused

bit 4 — "1" if MemDisk active

bit 3 — Reserved

bit 2 — "1" if Disk Verify is enabled

* bit 1 — "1" if TYPE-AHEAD is active

bit 0 — "1" if SPOOL is active

IY + 4 = EFLAG\$ (ECI flag under Version 6.2 only)

Indicates the presence of an ECI program. If any of the bits are set, an ECI is used, rather than the SYS1 interpreter. The ECI program may use these bits as necessary. However, at least one bit must be set or the ECI is not executed.

IY + 5 = FEMSK\$ (mask for port 0FEH)
 IY + 8 = IFLAG\$ (international flag)
 * bit 7 — If "1," 7-bit printer filter is active
 If "0," normal 8-bit filters are present
 * bit 6 — If "1," international character translation will be performed by printer driver
 If "0," characters received by printer driver will be sent to the printer unchanged
 bit 5 — Reserved for future languages
 bit 4 — Reserved for future languages
 bit 3 — Reserved for future languages
 bit 2 — Reserved for future languages
 bit 1 — If "1," German version of TRSDOS is present
 bit 0 — If "1," French version of TRSDOS is present
 If bits 5-0 are all zero, then USA version of TRSDOS is present.
 IY + 10 = KFLAG\$ (keyboard flag)
 bit 7 — "1" if a character is present in the type-ahead buffer
 bit 6 — Currently unused
 * bit 5 — "1" if CAPS lock is set
 bit 4 — Currently unused
 bit 3 — Currently unused
 * bit 2 — "1" if **ENTER** has been pressed
 * bit 1 — "1" if **SHIFT @** has been pressed (PAUSE)
 * bit 0 — "1" if **BREAK** has been pressed
 Note: To use bits 0-2, you must first reset them and then test to see if they become set.
 IY + 12 = MODOUT (image of port 0ECH)
 IY + 13 = NFLAG\$ (network flag under Version 6.2)
 bit 7 — Reserved for system use.
 bit 6 — If set, the application program is in the task processor. Programmers must **not** modify this bit.
 bit 5 — Reserved for system use.
 bit 4 — Reserved for system use.
 bit 3 — Reserved for system use.
 bit 2 — Reserved for system use.
 bit 1 — Reserved for system use.
 * bit 0 — If set, the "file open bit" is written to the directory.
 IY + 14 = OPREG\$ (memory management & video control image)
 IY + 17 = RFLAG\$ (retry flag under Version 6.2 only)
 Indicates the number of retries for the floppy disk driver.
 This should be an even number larger than two.
 IY + 18 = SFLAG\$ (system flag)
 bit 7 — "1" if DEBUG is to be turned on
 * bit 6 — "1" if extended error messages desired (see @ERROR for message format); overrides the setting of bit 6 of register C on @ERROR (SVC 26) and should be used only when testing
 bit 5 — "1" if DO commands are being executed
 * bit 4 — "1" if BREAK disabled
 bit 3 — "1" if the hardware is running at 4 mhz (SYSTEM (FAST)). If "0," the hardware is running at 2 mhz (SYSTEM (SLOW)).
 * bit 2 — "1" if LOAD called from RUN
 * bit 1 — "1" if running an EXECute only file
 * bit 0 — "1" specifies no check for matching LRL on file open and do not set file open bit in directory. This bit should be set just before executing an @OPEN (SVC 59) if you want to force the opened file to be READ only during current I/O operations. As soon as either call is executed, SFLAG\$ bit 0 is reset. If you want to disable LRL checking on another file, you must set SFLAG\$ bit 0 again.

IY + 19 = TFLAG\$ (type flag under Version 6.2 only)
 Identifies the Radio Shack hardware model. TFLAG\$
 allows programs to be aware of the hardware environ-
 ment and the character sets available for the display.
 Current assignments are:
 2 indicates Model II
 4 indicates Model 4
 5 indicates Model 4P
 12 indicates Model 12
 IY + 20 = UFLAG\$ (user flag under Version 6.2 only)
 May be set by application programs and is sysgened
 properly.
 IY + 21 = VFLAG\$
 bit 7 — Reserved for system use
 * bit 6 — "1" selects solid cursor, "0" selects blinking cursor
 bit 5 — Reserved for system use
 * bit 4 — "1" if real time clock is displayed on the screen
 bits 0-3 — Reserved for system use
 IY + 22 = WRINTMASK\$ (mask for WRINTMASK port)
 IY + 26 = SVCTABPTR\$ (pointer to the high order byte of the SVC table
 address; low order byte = 00)
 IY + 27 = Version ID byte (60H = TRSDOS version 6.0.x.x,
 61H = TRSDOS version 6.1.x.x, etc.)
 IY - 47 = Operating system release number. Provides a third and fourth
 character (12H = TRSDOS version x.x.1.2)
 IY + 28
 to
 IY + 30 = @ICNFG vector
 IY + 31
 to
 IY + 33 = @KITSK vector

Get Filename

Gets the filename and extension from the directory using the specified Directory Entry Code (DEC) for the file.

Entry Conditions:

A = 80 (X'50')

DE = *pointer to 15-byte buffer to receive filename/extension:drive, followed by a X'0D' as a terminator*

B = *DEC of desired file*

C = *logical drive number of drive containing file (0-7)*

Exit Conditions:

Success, Z flag set.

HL = *pointer to directory entry specified by register B*

Failure, NZ flag set.

A = *error number*

HL is altered.

General:

AF and BC are always altered.

If the drive does not contain a disk, this SVC may hang indefinitely waiting for formatted media to be placed in the drive. The programmer should perform a @CKDRV SVC before executing this call.

If the Directory Entry Code is invalid, the SVC may not return or it may return with the Z flag set and HL pointing to a random address. Care should be taken to avoid using the wrong value for the DEC in this call.

Example:

See Sample Program C, lines 274-286.



Assign File or Device Specification

Moves a file or device specification from an input buffer into a File Control Block (FCB). Conversion of lower case to upper case is made automatically.

Entry Conditions:

A = 78 (X'4E')

HL = *pointer to buffer containing filespec or devspec*

DE = *pointer to 32-byte FCB or DCB*

Exit Conditions:

Success always.

If the Z flag is set, the file specification is valid.

HL = *pointer to terminating character*

DE = *pointer to start of FCB*

If the NZ flag is set, a syntax error was found in the filespec.

HL = *pointer to invalid character*

DE = *pointer to start of FCB*

A = *invalid character*

General:

AF and BC are altered.

Example:

See Sample Program C, lines 53-65.

Get One Byte From Device or File

Gets a byte from a logical device or a file. The DCB TYPE byte (DCB + 0, Bit 0) must permit a GET operation for this call to be successful.

Entry Conditions:

A = 3 (X'03')

DE = *pointer to DCB or FCB*

Exit Conditions:

Success, Z flag set.

A = *character read from the device or file*

Failure, NZ flag set. Test register A:

If A = 0, no character was available.

If A ≠ 0, A contains *error number*.

Example:

See the section "Device Driver and Filter Templates."

Get Device Control Block Address

Finds the location of a Device Control Block (DCB). If DE = 0 (no device name specified), HL returns the address of the first unused DCB found.

Entry Conditions:

A = 82 (X'52')

DE = 2-character device name (E = first character, D = second character)

Exit Conditions:

Success, Z flag set. DCB was found.

HL = pointer to start of DCB

Failure, NZ flag set. No DCB was available.

A = Error 8 (Device not available)

HL is altered.

General:

AF is always altered by this SVC.

Example:

See the section "Device Driver and Filter Templates."

Get Drive Code Table Address

Gets the address of the Drive Code Table for the requested drive.

Entry Conditions:

A = 81 (X'51')

C = *logical drive number (0-7)*

Exit Conditions:

Success always.

IY = *pointer to the DCT entry for the specified drive*

AF is always altered by this SVC.

General:

If the drive number is out of range, the IY pointer will be invalid. This call does not return Z/NZ to indicate if the drive number specified is valid (0-7) or enabled.

Example:

See the example for @DCSTAT in Sample Program D, lines 27-33.

Get Memory Module Address

Locates a memory module, if the standard memory header is at the start of the module. The scanning starts with the system drivers in low memory, then moves to any high memory modules. If any routine is encountered that does not start with a proper header, scanning stops.

Entry Conditions:

A = 83 (X'53')

DE = *pointer to memory module name in upper case, terminated with any character in the range 00-31*

Exit Conditions:

Success always.

If the Z flag is set, the module was found.

HL = *pointer to first byte of memory header*

DE = *pointer to first byte after module name*

If the NZ flag is set, the module was not found.

HL is altered.

General:

AF is always altered by this SVC.

Example:

See Sample Program F, lines 144-154.

Hard Disk Format

Passes a format drive command to a hard disk driver. If the hard disk controller accepts it as a valid command, then it formats the entire disk drive. If the hard disk controller does not accept it, then an error is returned. Radio Shack hardware does not currently support @HDFMT.

Entry Conditions:

A = 52 (X'34')

C = *logical drive number (0-7)*

Exit Conditions:

Success, Z flag set.

Failure, NZ flag set.

A = *error number*

Convert Binary to Decimal ASCII

Converts a binary number in HL to decimal ASCII.

Entry Conditions:

A = 97 (X'61')

HL = *number to convert*

DE = *pointer to 5-character buffer to hold converted number*

Exit Conditions:

Success always.

DE = *pointer to end of buffer + 1*

AF, BC, and HL are altered by this SVC.

Example:

See Sample Program B, lines 73-76.

Convert 1 Byte to Hex ASCII

Converts a 1-byte number to hexadecimal ASCII.

Entry Conditions:

A = 98 (X'62')

C = *number to convert*

HL = *pointer to a 2-character buffer to hold the converted number*

Exit Conditions:

Success always.

HL = *pointer to the end of buffer + 1*

Only AF is altered by this SVC.

Example:

See Sample Program B, lines 236-246.

Convert 2 Bytes to Hex ASCII

Converts a 2-byte number to hexadecimal ASCII.

Entry Conditions:

A = 99 (X'63')

DE = *number to convert*

HL = *pointer to 4-character buffer to hold converted number*

Exit Conditions:

Success always.

HL = *pointer to end of buffer + 1*

Only AF is altered by this SVC.

Example:

See Sample Program B, lines 248-258.

Get or Alter HIGH\$ or LOW\$

Provides the means to read or alter the HIGH\$ and LOW\$ values.

Note: HIGH\$ must be greater than LOW\$. LOW\$ is reset to X'2FFF' by @EXIT, @ABORT, and @CMNDI.

Entry Conditions:

A = 100 (X'64')

B selects HIGH\$ or LOW\$

If B = 0, SVC deals with HIGH\$

If B ≠ 0, SVC deals with LOW\$

HL selects one of the following functions:

If HL = 0, the current HIGH\$ or LOW\$ is returned

If HL ≠ 0, then HIGH\$ or LOW\$ is set to the value in HL

Exit Conditions:

Success, Z flag set.

HL = *current HIGH\$ or LOW\$*. If HL ≠ 0 on entry, then HIGH\$ or LOW\$ is now set to that value.

Failure, NZ flag set.

A = *error number*

General:

If bit 0 of CFLAG\$ is set (see @FLAGS), then HIGH\$ cannot be changed with this call. The call returns error 43, "SVC parameter error."

Example:

See Sample Program F, lines 75-86.

Open or Initialize File

Opens a file. If the file is not found, this SVC creates it according to the file specification.

Entry Conditions:

A = 58 (X'3A')

HL = *pointer to 256-byte disk I/O buffer*

DE = *pointer to FCB containing the file specification*

B = *Logical Record Length to be used while file is open*

Exit Conditions:

Success, Z flag set. File was opened or created.

The CF flag is set if a new file was created.

Failure, NZ flag set.

A = *error number*

General:

Only AF is altered by this SVC.

The file open bit is set in the directory if the access level is UPDATE or greater.

Example:

See Sample Program C, lines 260-272.

Reboot the System

Does a software reset. Floppy drive 0 must contain a system disk. @IPL uses the standard boot sequence, the same as for a hard reset (pressing the reset button). Memory locations X'41E5'-X'4225' and X'4300'-X'43FF' are altered during the boot of the machine.

Entry Conditions:

A = 0 (X'00')

General:

This SVC does not return.

Scan Keyboard and Return

Scans the keyboard and returns a character if a key is pressed. If no key is pressed, a zero value is returned.

Entry Conditions:

A = 8 (X'08')

Exit Conditions:

Success, Z flag set.

A = *character pressed*

Failure, NZ set.

If A = 0, no character was available.

If A ≠ 0, then A contains *error number*.

General:

DE is altered by this SVC.

Example:

See Sample Program C, lines 198-200.

Scan *KI Device, Wait for Character

Scans the *KI device and returns with a character. It does not return until a character is input to the device.

Note: The system suspends execution of the program that issued the SVC until a character can be obtained. Background tasks will continue to run normally.

Entry Conditions:

A = 1 (X'01')

Exit Conditions:

Success, Z flag set.

A = *character entered*

Failure, NZ flag set.

A = *error number*

General:

DE is altered by this SVC.

Example:

See Sample Program B, lines 202-203.

Accept a Line of Input

Accepts a line of input until terminated by either an **(ENTER)** or a **(BREAK)**. Entries are displayed on the screen, starting at the current cursor position. Backspace, tab, and line delete are supported. If JCL is active, the line is fetched from the active JCL file.

Entry Conditions:

A = 9 (X'09')
HL = *pointer to user line buffer of length B + 1*
B = *maximum number of characters to input*
C = 0

Exit Conditions:

Success, Z flag set.
HL = *pointer to start of buffer*
B = *actual number of characters input*
CF is set if **(BREAK)** terminated the input.
Failure, NZ flag set.
A = *error number*

General:

DE and C are altered by this SVC.

Example:

See Sample Program C, lines 39-47.

Remove Currently Executing Task

When called by an executing task driver, removes the task assignment from the task table and returns to the foreground application that was interrupted.

Entry Conditions:

A = 32 (X'20')

General:

This SVC does not return.

Example:

See the example for @RMTSK in Sample Program F, lines 134-142.

Load Program File

Loads a program file. The file must be in load module format.

Entry Conditions:

A = 76 (X'4C')

DE = *pointer to FCB containing filespec of the file to load*

Exit Conditions:

Success, Z flag set.

HL = *transfer address retrieved from file*

Failure, NZ flag set.

A = *error number*

Example:

See Sample Program A, lines 50-56.

Calculate Current Logical Record Number

Returns the current logical record number.

Entry Conditions:

A = 63 (X'3F')

DE = pointer to the file's FCB

Exit Conditions:

Success, Z flag set.

BC = *logical record number*

Failure, NZ flag set.

A = *error number*

General:

AF is altered by this SVC.

Example:

See Sample Program C, lines 305-311.

Calculate the EOF Logical Record Number

Returns the EOF (End of File) logical record number.

Entry Conditions:

A = 64 (X'40')

DE = *pointer to FCB for the file to check*

Exit Conditions:

Success, Z flag set.

BC = *the EOF logical record number*

Failure, NZ flag set.

A = *error number*

General:

Only AF is altered by this SVC.

Example:

See the example for @LOC in Sample Program C, lines 305-311.

Issue Log Message

Issues a log message to the Job Log. The message can be any character string terminating with a carriage return (X'0D').

Entry Conditions:

A = 11 (X'0B')

HL = *pointer to first character in message line*

Exit Conditions:

Success, Z flag set.

Failure, NZ flag set.

A = *error number*

General:

Only AF is altered by this SVC.

Example:

```
LD    HL,TEXT    ;Point at message to output
LD    A,@LOGGER  ;and output it to the Job
                     ;Log
RST    28H        ;Call the @LOGGER SVC
...
TEXT:  DEFM 'This is a message for the Job Log'
       DEFB 0DH    ;Message must be terminated
                     ;with an <ENTER>.
```

Display and Log Message

Displays and logs a message. Performs the same function as @DSPLY followed by @LOGGER.

Entry Conditions:

A = 12 (X'0C')

HL = *pointer to first character in message line*

Exit Conditions:

Success, Z flag set.

Failure, NZ flag set.

A = *error number*

General:

Only AF is altered by this SVC.

To avoid a looping condition that could result from the display device generating an error, no error checking should be done after returning from @LOGOT.

Example:

```
LD    HL,TEXT    ;Point at message to output
LD    A,@LOGOT    ;and output it to the Job
                ;Log AND the display
RST    28H        ;Call the @LOGOT SVC
...
TEXT:  DEFM 'This message will be displayed both in'
        DEFM 'the Job Log and on the display.'
        DEFB 0DH    ;Must terminate text with an
                ;<ENTER>.
```

Send Message to Device

Sends a message line to any device or file.

Entry Conditions:

A = 13 (X'0D')

DE = pointer to DCB or FCB of device or file to receive output

HL = pointer to message line terminated with X'0D' or X'03'

Exit Conditions:

Success, Z flag set.

Failure, NZ flag set.

A = error number

General:

Only AF is altered by this SVC.

Example:

```
LD    HL,TEXT    ;Point at message to output
LD    DE,DCBP    ;Point at the device control
                     ;block for our device
LD    A,@MSG     ;and write this text to it
RST   28H        ;Call the @MSG SVC
...
TEXT:  DEFM  'D555-555<LOGIN USER>' ;Text to write to
                     ;this device. In this case,
                     ;it is a dialing modem.
      DEFB  03H    ;Terminate the message
```

8-Bit Multiplication

Performs an 8-bit by 8-bit unsigned integer multiplication. The resultant product must fit into an 8-bit field.

Entry Conditions:

A = 90 (X'5A')

C = multiplicand

E = multiplier

Exit Conditions:

Success always.

A = product

DE is altered by this SVC.

Example:

See Sample Program B, lines 150-153.

16-Bit by 8-Bit Multiplication

Performs an unsigned integer multiplication of a 16-bit multiplicand by an 8-bit multiplier. The resultant product is stored in a 3-byte register field.

Entry Conditions:

A = 91 (X'5B')
HL = *multiplicand*
C = *multiplier*

Exit Conditions:

Success always.
HL = *two high-order bytes of product*
A = *low-order byte of product*
DE is altered by this SVC.

Example:

See Sample Program B, lines 183-187.

Open Existing File or Device

Opens an existing file or device.

Entry Conditions:

A = 59 (X'3B')

HL = *pointer to 256-byte disk I/O buffer*

DE = *pointer to FCB or DCB containing filespec or devspec*

B = *logical record length for open file*

Exit Conditions:

Success, Z flag set.

Failure, NZ flag set.

A = *error number*

General:

AF is altered by this SVC.

The file open bit is set in the directory if the access level is UPDATE or greater.

Example:

See Sample Program C, lines 134-150.

Parse Parameter String

Parses an optional parameter string. Its primary function is to parse command parameters contained in a command line starting with a parenthesis. The acceptable parameter format is:

PARM = X'nnnn'....hexadecimal entry

PARM = nnnnndecimal entry

PARM = "string" ...alphanumeric entry

PARM = flagON, OFF, Y, N, YES, or NO

Note: Entering a parameter with no equal sign or value is the same as using PARM = ON. Entering PARM = with no value is the same as using PARM = OFF.

Entry Conditions:

A = 17 (X'11')

DE = *pointer to beginning of your parameter table*

HL = *pointer to command line to parse* (the parameter string is enclosed within parentheses)

Exit Conditions:

Success always.

If Z is set, either valid parameters or no parameters were found.

If NZ is set, a bad parameter was found.

General:

NZ is not returned if parameter types other than those specified are entered. The application must check the validity of the response byte.

The valid parameters are contained in a user table which must be in one of the following formats. (Parameter names must consist of alphanumeric characters, the first of which is a letter.)

For use with TRSDOS Version 6, use this format:

The parameter table starts with a single byte X'80'. Each parameter is stored in a variable length field as described below.

1) Type Byte (Type and length byte)

Bit 7 — If set, accept numeric value

Bit 6 — If set, accept flag parameter

Bit 5 — If set, accept "string" value

Bit 4 — If set, accept first character of name as abbreviation

Bits 3-0 — Length of parameter name

2) Actual Parameter Name

3) Response byte (Type and length found)

Bit 7 — Numeric value found

Bit 6 — Flag parameter found

Bit 5 — String parameter found

Bits 4-0 — Length of parameter entered. If length is 0 and the 2-byte vector points to a quotation mark (X'22'), then the parameter was a null string. Otherwise, a length of 0 indicates that the parameter was longer than 31 characters.

4) 2-byte address vector to receive the parsed parameter values.

The 2-byte memory area pointed to by the address field of your table receives the value of PARM if PARM is non-string. If a string is entered, the 2-byte memory area receives the address of the first byte of "string." The entries ON, YES, and Y return a value of X'FFFF'; OFF, NO, and N return X'0000'. If a parameter name is specified on the command line and is fol-

lowed by an equal sign and no value, then X'0000' or NO is returned. If a parameter name is used on the command line without the equal sign, then a value of X'FFFF' or ON is assumed. For any allowed parameter that is completely omitted on the command line, the 2-byte area remains unchanged and the response byte is 0.

The parameter table is terminated with a single byte X'00'.

For compatibility with LDOS 5.1.3, use this format:

A 6-character "word" left justified and padded with blanks followed by a 2-byte address to receive the parsed values. Repeat word and address for as many parameters as are necessary. You must place a byte of X'00' at the end of the table.

Example:

```

LD      HL,COMAND ;Point at command buffer
LD      DE,PARM   ;Point at parameter list
LD      A,@PARAM  ;Parse the items on the
                  ;command line
RST     28H       ;Call the @PARAM SVC
JR      NZ,ERROR  ;An error occurred (not
                  ;included here)
LD      A,(RESP)  ;Get response code
AND     040H      ;Test response flags
JR      Z,BAD     ;User specified something
                  ;like UPDATE=X'1234' or
                  ;UPDATE="HELLO"
LD      A,(VAL)   ;Get 1st byte of VAL word
OR      A         ;Test the value
JR      Z,OFF     ;UPDATE=OFF or UPDATE=NO was
                  ;specified
JR      ON        ;UPDATE=ON or UPDATE=YES was
                  ;specified
...
COMAND:  DEFS     80      ;Area where command is
                        ;stored
PARM:    DEFB     80H     ;Table header code
          DEFB     40H+6  ;40 says we want a flag
                        ;(YES/NO). 6 is length of
                        ;the parameter name
RESP:    DEFM     'UPDATE' ;Parameter name
          DEFB     0      ;Response area
          DEFW     VAL    ;Vector to VAL
          DEFB     0      ;End of Table code
VAL:     DEFS     2      ;Area to receive a parameter
                        ;value

```


Suspend Program Execution

Suspends program execution for a specified period of time and goes into a "holding" state. The delay is at least 14.3 microseconds per count.

Entry Conditions:

A = 16 (X'10')
BC = *delay count*

Exit Conditions:

Success always.

Example:

```
LD      BC,36A2H    ;Wait for about 200 milli-
                    ;seconds. 14.3 usecs *
                    ;13986 is approx. 200
                    ;msecs
LD      A,@PAUSE    ;Suspend execution
RST     2BH         ;Call the @PAUSE SVC
```

Position to End Of File

Positions an open file to the End Record Number (ERN). An end-of-file-encountered error (X'1C') is returned if the operation is successful. Your program may ignore this error.

Entry Conditions:

A = 65 (X'41')

DE = *pointer to FCB of the file to position*

Exit Conditions:

NZ flag always set.

If A = X'1C'; then success.

If A ≠ X'1C'; then failure.

A = *error number*

General:

AF is always altered by this SVC.

Example:

See the example for @LOC in Sample Program C, lines 305-311.

Position File

Positions a file to a logical record. This is useful for positioning to records of a random access file.

When the @POSN routine is used, Bit 6 of FCB + 1 is automatically set. This ensures that the EOF (End Of File) is updated when the file is closed only if the NRN (Next Record Number) exceeds the current ERN (End Record Number).

Note that @POSN must be used for *each* write, even if two records are side by side.

Entry Conditions:

A = 66 (X'42')

DE = *pointer to FCB for the file to position*

BC = *the logical record number*

Exit Conditions:

If Z flag is set or A = X'1C' or X'1D', then success.

The file was positioned.

Otherwise, failure.

A = *error number*

General:

AF is always altered by this SVC.

Example:

See the example for @LOC in Sample Program C, lines 305-311.

Prints Message Line

Outputs a message line to the printer. The line must be terminated with either a carriage return (X'0D') or an ETX (X'03').

Entry Conditions:

A = 14 (X'0E')

HL = *pointer to message to be output*

Exit Conditions:

Success, Z flag set.

Failure, NZ flag set.

A = *error number*

General:

AF and DE are altered by this SVC.

Example:

```
LD    HL,TEXT    ;Text to be output to the
                  ;Printer
LD    A,@PRINT    ;Write this message to the
                  ;Printer device
RST    28H        ;Call the @PRINT SVC
...
TEXT: DEFB 0CH     ;Do a Top of Form
      DEFM 'Report continued' Page
      DEFB 3       ;Terminate with a <ETX> or
                  ;an <ENTER>
```

Send Character to Printer

Outputs a byte to the line printer.

Entry Conditions:

A = 6 (X'06')

C = character to print

Exit Conditions:

Success, Z flag set.

Failure, NZ flag set.

A = error number

General:

AF and DE are altered by this SVC.

If the line printer is attached but becomes unavailable (out of paper, out of ribbon, turned off, off-line, buffer full, etc.), the printer driver waits approximately ten seconds. If the printer is still not ready, a "Device not available" error is returned.

Example:

```
LD    A,(PAGE)    ;Get the page number
ADD   A,'0'        ;Make it ASCII
LD    C,A          ;Put the value here
LD    A,@PRT       ;Write this character to the
                   ;Printer
RST   2BH          ;Call the @PRT SVC
...
PAGE: DEFB 2       ;Start with page 2
```

Write One Byte to Device or File

Outputs a byte to a logical device or file. The DCB TYPE byte (DCB + 0, Bit 1) must permit PUT operation.

Entry Conditions:

A = 4 (X'04')

DE = *pointer to DCB or FCB of the output device*

C = *byte to output*

Exit Conditions:

Success, Z flag set.

Failure, NZ flag set.

A = *error number*

General:

AF is always altered by this SVC.

Example:

See the section "Device Driver and Filter Templates."

Get Directory Record or Free Space

Reads the directory information of visible files from a disk directory, or gets the amount of free space on a disk.

Entry Conditions:

A = 35 (X'23')

HL = *pointer to RAM buffer to receive information*

B = *logical drive number (0-7)*

C selects one of the following functions:

If C = 0, get directory records of all visible files.

If C = 255, get free space information.

If C = 1-254, get a single directory record (see below).

Exit Conditions:

Success, Z flag set.

Failure, NZ flag set.

A = *error number*

Each directory record requires 22 bytes of space in the buffer. If C = 0, one additional byte is needed to mark the end of the buffer.

For single directory records, the number in the C register should be one less than the desired directory record. For example, if C = 1, directory record 2 is fetched and put in the buffer. If a single record request is for an inactive record or an invisible file, the A register returns an error code 25 (File access denied).

The directory information is placed in the buffer as follows:

Byte	Contents
00-14	filename/ext:d (left justified, padded with spaces)
15	protection level, 0 to 6
16	EOF offset byte
17	logical record length, 0 to 255
18-19	ERN of file
20-21	file size in K (1024-byte blocks)
22	LAST RECORD ONLY. Contains "+" to mark buffer end.

If C = 255, HL should point to a 4-byte buffer. Upon return, the buffer contains:

Bytes 00-01 Space in use in K, stored LSB, MSB

Bytes 02-03 Space available in K, stored LSB, MSB

Example:

See the example for @DODIR in Sample Program E, lines 32-40.

Read a Sector Header

Reads the next ID header when supported by the controller driver. The floppy disk driver supplied treats this as a @RDSEC (SVC 49).

Entry Conditions:

A = 48 (X'30')
HL = *pointer to buffer to receive the data*
D = *cylinder to read*
C = *logical drive number*
E = *sector to read*

Exit Conditions:

Success, Z flag set.
Failure, NZ flag set.
A = *error number*

Example:

See the example for @RDSEC in Sample Program D, lines 63-66.

Read Sector

Transfers a sector of data from the disk to your buffer.

Entry Conditions:

A = 49 (X'31')
HL = *pointer to the buffer to receive the sector*
D = *cylinder to read*
E = *sector to read*
C = *logical drive number (0-7)*

Exit Conditions:

Success, Z flag set.
Failure, NZ flag set.
A = *error number*

General:

Only AF is altered by this SVC

Example:

See Sample Program D, lines 63-66.

Read System Sector

Reads the specified system (directory) sector. If the cylinder number in register D is not the directory cylinder, the value in D is changed to reflect the real directory cylinder and the sector is then read.

Entry Conditions:

A = 85 (X'55')
HL = *pointer to the buffer to receive the sector*
D = *cylinder to read*
E = *sector to read*
C = *logical drive number (0-7)*

Exit Conditions:

Success, Z flag set.
Failure, NZ flag set.
A = *error number*

General:

Only AF is altered by this SVC.

Example:

See Sample Program D, lines 78-92.

Read a Track

Reads an entire track when supported by the controller driver. The floppy disk driver supplied treats this as a @RDSEC (SVC 49) and does not do a track read.

Entry Conditions:

A = 51 (X'33')
HL = *pointer to buffer to receive the sector*
D = *track to read*
C = *logical drive number*
E = *sector to read*

Exit Conditions:

Success, Z flag set.
Failure, NZ flag set.
A = *error number*

General:

AF is altered by the supplied floppy disk driver.

Example:

See the example for @RDSEC in Sample Program D, lines 63-66.

Read a Record

Reads a logical record from a file. If the LRL defined at open time was 256 (specified by 0), then the NRN sector is transferred to the buffer established at open time. For LRL between 1 and 255, the next logical record is placed into a user record buffer, UREC. The 3-byte NRN is updated after the read operation.

Entry Conditions:

A = 67 (X'43')

DE = *pointer to FCB for the file to read*

HL = *pointer to user record buffer UREC* (needed if LRL = 1-255; unused if LRL = 256)

Exit Conditions:

Success, Z flag set.

Failure, NZ flag set.

A = *error number*

Example:

See Sample Program C, lines 300-304.

Remove File or Device

Removes a file or device.

If a file is to be removed, the File Control Block must be in an open condition. When this SVC is performed, the file's directory is updated and the space occupied by the file is deallocated.

If a device was specified, the device is closed. To remove a device, use the REMOVE library command.

Entry Conditions:

A = 57 (X'39')

DE = *pointer to FCB or DCB to remove*

Exit Conditions:

Success, Z flag set.

Failure, NZ flag set.

A = *error number*

Example:

See Sample Program C, lines 223-231.

Rename File or Device

Changes a file's filename and/or extension.

Entry Conditions:

A = 56 (X'38')

DE = *pointer to an FCB containing the file's current name*

This FCB must be in a closed state.

HL = *pointer to new filename string terminated with a X'0D' or X'03'* This filespec must be in upper case and must be a valid filespec. You can convert the filespec to upper case and check its validity by using the @FSPEC SVC before using @RENAM.

Exit Conditions:

Success, Z flag set.

Failure, NZ flag set.

A = *error number*

General:

After the call is completed, the FCB pointed to by DE is altered.

Only AF is altered by this SVC.

Example:

```
LD    DE,FCB          ;Point at a closed FCB
                        ;containing the old
                        ;filespec
LD    HL,NEW           ;Point to the new filespec
                        ;to use
LD    A,@RENAM         ;Change the name of the
                        ;file
RST   28H              ;Call the @RENAM SVC
...
FCB: DEFS 32           ;A File Control Block used
                        ;by the @RENAM SVC. In
                        ;this example, it is
                        ;assumed that an @FSPEC
                        ;SVC has loaded a filespec
                        ;into the FCB before the
                        ;@RENAM SVC is performed.
NEW: DEFM 'NEWNAME/TXT' ;The new filespec for the
                        ;file
      DEFB 0DH          ;Terminate the filespec
```

Rewind File to Beginning

Rewinds a file to its beginning and resets the 3-byte NRN to 0. The next record to be read or written sequentially is the first record of the file.

Entry Conditions:

A = 68 (X'44')

DE = *pointer to FCB for the file to rewind*

Exit Conditions:

Success, Z flag set. File positioned to record number 0.

Failure, NZ flag set.

A = *error number*

General:

AF is always altered by this SVC.

Example:

See the example for @LOC in Sample Program C, lines 305-311.

Remove Interrupt Level Task

Removes an interrupt level task from the Task Control Block table.

Entry Conditions:

A = 30 (X'1E')

C = *task slot assignment to remove* (0-11)

Exit Conditions:

Success always.

HL and DE are altered by this SVC.

Example:

See Sample Program F, lines 134-142.

Replace Task Vector

Exits the task process executing and replaces the currently executing task's vector address in the Task Control Block table with the address following the SVC instruction. Return is made to the foreground application that was interrupted.

Entry Conditions:

A = 31 (X'1F')

General:

This SVC does not return.

Example:

```
LD      A,RPTSK ;Replace this task with the
               ;one located at the
               ;following address:
RST      28H    ;Call the @RPTSK SVC
NEWADD: DEFW 0   ;Address of the new task is
               ;loaded here. This word
               ;must be immediately after
               ;the @RPTSK SVC. The label
               ;NEWADD is present only to
               ;allow the address to be
               ;stored.
```

Reread Sector

Forces a reread of the current sector to occur before the next I/O request is performed. Its most probable use is in applications that reuse the disk I/O buffer for multiple files, to make sure that the buffer contains the proper file sector. This routine is valid only for byte I/O or blocked files. Do not use it when positioned at the start of a file.

Entry Conditions:

A = 69 (X'45')

DE = *pointer to FCB for the file to reread*

Exit Conditions:

Success, Z flag set.

Failure, NZ flag set.

A = *error number*

General:

AF is always altered by this SVC.

Example:

```
LD    DE,FCB      ;Point to File Control Block
                     ;of the file that requires
                     ;the re-read
LD    A,@RREAD    ;Before next I/O, reload
                     ;the current sector into
                     ;the system buffer for
                     ;this file
RST    28H        ;Call the @RREAD SVC
```

Test for Drive Busy

Performs a test of the last selected drive to see if it is in a busy state. If busy, it is re-selected until it is no longer busy.

Entry Conditions:

A = 47 (X'2F')

C = *logical drive number (0-7)*

Exit Conditions:

Success always.

Only AF is altered by this SVC.

Example:

```
LD    C,1          ;Test Drive 1 to see if it
                   ;is busy.
LD    A,@RSLCT     ;If it is, continue
                   ;selecting it
RST   28H          ;Call the @RSLCT SVC
```

Issue FDC RESTORE Command

Issues a disk controller RESTORE command.

Entry Conditions:

A = 44 (X'2C')

C = *logical drive number (0-7)*

Exit Conditions:

Success, Z flag set.

Failure, NZ flag set.

A = *error number*

Example:

See the example for @CKDRV in Sample Program D, lines 38-39.

Run Program

Loads and executes a program file. If an error occurs during the load, the system prints the appropriate message and returns.

Entry Conditions:

A = 77 (X'4D')

DE = *pointer to FCB containing the filespec of the file to RUN*

Note: The FCB must be located where the program being loaded will not overwrite it.

Exit Conditions:

Success, the new program is loaded and executed.

Failure, the error is displayed and return is made to your program.

HL = *return code* (See the section "Converting to TRSDOS Version 6" for information on return codes.)

General:

HL is returned unchanged if no error occurred and can be used as a pointer to a command line.

Example:

See Sample Program A, lines 62-74.

Rewrite Sector

Rewrites the current sector, following a write operation. The @WRITE function advances the NRN after the sector is written. @RWRIT decrements the NRN and writes the disk buffer again. Do not use @RWRIT when positioned to the start of a file.

Entry Conditions:

A = 70 (X'46')

DE = pointer to FCB for the file to rewrite

Exit Conditions:

Success, Z flag set.

Failure, NZ flag set.

A = error number

Example:

```
LD    DE,FCB      ;Point to the File Control
                      ;Block
LD    A,@RWRIT    ;Perform a re-write of the
                      ;current sector
RST   28H         ;Call the @RWRIT SVC
```

Seek a Cylinder

Seeks a specified cylinder and sector. @SEEK does not return an error if you specified a non-existent drive or an invalid cylinder. @SEEK performs no action if the specified drive is a hard disk.

Note: Seek of a sector is not supported by TRS-80 hardware. An implied seek is included in sector reads and writes.

Entry Conditions:

A = 46 (X'2E')
C = *logical drive number*
D = *cylinder to seek*
E = *sector to seek*

Exit Conditions:

Success always.
Only AF is altered by this SVC.

Seek Cylinder and Sector

Seeks the cylinder and sector corresponding to the next record of the specified file. (This is done by examining the NRN field of the FCB.) No error is returned on physical seek errors.

Entry Conditions:

A = 71 (X'47')

DE = *pointer to the file's FCB*

Exit Conditions:

Success always.

Example:

```
LD    DE,FCB    ;Point to the File Control
                     ;Block
LD    A,@SEEKSC ;Cause the next sector to be
                     ;SEEKed before it is
                     ;actually needed
RST   28H        ;Call the @SEEKSC SVC
```

Skip a Record

Causes a skip past the next logical record. Only the record number contained in the FCB is changed; no physical I/O takes place.

Entry Conditions:

A = 72 (X'48')

DE = *pointer to FCB for the file to skip*

Exit Conditions:

If the Z flag is set or if A = X'1C' or X'1D', then the operation was successful.

Otherwise, A = *error number*. If A = X'1C' is returned, the file pointer is positioned at the end of the file. Any Appending operations would be performed here. If A = X'1D' is returned, the file pointer is positioned beyond the end of the file.

General:

AF is altered by this SVC.

BC contains the current record number. This is the same value as that returned by the @LOC SVC.

Example:

See the example for @LOC in Sample Program C, lines 305-311.

Select a New Drive

Selects a drive. The time delay specified in your configuration (SYSTEM (DELAY = Y/N)) is made if the drive selection requires it.

Entry Conditions:

A = 41 (X'29')

C = *logical drive number (0-7)*

Exit Conditions:

Success, Z flag set.

Failure, NZ flag set.

A = *error number*

General:

Only AF is altered by this SVC.

Sound Generation

Generates sound using specified tone and duration codes. Interrupts are disabled during execution.

Entry Conditions:

A = 104 (X'68')

B = *function code*

bits 0-2: tone selection (0-7 with 0 = highest and 7 = lowest)

bits 3-7: tone duration (0-31 with 0 = shortest and 31 = longest)

Exit Conditions:

Success always.

Only AF is altered by this SVC.

Example:

See Sample Program B, lines 43-45.

Issue FDC STEP IN Command

Issues a disk controller STEP IN command. This moves the drive head to the next higher-numbered cylinder. @STEPI is intended for sequential read/write operations, such as disk formatting.

Entry Conditions:

A = 45 (X'2D')

C = *logical drive number*

Exit Conditions:

Success, Z flag set.

Failure, NZ flag set.

A = *error number*

General:

Only AF is altered by this SVC.

Get Time

Gets the system time in display format (HH:MM:SS).

Entry Conditions:

A = 19 (X'13')

HL = *pointer to buffer to receive the time string*

Exit Conditions:

Success always.

HL = *pointer to the end of buffer + 1*

DE = *pointer to start of TIME\$ storage area in TRSDOS*

AF and BC are altered by this SVC.

Example:

See the example for @DATE in Sample Program F, lines 252-253.

Video Functions

Performs various functions related to the video display. The B register is used to pass the function number.

Entry Conditions:

A = 15 (X'0F')

B selects one of the following functions:

If B = 1, return the character at the screen position specified by HL.

H = row on the screen (0-23), where 0 is the top row

L = column on the screen (0-79), where 0 is the leftmost column

If B = 2, display the specified character at the position specified by HL.

C = character to be displayed

H = row on the screen (0-23), where 0 is the top row

L = column on the screen (0-79), where 0 is the leftmost column

If B = 3, move the cursor to the position specified by HL. This is done even if the cursor is not currently displayed.

H = row on the screen (0-23), where 0 is the top row

L = column on the screen (0-79), where 0 is the leftmost column

If B = 4, return the current position of the cursor.

If B = 5, move a 1920-byte block of data to video memory.

HL = pointer to 1920-byte buffer to move to video memory

If B = 6, move a 1920-byte block of data from video memory to a buffer you supply. In 40 line by 24 character mode, there must be a character in each alternating byte for proper display.

HL = pointer to 1920-byte buffer to store copy of video memory HL must be in the range X'23FF' < HL < X'EC01.

If B = 7, scroll protect the specified number of lines from the top of the screen.

C = number of lines to scroll protect (0-7). Once set, scroll protect can be removed only by executing @VDCTL with B = 7 and C = 0, or by resetting the system. Clearing the screen with **SHIFT CLEAR** erases the data in the scroll protect area, but the scroll protect still exists.

If B = 8, change cursor character to specified character. If the cursor is currently not displayed, the character is accepted anyway and is used as the cursor character when it is turned back on. The default cursor character is an underscore (X'5F') under Version 6.2 and a X'B0' under previous versions.

C = character to use as the cursor character

If B = 9, (under Version 6.2 only) transfer 80 characters to or from the screen.

If C = 0, move characters from the buffer to the screen

If C = 1, move characters from the screen to the buffer

H = row on the screen

DE = pointer to 80 byte buffer

Note: The video RAM area in the Models 4 and 4P is 2048 bytes (2K). The first 1920 bytes can be displayed. The remaining bytes contain the type-ahead buffer and other system buffers.

Exit Conditions:

If B = 1:

Success, Z flag set.

A = character found at the location specified by HL

DE is altered.

Failure, NZ flag set.

A = error number

If B = 2:

Success, Z flag set.

DE is altered.

Failure, NZ flag set.

A = error number

If B = 3:

Success, Z flag set.

DE and HL are altered.

Failure, NZ flag set.

A = error number

If B = 4:

Success always.

HL = row and column position of the cursor. H = row on the screen (0-23), where 0 is the top row; L = column on the screen (0-79), where 0 is the leftmost column.

If B = 5:

Success always.

HL = pointer to the last byte moved to the video + 1

BC and DE are altered.

If B = 6:

Success always.

BC, DE, and HL are altered.

If B = 7:

Success always.

BC and DE are altered.

If B = 8:

Success always.

A = previous cursor character

DE is altered.

If B = 9 (under Version 6.2 only):

Success, Z flag set.

BC, HL, DE are altered.

Failure, NZ flag set because H is out of range.

A = error code 43 (X'2B').

General:

Functions 5, 6, and 7 do not do range checking on the entry parameters.

If HL is not in the valid range in functions 5 and 6, the results may be unpredictable.

Only function 3 (B = 3) moves the cursor.

If C is greater than 7 in function 7, it is treated as modulo 8.

AF and B are altered by this SVC.

Example:

See Sample Program F, lines 304-327.

Write and Verify a Record

Performs a @WRITE operation followed by a test read of the sector (if the write required physical I/O) to verify that it is readable.

If the logical record length is less than 256, then the logical record in the user buffer UREC is transferred to the file. If the LRL is equal to 256, a full sector write is made using the disk I/O buffer identified at file open time.

Entry Conditions:

A = 73 (X'49')

DE = *pointer to FCB for the file to verify*

Exit Conditions:

Success, Z flag set.

HL = *pointer to user buffer containing the logical record*

Failure, NZ flag set.

A = *error number*

General:

Only AF is altered by this SVC.

Example:

See Sample Program C, lines 338-346.

Verify Sector

Verifies a sector without transferring any data from disk.

Entry Conditions:

A = 50 (X'32')
D = *cylinder to verify*
E = *sector to verify*
C = *logical drive number (0-7)*

Exit Conditions:

Success, Z flag set.
Failure, NZ flag set
A = *error number*

General:

AF is always altered by this SVC.
If the sector is a system sector, the sector is readable if an error 6 is returned; any other error number signifies an error has occurred.

Example:

See the example for @WRSEC in Sample Program D, lines 68-76.

Write End Of File

Forces the system to update the directory entry with the current end-of-file information.

Entry Conditions:

A = 74 (X'4A')

DE = *pointer to the FCB for the file to WEOF*

Exit Conditions:

Success, Z flag set.

Failure, NZ flag set.

A = *error number*

General:

AF is always altered by this SVC.

Example:

```
LD    DE,FCB    ;Point at the File Control
                     ;Block
LD    A,@WEOF    ;Force the directory entry
                     ;to be updated now,
                     ;instead of when the file
                     ;is closed
RST   2BH        ;Call the @WEOF SVC
```

Locate Origin of SVC

Used to resolve the relocation address of the calling routine.

Entry Conditions:

A = 7 (X'07')

Exit Conditions:

Success always.

HL = *pointer to address following RST 28H instruction*

AF is always altered by this SVC.

Example:

See Sample Program F, lines 36-60.

Write a Record

Causes a write to the next record identified in the File Control Block.

If the logical record length is less than 256, then the logical record in the user buffer UREC is transferred to the file. If the LRL is equal to 256, a full sector write is made using the disk I/O buffer identified at file open time.

Entry Conditions:

A = 75 (X'4B')

HL = *pointer to user record buffer UREC* (unused if LRL = 256)

DE = *pointer to FCB for the file to write*

Exit Conditions:

Success, Z flag set.

Failure, NZ flag set.

A = *error number*

General:

AF is always altered by this SVC.

Example:

See the example for @VER in Sample Program C, lines 338-346.

Write a Sector

Writes a sector to the disk.

Entry Conditions:

A = 53 (X'35')

HL = *pointer to the buffer containing the sector of data*

D = *cylinder to write*

E = *sector to write*

C = *logical drive number (0-7)*

Exit Conditions:

Success, Z flag set.

Failure, NZ flag set.

A = *error number*

General:

Only AF is altered by this SVC.

Example:

See Sample Program D, lines 68-76.

Write a System Sector

Writes a system sector (used in directory cylinder).

Entry Conditions:

A = 54 (X'36')

HL = *pointer to the buffer containing the sector of data*

D = *cylinder to write*

E = *sector to write*

C = *logical drive number*

Exit Conditions:

Success, Z flag set.

Failure, NZ flag set.

A = *error number*

General:

Only AF is altered by this SVC.

Example:

See Sample Program D, lines 94-104.

Write a Track

Writes an entire track of properly formatted data. The data format must conform to that described in the disk controller's reference manual. @WRTRK must always be preceded by @SLCT.

Entry Conditions:

A = 55 (X'37')
HL = *pointer to format data*
D = *track to write*
C = *logical drive number (0-7)*

Exit Conditions:

Success, Z flag set.
Failure, NZ flag set.
A = *error number*

General:

Only AF is altered by this SVC.

Numerical List of SVCs

Following is a numerical list of the SVCs:

Dec	Hex	Label	Function
0	00	@IPL	Reboot the system
1	01	@KEY	Scan *KI device, wait for character
2	02	@DSP	Display character at cursor, advance cursor
3	03	@GET	Get one byte from a logical device
4	04	@PUT	Write one byte to a logical device
5	05	@CTL	Make a control request to a logical device
6	06	@PRT	Send character to the line printer
7	07	@WHERE	Locate origin of CALL
8	08	@KBD	Scan keyboard and return
9	09	@KEYIN	Accept a line of input
10	0A	@DSPLY	Display a message line
11	0B	@LOGGER	Issue a log message
12	0C	@LOGOT	Display and log a message
13	0D	@MSG	Message line handler
14	0E	@PRINT	Print a message line
15	0F	@VDCTL	Position/locate cursor, get/put character at cursor
16	10	@PAUSE	Suspend program execution
17	11	@PARAM	Parse an optional parameter string
18	12	@DATE	Get system date in the format MM/DD/YY
19	13	@TIME	Get system time in the format HH:MM:SS
20	14	@CHNIO	Pass control to the next module in a device chain
21	15	@ABORT	Load HL with X'FFFF' error and goto @EXIT
22	16	@EXIT	Exit program and return to TRSDOS
23			Reserved for future use
24	18	@CMNDI	Entry to command interpreter with return to the system
25	19	@CMNDR	Entry to command interpreter with return to the user
26	1A	@ERROR	Entry to post an error message
27	1B	@DEBUG	Enter DEBUG
28	1C	@CKTSK	Check if task slot in use
29	1D	@ADTSK	Add an interrupt level task
30	1E	@RMTSK	Remove an interrupt level task
31	1F	@RPTSK	Replace the currently executing task vector
32	20	@KLTSK	Remove the currently executing task
33	21	@CKDRV	Check for drive availability
34	22	@DODIR	Do a directory display/buffer
35	23	@RAMDIR	Get directory record(s) or free space into RAM
36-39			Reserved for future use
40	28	@DCSTAT	Test if drive is assigned in DCT
41	29	@SLCT	Select a new drive
42	2A	@DCINIT	Initialize the FDC
43	2B	@DCRES	Reset the FDC
44	2C	@RSTOR	Issue FDC RESTORE command
45	2D	@STEPI	Issue FDC STEP IN command

Dec	Hex	Label	Function
46	2E	@SEEK	Seek a cylinder
47	2F	@RSLCT	Test if requested drive is busy
48	30	@RDHDR	Read a sector header
49	31	@RDSEC	Read a sector
50	32	@VRSEC	Verify a sector
51	33	@RDTRK	Read a track
52	34	@HDFMT	Hard disk format
53	35	@WRSEC	Write a sector
54	36	@WRSSC	Write a system sector
55	37	@WRTRK	Write a track
56	38	@RENAM	Rename a file
57	39	@REMOV	Remove a file or device
58	3A	@INIT	Open or initialize a file or device
59	3B	@OPEN	Open an existing file or device
60	3C	@CLOSE	Close a file or device
61	3D	@BKSP	Backspace one logical record
62	3E	@CKEOF	Check for end of file
63	3F	@LOC	Calculate the current logical record number
64	40	@LOF	Calculate the EOF logical record number
65	41	@PEOF	Position to the end of file
66	42	@POSN	Position a file to a logical record
67	43	@READ	Read a record from a file
68	44	@REW	Rewind a file to its beginning
69	45	@RREAD	Reread the current sector
70	46	@RWRIT	Rewrite the current sector
71	47	@SEEKSC	Seek a specified cylinder and sector
72	48	@SKIP	Skip the next record
73	49	@VER	Write a record to a file and verify
74	4A	@WEOF	Write end of file
75	4B	@WRITE	Write a record to a file
76	4C	@LOAD	Load a program file
77	4D	@RUN	Load and execute a program file
78	4E	@FSPEC	Fetch a file or device specification
79	4F	@FEXT	Set up a default file extension
80	50	@FNAME	Fetch filename/extension from directory
81	51	@GTDCT	Get Drive Code Table address
82	52	@GTDCB	Find specified or first free DCB
83	53	@GTMOD	Find specified memory module address
84			Reserved for future use
85	55	@RDSSC	Read a system sector
86			Reserved for future use
87	57	@DIRRD	Read directory record
88	58	@DIRWR	Write directory record
89			Reserved for future use
90	5A	@MUL8	Multiply 8-bit unsigned integers
91	5B	@MUL16	Multiply 16-bit by 8-bit unsigned integers
92			Reserved for future use
93	5D	@DIV8	Divide 8-bit unsigned integers
94	5E	@DIV16	Divide 16-bit by 8-bit unsigned integers
95			Reserved for future use
96	60	@DECHEX	Convert decimal ASCII to 16-bit binary value
97	61	@HEXDEC	Convert a number in HL to decimal ASCII

Dec	Hex	Label	Function
98	62	@HEX8	Convert a 1-byte number to hex ASCII
99	63	@HEX16	Convert a 2-byte number to hex ASCII
100	64	@HIGH\$	Obtain or set the highest and lowest unused RAM addresses
101	65	@FLAGS	Point IY to the system flag table
102	66	@BANK	Check, set, or reset a 32K bank of memory
103	67	@BREAK	Set user or system break vector
104	68	@SOUND	Generate sound (tone and duration)
105-127			Reserved for future use.

Alphabetical List of SVCs

Following is an alphabetical list of the SVC labels and numbers:

Label	Dec	Hex
@ABORT	21	15
@ADTSK	29	1D
@BANK	102	66
@BKSP	61	3D
@BREAK	103	67
@CHNIO	20	14
@CKDRV	33	21
@CKEOF	62	3E
@CKTSK	28	1C
@CLOSE	60	3C
@CMNDI	24	18
@CMNDR	25	19
@CTL	5	5
@DATE	18	12
@DCINIT	42	2A
@DCRES	43	2B
@DCSTAT	40	28
@DEBUG	27	1B
@DECHEX	96	60
@DIRRD	87	57
@DIRWR	88	58
@DIV8	93	5D
@DIV16	94	5E
@DODIR	34	22
@DSP	2	2
@DSPLY	10	0A
@ERROR	26	1A
@EXIT	22	16
@FEXT	79	4F
@FLAGS	101	65
@FNAME	80	50
@FSPEC	78	4E
@GET	3	3
@GTDCB	82	52
@GTDCT	81	51
@GTMOD	83	53
@HDFMT	52	34
@HEXDEC	97	61
@HEX8	98	62
@HEX16	99	63
@HIGH\$	100	64
@INIT	58	3A
@IPL	0	0
@KBD	8	8
@KEY	1	1
@KEYIN	9	9
@KLTSK	32	20
@LOAD	76	4C
@LOC	63	3F
@LOF	64	40
@LOGGER	11	0B
@LOGOT	12	0C
@MSG	13	0D

Label	Dec	Hex
@MUL8	90	5A
@MUL16	91	5B
@OPEN	59	3B
@PARAM	17	11
@PAUSE	16	10
@PEOF	65	41
@POSN	66	42
@PRINT	14	0E
@PRT	6	6
@PUT	4	4
@RAMDIR	35	23
@RDHDR	48	30
@RDSEC	49	31
@RDSSC	85	55
@RDTRK	51	33
@READ	67	43
@REMOV	57	39
@RENAM	56	38
@REW	68	44
@RMTSK	30	1E
@RPTSK	31	1F
@RREAD	69	45
@RSLCT	47	2F
@RSTOR	44	2C
@RUN	77	4D
@RWRT	70	46
@SEEK	46	2E
@SEEKSC	71	47
@SKIP	72	48
@SLCT	41	29
@SOUND	104	68
@STPI	45	2D
@TIME	19	13
@VDCTL	15	0F
@VER	73	49
@VRSEC	50	32
@WEOF	74	4A
@WHERE	7	7
@WRITE	75	4B
@WRSEC	53	35
@WRSSC	54	36
@WRTRK	55	37

Sample Programs

The following sample programs use many of the supervisor calls described in this manual. These programs are not meant to be examples of the most efficient programming, but are designed to illustrate as many supervisor calls as possible.

Sample Program A

Ln #	Source Line
00001	; This program asks the user whether to run a program
00002	; or debug it and executes the SVCs required to perform
00003	; the chosen action.
00004	
00005	PSECT 5000H ;The program begins at x'5000'
00007	
00008	; Define the equates for the SVCs that will be used.
00009	
00010	@DEBUG: EQU 27 ;Enter the debugger (DEBUG)
00011	@DSPLY: EQU 10 ;Display a message
00012	@FSPEC: EQU 78 ;Verify a filespec or devspec and
00013	;load it into a File Control Block
00014	@KEY: EQU 1 ;Get a character from the keyboard
00015	@LOAD: EQU 76 ;Load a program into memory
00016	@RUN: EQU 77 ;Execute a program
00017	
00018	MESS1: DEFM 'Do you wish to RUN this Program or DEBUG it ?'
00019	DEFB 0AH ;This moves the cursor to the next line
00020	DEFM 'Press <ENTER> to RUN or <BREAK> to DEBUG'
00021	DEFB 0DH ;Terminate the message string
00022	
00023	PROGRM: DEFM 'DIREX/CMD' ;Sample program to debug or execute
00024	DEFB 0DH ;Terminate the filespec
00025	
00026	FCB1: DEFS 32 ;File Control Block for the program
00027	
00028	; Get the File Control Block for the program 'DIREX/CMD'.
00029	
00030	START: LD HL,PROGRM ;Point at the filespec we want to
00031	;execute or load into memory
00032	LD DE,FCB1 ;Point at the File Control Block
00033	LD A,@FSPEC ;Perform a validity check on the filespec
00034	;and copy the filespec into the FCB.
00035	RST 28H ;Call the @FSPEC svc
00036	
00037	LD HL,MESS1 ;Point at our prompting message
00038	LD A,@DSPLY ;and print it on the display
00039	RST 28H ;Call the @DSPLY svc
00040	
00041	LD A,@KEY ;Get the reply from the keyboard
00042	RST 28H ;Call the @KEY svc
00043	
00044	CP 0DH ;Was the character an <ENTER>?
00045	JR Z,RUNIT ;If Z was set, then run the program
00046	
00047	; If it wasn't an <ENTER>, then we assume it was a <BREAK> and
00048	; load the program and enter the debugger.
00049	
00050	LD DE,FCB1 ;Point at the File Control Block
00051	LD A,@LOAD ;and have this program loaded into memory
00052	RST 28H ;Call the @LOAD svc
00053	
00054	; Note that this program must not be overwritten by the program
00055	; we are loading. In this example, it is known that the program
00056	; we are loading starts at x'3000' and ends below x'5000'.
00057	
00058	LD A,@DEBUG ;Now invoke the system debugger, DEBUG
00059	RST 28H ;Call the @DEBUG svc
00060	;Note that @DEBUG does not return
00061	
00062	; Execute the program
00063	
00064	RUNIT: LD DE,FCB1 ;Point at the File Control Block
00065	LD A,@RUN ;Tell TRSDOS to load and execute the
00066	;program
00067	RST 28H ;Call the @RUN svc

Sample Program A, continued

```
00068                                ;Note that @RUN returns only if it can't
00069                                ;find the program
00070
00071 ;   Note that the program that is loaded by the @RUN svc must not
00072 ;   overwrite the File Control Block in this program.  In this case,
00073 ;   it is known that the program we are executing starts at x'3000'
00074 ;   and ends below the starting point of this program, x'5000'.
00075
00076     END      START
```

Sample Program B

```

00001 ;This program accepts numbers from the keyboard
00002 ;and uses them to demonstrate the
00003 ;arithmetic and numeric conversion SVCs.
00004
00005 ;It also uses the sound function to produce a tone at the
00006 ;beginning of the program.
00007
00008         PSECT    3000H
00009
00010 ;
00011 ;       These are the SVCs used in this program.
00012
00013 @DECHEX:EQU    96                ;Convert decimal ASCII to binary
00014 @DIV8: EQU     93                ;Perform 8-bit division
00015 @DIV16: EQU    94                ;Perform 16-bit division
00016 @DSP: EQU      2                ;Display a character
00017 @DSPLY: EQU    10               ;Display a message
00018 @EXIT: EQU     22               ;Return to TRSDOS Ready or the caller
00019 @HEX8: EQU     98                ;Convert an 8-bit value to hex ASCII
00020 @HEX16: EQU    99                ;Convert a 16-bit value to hex ASCII
00021 @HEXDEC:EQU    97                ;Convert a binary value to Decimal ASCII
00022 @KEY: EQU       1                ;Read a character from *KI
00023 @KEYIN: EQU     9                ;Accept an input line from *KI
00024 @MUL8: EQU     90                ;Perform 8-bit multiplication
00025 @MUL16: EQU    91                ;Perform 16-bit multiplication
00026 @SOUND: EQU    104              ;Produce a tone
00027
00028 ;       Other equates.
00029
00030 NUM5: EQU       5
00031 NUM4: EQU       4
00032 NUM3: EQU       3
00033 NUM2: EQU       2
00034 NUM1: EQU       1
00035 BRK: EQU       80H              ;Character code for <BREAK> key
00036 CCC: EQU       0DH              ;Next line position
00037
00038
00039 ;Perform a subroutine 2 times to display prompting messages, key in
00040 ;and display divisor and dividend, convert those numbers to
00041 ;binary for the divide, and position the cursor.
00042
00043 START: LD       B,5AH            ;Make the longest, highest tone
00044        LD       A,@SOUND        ;Make the noise
00045        RST      28H
00046        CALL     KEYIN            ;Perform keyin subroutine for dividend
00047        LD       A,C
00048        LD       (DIVD1),A        ;Store the dividend in memory
00049        LD       HL,MESS9         ;Address of hex message
00050        CALL     DSPLY            ;Display hex message
00051        LD       A,(DIVD1)        ;Get the divisor into C for conversion
00052        LD       C,A              ;from binary to hex
00053        CALL     HEX8             ;Convert the number to hex
00054        CALL     KEYIN            ;Perform subroutine for divisor
00055        LD       A,C
00056        LD       (DIVR1),A        ;Store the divisor in memory
00057
00058 ;Now we are ready to perform the divide on the numbers entered.
00059
00060        LD       C,A              ;Put the divisor back for the @DIV8 SVC
00061        LD       A,(DIVD1)        ;Get the dividend into E
00062        LD       E,A              ;for the @DIV8 SVC
00063        LD       A,@DIV8          ;Call the @DIV8 SVC
00064        RST      28H
00065
00066 ;Now display the answer and the remainder in decimal.
00067
00068        LD       (ANS1),A          ;Store the answer in memory

```


Sample Program B, continued

```

00069      LD      A,E           ;Get the remainder
00070      LD      (REM1),A       ;Store the remainder in memory
00071      LD      HL,MESS3       ;Load address of answer message
00072      CALL     DSPLAY        ;Display the message
00073      LD      A,(ANS1)        ;Get the answer into L for conversion
00074      LD      L,A            ;Number to convert
00075      LD      H,0             ;Put a 0 in the MSB
00076      CALL     HEXDEC        ;Perform subroutine to display decimal value
00077      LD      HL,MESS4        ;Address of remainder message
00078      CALL     DSPLAY        ;Display remainder message
00079      LD      A,(REM1)        ;Put remainder in A for hex conversion
00080      LD      L,A            ;Number to convert
00081      LD      H,0             ;Put 0 in the MSB
00082      CALL     HEXDEC        ;Display decimal value
00083
00084      ;Now divide with a 16-bit dividend.
00085
00086      LD      HL,MESS6        ;Address of 2nd dividend message
00087      CALL     DSPLAY        ;Display next message
00088      LD      A,@KEYIN        ;Key in up to 5 digits
00089      LD      HL,BUF6         ;Store the number
00090      LD      B,NUM5          ;Maximum length of number
00091      LD      C,0
00092      RST      28H
00093      LD      A,@DECHEX       ;Convert the number to binary
00094      RST      28H
00095      LD      (DIVD2),BC      ;Store the dividend
00096      LD      HL,MESS9        ;Address of hex message
00097      CALL     DSPLAY        ;Display hex message
00098      LD      DE,(DIVD2)      ;Put dividend into DE for conversion
00099      CALL     HEX16          ;Convert the number from binary to hex
00100      CALL     KEYIN          ;Key in divisor
00101      LD      A,C             ;Put the divisor into A
00102      LD      (DIVR1),A       ;Store the divisor in memory
00103      LD      HL,MESS3        ;Address of answer message
00104      CALL     DSPLAY        ;Display the message
00105      LD      HL,(DIVD2)      ;Put dividend into HL
00106      LD      A,(DIVR1)       ;Get divisor into C
00107      LD      C,A
00108      LD      A,@DIV16
00109      RST      28H
00110      LD      (REM1),A        ;Store the remainder
00111      LD      (ANS2),HL       ;Put the answer into HL
00112      CALL     HEXDEC        ;Display answer in decimal
00113      LD      HL,MESS4        ;Address of remainder message
00114      CALL     DSPLAY        ;Display remainder message
00115      LD      A,(REM1)        ;Get the remainder
00116      LD      L,A            ;into L
00117      LD      H,0             ;Put a 0 in MSB
00118      CALL     HEXDEC        ;Convert the remainder to decimal
00119
00120      ;Now try some multiplication of 8 bits.
00121
00122      LD      HL,MESS8        ;Address of MUL8 message
00123      CALL     DSPLAY        ;Display first multiplicand message
00124      LD      A,@KEYIN        ;Key in a 2-digit number
00125      LD      HL,BUF2         ;Put it here
00126      LD      B,NUM2          ;Maximum number of characters
00127      LD      C,0
00128      RST      28H
00129      LD      A,@DECHEX       ;Convert the number to binary for math
00130      RST      28H
00131      LD      (MCAND1),BC     ;Store the multiplicand
00132      LD      HL,MESS10       ;Address of MUL8 multiplier message
00133      CALL     DSPLAY        ;Display first multiplier message
00134      LD      A,@KEYIN        ;Key in the multiplier
00135      LD      HL,BUF2         ;Put it here

```

Sample Program B, continued

```

00136      LD      B,NUM1      ;Maximum number of characters
00137      LD      C,0
00138      RST      28H
00139      LD      A,@DECHX      ;Convert the multiplier to binary for math
00140      RST      28H
00141      LD      (MIER1),BC      ;Store multiplier in memory
00142      LD      HL,MESS13      ;Address of multiplier message
00143      LD      A,@DSPLY      ;Display multiplier message
00144      RST      28H
00145
00146      ;Now multiply the two numbers just entered.
00147
00148      LD      A,(MCAND1)      ;Get the multiplicand into C
00149      LD      C,A
00150      LD      A,(MIER1)      ;Get the multiplier into E
00151      LD      E,A
00152      LD      A,@MUL8
00153      RST      28H
00154      LD      L,A      ;Put the product into L
00155      LD      H,0      ;Put 0 in the MSB
00156      CALL    HEXDEC      ;Convert the product to decimal
00157
00158      ;Now multiply a 16-bit by an 8-bit.
00159
00160      LD      HL,MESS11      ;Address of multiplicand message
00161      CALL    DSPLY      ;Display 2nd multiplicand message
00162      LD      A,@KEYIN      ;Enter larger multiplicand
00163      LD      HL,BUF5      ;Put it here
00164      LD      B,NUM4      ;Maximum number of characters
00165      LD      C,0
00166      RST      28H
00167      LD      A,@DECHX      ;Convert the number to binary for math
00168      RST      28H
00169      LD      (MCAND2),BC      ;Store the multiplicand in memory
00170      LD      HL,MESS12      ;Address of multiplier message
00171      CALL    DSPLY      ;Display message
00172      LD      A,@KEYIN      ;Enter larger multiplier
00173      LD      HL,BUF3      ;Put it here
00174      LD      B,NUM2      ;Maximum number of characters
00175      LD      C,0
00176      RST      28H
00177      LD      A,@DECHX      ;Convert the number to binary for math
00178      RST      28H
00179      LD      (MIER1),BC      ;Store the multiplier in memory
00180      LD      HL,MESS13      ;Address of product message
00181      LD      A,@DSPLY      ;Display the message
00182      RST      28H
00183      LD      HL,(MCAND2)      ;Put multiplicand into HL
00184      LD      A,(MIER1)      ;Get the multiplier into C
00185      LD      C,A
00186      LD      A,@MUL16      ;Multiply the two numbers
00187      RST      28H
00188      LD      H,L      ;Get the 2nd byte of the product into
00189      ;H for conversion
00190      LD      L,A      ;Get the LSB into L for conversion
00191      LD      DE,BUF5      ;Convert the high-order byte to decimal
00192      LD      A,@HEXDEC      ;for the display
00193      RST      28H
00194      LD      A,CCC      ;Tell the display when to stop
00195      LD      (DE),A
00196      LD      HL,BUF5
00197      LD      A,@DSPLY      ;Display the product
00198      RST      28H
00199      LD      HL,MESS14      ;Address of end message
00200      LD      A,@DSPLY      ;Display end message
00201      RST      28H
00202      LD      A,@KEY      ;Allow the user to enter any character
00203      RST      28H      ;or hit <BREAK>

```

Sample Program B, continued

```

00204      CP      BRK      ;Is it <BREAK>?
00205      JP      NZ,START ;Yes, go back to beginning
00206      LD      A,@EXIT  ;No, exit the program
00207      RST      28H
00208
00209      ;These are the subroutines used by the calls to
00210      ;display a message, key in a 3-digit number, and convert it
00211      ;from decimal to binary.
00212
00213 KEYIN:  LD      HL,MESS1
00214      CALL     DSPLY      ;Display message
00215      LD      HL,BUF4     ;Put the number here
00216      LD      B,NUM3      ;Maximum number of characters
00217      LD      C,0
00218      LD      A,@KEYIN    ;Key in a number
00219      RST      28H
00220      LD      A,@DECHEX   ;Convert the number to binary
00221      RST      28H
00222      RET              ;Return to next sequential instruction
00223
00224      ;Display what was loaded into HL before the call.
00225
00226 DSPLY:  LD      A,@DSPLY  ;@DISPLAY SVC
00227      RST      28H
00228      DEC      HL          ;Set HL back to blank byte
00229      LD      B,(HL)       ;Load B with the number of bytes
00230      DSPLYLP: LD      C,' ' ;Put a blank into C
00231      LD      A,@DSP       ;Display the blank
00232      RST      28H         ;until the correct number
00233      DJNZ     DSPLYLP     ;of blanks have been displayed
00234      RET              ;Return to next instruction
00235
00236      ;Convert 1 byte to hexadecimal.
00237
00238 HEX8:   LD      A,@HEX8   ;Convert 1 byte to hex ASCII
00239      LD      HL,BUF3      ;Put the converted value here
00240      RST      28H
00241      LD      A,CCC        ;Tell display when to stop
00242      LD      (HL),A       ;Put CCC at end of buffer
00243      LD      A,@DSPLY     ;Display the hex value
00244      LD      HL,BUF3
00245      RST      28H
00246      RET              ;Return to next instruction
00247
00248      ;Convert 2 bytes to hexadecimal.
00249
00250 HEX16:  LD      A,@HEX16  ;Convert a 2-byte number to hex ASCII
00251      LD      HL,BUF6      ;Put the converted value here
00252      RST      28H
00253      LD      A,CCC        ;CCC at end of buffer so display
00254      LD      (HL),A       ;knows when to stop
00255      LD      A,@DSPLY     ;Display the converted value
00256      LD      HL,BUF6      ;Address of converted value
00257      RST      28H
00258      RET              ;Return to next instruction
00259
00260      ;Convert from binary to decimal and display decimal value.
00261
00262 HEXDEC: LD      A,@HEXDEC ;Convert from binary to decimal
00263      LD      DE,BUF5      ;Put converted value here
00264      RST      28H
00265      LD      A,CCC        ;CCC at end of buffer so display
00266      LD      (DE),A       ;knows when to stop
00267      LD      A,@DSPLY     ;Display the hex value
00268      LD      HL,BUF5      ;It's here
00269      RST      28H
00270      RET              ;Return to next instruction
00271

```

Sample Program B, continued

```

00272 ;These are the storage declarations.
00273
00274 BUF6:  DEFS    6
00275 BUF5:  DEFS    5
00276 BUF4:  DEFS    4
00277 BUF3:  DEFS    3
00278 BUF2:  DEFS    2
00279 DIVR1:  DEFB    0
00280 DIVD1:  DEFB    0
00281 ANS1:   DEFB    0
00282 REM1:   DEFB    0
00283 MCAND1: DEFB    0
00284 MIER1:  DEFB    0
00285 MCAND2: DEFW    0
00286 DIVD2:  DEFW    0
00287 ANS2:   DEFW    0
00288
00289 ;Below are messages and prompting text used in the program.
00290
00291                DEFB    13                ;Number of blanks to print after message 1
00292 MESS1:  DEFM    'Enter a number (1-255).'
00293                DEFB    3                ;Message-terminating character
00294                DEFB    21                ;Number of blanks to print after message 3
00295 MESS3:  DEFM    'The answer is'
00296                DEFB    3                ;Terminating character
00297                DEFB    18                ;Blanks after message
00298 MESS4:  DEFM    'The remainder is'
00299                DEFB    3                ;Terminating character
00300                DEFB    6                ;Blanks after message
00301 MESS6:  DEFM    'Enter a number (4369-65535).'
00302                DEFB    3                ;Terminating character
00303                DEFB    15                ;Blanks after message
00304 MESS8:  DEFM    'Enter a number (1-28).'
00305                DEFB    3                ;Terminating character
00306                DEFB    16                ;Blanks after message
00307 MESS9:  DEFM    'In hex ASCII, that is'
00308                DEFB    3                ;Terminating character
00309                DEFB    17                ;Blanks after message
00310 MESS10: DEFM    'Enter a number (1-9).'
00311                DEFB    3                ;Terminating character
00312                DEFB    11                ;Blanks after message
00313 MESS11: DEFM    'Enter a number (1-4100).'
00314                DEFB    3                ;Terminating character
00315                DEFB    15                ;Blanks after message
00316 MESS12: DEFM    'Enter a number (1-15).'
00317                DEFB    3                ;Terminating character
00318 MESS13: DEFM    'The product of those 2 numbers is '
00319                DEFB    3                ;Terminating character
00320 MESS14: DEFM    'Press <BREAK> to end or any other key to continue.'
00321                DEFB    0DH                ;Terminating character
00322
00323                END        START

```

Sample Program C

```

Ln #      Source'Line -4
000001 ;      This program prompts for two filenames, opens the first
000002 ;      file, and creates the second. Then the data in the first
000003 ;      file is copied to the second file. While the Copy progresses,
000004 ;      the current record number is displayed in parentheses.
000005
000006 PSECT 30000H ;This program starts at x'30000'
000008
000009 ;      First, declare the equates for the SVCs we intend to use.
000010 ;      This is not mandatory, but it makes the program easier to follow.
000011
000012 @CLOSE: EQU 60 ;Close a file or device
000013 @DIRRD: EQU 87 ;Read a directory record
000014 @DSP: EQU 2 ;Display character at cursor
000015 @DSPLY: EQU 10 ;Display a message
000016 @ERROR: EQU 26 ;Display an error message
000017 @EXIT: EQU 22 ;Exit and return to TRSDOS or the caller
000018 @FEXT: EQU 79 ;Add a default file extension
000019 @FNAME: EQU 80 ;Fetch a filespec from the directory
000020 @FSPEC: EQU 78 ;Verify and load a filespec into the FCB
000021 @HEXDEC: EQU 97 ;Convert a binary value to decimal ASCII
000022 @INIT: EQU 58 ;Open an existing file or create a new file
000023 @KBD: EQU 8 ;Scan the keyboard for a character
000024 @KEYIN: EQU 9 ;Accept a line of text from the *KI device
000025 @LOC: EQU 63 ;Return the current logical record number
000026 @OPEN: EQU 59 ;Open an existing file
000027 @READ: EQU 67 ;Read a record from an open file
000028 @REMOV: EQU 57 ;Delete a file from disk
000029 @VER: EQU 73 ;Write a record to disk. Does the same thing
000030 ;as @WRITE (Svc 75), but it also makes sure
000031 ;the written data is readable.
000032
000033 ;      First, prompt for the source filespec using the @DSPLY svc.
000034
000035 BEGIN: LD HL,MESG1 ;Get the first message
000036 LD A,@DSPLY ;Display a line on the screen
000037 RST 28H ;Call the @DSPLY svc
000038
000039 ;      Now, read the filename from the keyboard using the @KEYIN svc.
000040
000041 LD HL,FILE1 ;Put the name of the 1st file here
000042 LD B,24 ;Allow up to 24 characters
000043 LD C,0 ;A zero is required by the svc
000044 LD A,@KEYIN ;Get a filename from the user
000045 RST 28H ;Call the @KEYIN svc
000046 JP C,QUIT ;The user pressed <Break>
000047 JP NZ,ERR ;An Error occurred
000048
000049 LD A,B ;Get the number of characters
000050 OR A ;See if that value was zero
000051 JR Z,BEGIN ;Nothing was entered, ask again
000052
000053 ;      The user has typed something, so it must be checked for validity
000054 ;      using the @FSPEC svc.
000055
000056 LD HL,FILE1 ;Point at the text the user entered
000057 LD DE,FCB1 ;Point at the File Control Block
000058 ;that is to be used for the source file.
000059 LD A,@FSPEC ;The @FSPEC svc will make sure the filename
000060 ;that is in buffer named "file1" is valid.
000061 ;If it is, it is copied into the File
000062 ;Control Block (FCB) to be used by the @OPEN
000063 ;or @INIT svc later on.
000064 RST 28H ;Call the @FSPEC svc
000065 JR Z,ASK2 ;The name for file 1 is ok, so skip this
000066
000067 ;      At this point the filename specified for file 1 has been found

```

Sample Program C, continued

```

00068 ; to be in an invalid format. The following code will print the
00069 ; error message.
00070
00071 LD HL,BADFIL ;Point at the bad filename message
00072 LD A,@DSPLY ;Display it
00073 RST 28H ;Call the @DSPLY svc
00074 JR BEGIN ;Start over
00075
00076 ; At this point, the source filename appears to be valid.
00077 ; The code below asks for the second filename and checks it for
00078 ; validity also.
00079
00080 ASK2: LD HL,MESG2 ;Prompt for the target filename
00081 LD A,@DSPLY ;Print that on the screen
00082 RST 28H ;Call the @DSPLY svc
00083 LD HL,FILE2 ;Put the name of the 2nd file here
00084 LD B,24 ;Allow up to 24 characters
00085 LD C,0 ;A zero is required by the svc
00086 LD A,@KEYIN ;Get a filename from the user
00087 RST 28H ;Call the @KEYIN svc
00088 JP C,QUIT ;The user pressed <Break>
00089 JP NZ,ERR ;An Error occurred
00090
00091 LD A,B ;Get the number of characters
00092 OR A ;See if that value was zero.
00093 JR Z,ASK2 ;Nothing was entered, ask again
00094
00095 ; The user has typed something, so it must be checked for validity
00096 ; using the @FSPEC svc.
00097
00098 LD HL,FILE2 ;Point at the text the user entered
00099 LD DE,FCB2 ;Point at the File Control Block
00100 LD A,@FSPEC ;Check the name for validity
00101 RST 28H ;Call the @FSPEC svc
00102 JR Z,F2OK ;The name for file 2 is ok, so skip this
00103
00104 ; The name for file 2 is invalid so print an error message
00105
00106 LD HL,BADFIL ;Point at the bad filename message
00107 LD A,@DSPLY ;Display it
00108 RST 28H ;Call the @DSPLY svc
00109 JR BEGIN ;Start over
00110
00111 ; Now we will attempt to add an extension to the target file
00112 ; if the user did not specify one. We use the extension that
00113 ; was specified on the source file. If it does
00114 ; not have one, then we will not try to add one to the target file.
00115
00116 F2OK: LD HL,FCB1+1 ;Point at the source filename
00117 ;We start with the second character since
00118 ;the filename must be at least one character
00119 FDIV: LD A,(HL) ;Get a character from the filespec
00120 CP '/' ;Is the character the extension prefix?
00121 JR Z,EXTN ;Yes, this will be our default extension
00122 CP 0DH ;Have we reached the end of the filespec?
00123 JR Z,NOEXT ;Yes, there is no extension so don't add one
00124 CP 03H ;Test both terminators
00125 JR Z,NOEXT
00126 INC HL ;Advance the pointer to the next character
00127 JR FDIV ;Keep looking
00128
00129 EXTN: INC HL ;Advance pointer to first byte of extension
00130 LD DE,FCB2 ;Point at FCB for the target file (file 2)
00131 LD A,@FEXT ;Add an extension if one is not present
00132 RST 28H ;Call the @FEXT svc
00133
00134 ; Now we have two filenames. First we will open the source file
00135 ; to make sure it exists.

```

Sample Program C, continued

```

00136
00137 NOEXT: LD      DE,FCB1      ;Point at the File Control Block for file1
00138      LD      HL,BUF1      ;Point at the system buffer. This buffer
00139                                ;is used by the system to block data that
00140                                ;is written to disk and de-block data that
00141                                ;is read from disk when the Logical Record
00142                                ;Length of the file is not 256. If it is
00143                                ;256, then this buffer is not used.
00144      LD      B,0           ;Use LRL 256 for now since we don't know
00145                                ;what to use yet.
00146      LD      A,@OPEN      ;Open the file
00147      RST     28H          ;Call the @OPEN svc
00148      JR      Z,SIZ        ;The file opened and is LRL 256.
00149      CP      42           ;Was the error a LRL Open Fault?
00150      JP      NZ,ERR        ;No, perhaps the file does not exist.
00151
00152 ;      At this point, the file is open and we can now examine the
00153 ;      directory to find out what LRL it was created with so we can
00154 ;      use that value to make the copy.
00155
00156 SIZ:  LD      A,(FCB1+6)    ;Get the byte in the FCB which contains
00157                                ;the drive number the file is on
00158      AND     7             ;Erase all other information in that byte
00159      LD      C,A           ;Save that value here
00160      LD      A,(FCB1+7)    ;This reads the Directory Entry Code (DEC)
00161                                ;out of the FCB so we can use it
00162      LD      B,A           ;Store the DEC here
00163      PUSH    BC            ;Save that value for now
00164      LD      A,@CLOSE      ;We can close the source file for now
00165      RST     28H          ;Call the @CLOSE svc
00166
00167      POP     BC            ;Get the DEC value back off the stack
00168      LD      A,@DIRRD      ;Read the directory record for that file
00169      RST     28H          ;Call the @DIRRD svc
00170
00171      LD      IX,HL          ;Put the pointer to the directory record
00172      LD      A,(IX+4)      ;here and read the DIR+4 entry which
00173                                ;contains the LRL of the source file.
00174      LD      (LRL),A       ;Save that value
00175
00176 ;      Before we go any further, we should check to see if the target file
00177 ;      already exists.
00178
00179      LD      DE,COPY       ;First, make a copy of the FCB
00180      LD      HL,FCB2       ;in case we have to delete a file
00181      LD      BC,32         ;Move the entire block
00182      LDIR
00183
00184      LD      DE,FCB2       ;Point at the target File Control Block
00185      LD      HL,BUF2       ;Use this as the buffer for now
00186      LD      B,0           ;Use LRL 256 for now
00187      LD      A,@OPEN      ;Open it and see if it is there
00188      RST     28H          ;Call the @OPEN svc
00189      JR      Z,EXISTS      ;The file already exists, better ask
00190      CP      42           ;Was the error a LRL mismatch?
00191      JR      NZ,NOFILE     ;No, so the file does not exist.
00192
00193 EXISTS: LD      HL,FEXST    ;Point at a prompt asking if it is ok
00194                                ;to erase the file that already exists
00195      LD      A,@DSPLY      ;Print that message
00196      RST     28H          ;Call the @DSPLY svc
00197
00198 WAIT:  LD      A,@KBD       ;Wait for the user to type Y or N
00199      RST     28H          ;Call the @KBD svc
00200      JR      NZ,WAIT       ;Loop until something is typed
00201
00202      CP      'Y'          ;Was a 'Y' typed?
00203      JR      Z,KILLIT      ;Then kill the file

```

Sample Program C, continued

```

002004      CP      'y'                ;Check for lowercase too
002005      JR      Z,KILLIT
002006      CP      'N'                ;Do they want to leave the file alone?
002007      JR      Z,SHUT              ;No, just close the file and quit
002008      CP      'n'                ;Was it a lowercase 'N'?
002009      JR      NZ,WAIT             ;No, loop until we see something we like
002010
002011 SHUT:   LD      DE,FCB2          ;Close the target file
002012      LD      A,@CLOSE
002013      RST     28H                ;Call the @CLOSE svc
002014      JP      QUIT               ;Exit to TRSDOS
002015
002016 ;      At this point, we have been given the OK to delete the file
002017 ;      that has the same name as the target file.
002018
002019 KILLIT: LD      C,0DH             ;First move display to a new line
002020      LD      A,@DSP
002021      RST     28H                ;Display an <Enter>
002022      RST     28H                ;Call the @DSP svc
002023
002024      LD      DE,FCB2             ;Point at the target file's FCB
002025      LD      A,@REMOV
002026      RST     28H                ;Delete the file from disk
002027      RST     28H                ;Call the @REMOV svc. (This is the same
002028 ;      as the @KILL call on other TRSDOS systems.)
002029      JP      NZ,ERR              ;An error occurred, print it and quit
002030 ;      Note that after a @REMOV succeeds,
002031 ;      the filespec is removed from the FCB.
002032 ;      So we have to keep a copy around
002033 ;      in case we need it.
002034      LD      HL,COPY
002035      LD      DE,FCB2
002036      LD      BC,32
002037      LDIR                      ;Get the copy
002038 ;      Put it here
002039 ;      Move up to 32 bytes
002040 ;      Copy the FCB so we can continue
002041
002042 ;      Now we know what Logical Record Length (LRL) to use in the
002043 ;      copy, so we can open the source file and create the target file
002044 ;      with the correct record lengths.
002045
002046 NOFILE: LD      HL,FCB1           ;Point at the filename in the FCB
002047      LD      A,@DSPLY
002048      RST     28H                ;Print that name
002049      RST     28H                ;Call the @DSPLY svc
002050      LD      HL,SPACES
002051      LD      A,@DSPLY
002052      RST     28H                ;Point at some spaces
002053      RST     28H                ;Space over a few places on the screen
002054      RST     28H                ;Call the @DSPLY svc
002055
002056      LD      DE,FCB1             ;Point at File Control Block for source file
002057      LD      HL,BUF1
002058      LD      A,(LRL)
002059      LD      B,A
002060      LD      A,@OPEN
002061      RST     28H                ;Put data in this
002062      JP      NZ,ERR              ;Read the Logical Record Length
002063 ;      Load the Logical Record Length
002064 ;      Open the source file
002065 ;      Call the @OPEN svc
002066 ;      Open failed
002067
002068      LD      HL,ARROW
002069      LD      A,@DSPLY
002070      RST     28H                ;Point at the arrow text
002071      RST     28H                ;Print that to show the direction of copy
002072      RST     28H                ;Call the @DSPLY svc
002073
002074      LD      DE,FCB2             ;Point at File Control Block for target file
002075      LD      A,(LRL)
002076      CP      0
002077      JR      Z,LRL256            ;Get the Logical Record Length
002078 ;      Is the LRL 256?
002079      JR      Z,LRL256            ;Then we do something special
002080      LD      HL,BUF2
002081      JR      LRLCOM              ;Use a different buffer for target file
002082 ;      Jump to common code
002083 LRL256: LD      HL,BUF1          ;We use the same buffer when the LRL is 256
002084 ;      since there is no need to block and de-block
002085 ;      the data.
002086 LRLCOM: LD      B,A
002087      LD      A,@INIT
002088 ;      Load the Logical Record Length
002089 ;      Open the target file

```


Sample Program C, continued

```

00271      RST      28H          ;Call the @INIT svc
00272      JR       NZ,ERR       ;Init failed
00273
00274      LD        DE,FILE2     ;We are going to get the filename for
00275                          ;the target file from the system
00276                          ;instead of using the one we have. The
00277                          ;reason for this is that the system will
00278                          ;append the drive number to the filename
00279                          ;if one was not specified.
00280      LD        A,(FCB2+7)    ;Get the Directory Entry Code for the file
00281      LD        B,A           ;Put the DEC here
00282      LD        A,(FCB2+6)    ;Get the Drive Number from the FCB
00283      AND       7            ;Lose all data except the drive number
00284      LD        C,A           ;Store drive number here
00285      LD        A,@FNAME      ;Have the system produce a filespec
00286      RST      28H          ;Call the @FNAME svc
00287      LD        HL,FILE2     ;Now point at the filespec produced
00288      LD        A,@DSPLY      ;and print it out
00289      RST      28H          ;Call the @DSPLY svc
00290
00291      LD        HL,SPACES     ;Space over a few more places
00292      LD        A,@DSPLY      ;so the display will look neat
00293      RST      28H          ;Call the @DSPLY svc
00294
00295      ;      At this point, both files are open and ready to be used.
00296      ;      The following code reads a record from the source file
00297      ;      and writes it to the target file. This is done until an
00298      ;      end of file is encountered.
00299
00300  LOOP:   LD        DE,FCB1     ;Point at file 1 (source file)
00301          LD        HL,BUFFER   ;Put data here
00302          LD        A,@READ     ;Read a record from the source file
00303          RST      28H          ;Call the @READ svc
00304          JR       NZ,EOF       ;Jump if the eof has been reached
00305          LD        DE,FCB2     ;Point at file 2 (target file)
00306
00307      ;      Before writing the record, display the record number, which
00308      ;      is obtained from the @LOC svc.
00309
00310          LD        A,@LOC      ;Get the current record number
00311          RST      28H          ;Call the @LOC svc
00312
00313          PUSH     BC           ;Get the current record number
00314          POP      HL           ;and put it in register HL
00315          LD        DE,LOCMSG+1 ;Store the result here.
00316          LD        A,@HEXDEC   ;Convert binary to ASCII in decimal format
00317          RST      28H          ;Call the @HEXDEC svc
00318
00319          LD        A,' '       ;Get a blank
00320          LD        HL,LOCMSG    ;Look at the front of the buffer
00321  EDIT:   CP        (HL)        ;Is the character a blank?
00322          JR       NZ,NUMBR     ;A number has been found
00323          INC      HL           ;Advance the pointer
00324          JR       EDIT         ;Loop until we find a number
00325
00326  NUMBR:  DEC      HL           ;Back up one position
00327          LD        A,'('       ;Get the character we want to insert
00328          LD        (HL),A      ;Store that character.
00329                          ;The buffer now contains
00330                          ;<none or more spaces>(record number)
00331                          ;<7 left-cursor characters><etx>
00332          LD        HL,LOCMSG    ;Point at this text
00333          LD        A,@DSPLY     ;and display it on the screen
00334          RST      28H          ;Call the @DSPLY svc
00335
00336      ;      Now write the record to the target file.
00337
00338          LD        DE,FCB2     ;Point at the FCB for the target file

```

Sample Program C, continued

```

00339      LD      HL,BUFFER      ;Point at the data read from file 1
00340      LD      A,@VER          ;Write a record to the target file
00341                                ;The @VER does the same thing as the
00342                                ;@WRITE svc, only it also checks the
00343                                ;data to make sure it is readable.
00344      RST      28H            ;Call the @VER svc
00345      JR      NZ,ERR          ;An error occurred on write; possibly
00346                                ;the disk is full.
00347      JR      LOOP            ;Loop until an error occurs.
00348
00349      ;      This code checks the error to make sure it was an end of file
00350      ;      condition and, if so, closes the source & target files.
00351
00352      EOF:    CP      28        ;Was it an end of file encountered?
00353      JR      Z,EOFYES         ;Yes, close the file
00354      CP      29              ;Was it "Record number out of range"?
00355      JR      NZ,ERR          ;No, must be some other error
00356
00357      ;      It is possible to get Error 29 if the file being copied has
00358      ;      an EOF that is not a multiple of the file's LRL
00359
00360      EOFYES: LD      DE,FCB1    ;Point at file 1 (source file)
00361      LD      A,@CLOSE         ;Close the file
00362      RST      28H            ;Call the @CLOSE svc
00363      JR      NZ,ERR          ;An error occurred, abort
00364
00365      LD      DE,FCB2          ;Point at file 2 (target file)
00366      LD      A,@CLOSE         ;Close it also
00367      RST      28H            ;Call the @CLOSE svc
00368      JR      NZ,ERR          ;An error occurred, abort
00369
00370      LD      HL,OK            ;Print a message saying the copy is done
00371      LD      A,@DSPLY         ;Call the @DSPLY svc
00372      RST      28H
00373
00374      QUIT:   LD      A,@EXIT    ;Exit to TRSDOS or the calling program
00375      RST      28H            ;Call the @EXIT svc
00376
00377      ;      The @EXIT svc does not return.
00378
00379      ERR:    OR      040H      ;Turn on bit 6, which
00380                                ;will cause the @ERROR svc to print
00381                                ;the short error message. Bit 7
00382                                ;is not set, which instructs the @ERROR
00383                                ;to abort this program and return to
00384                                ;TRSDOS Ready.
00385      LD      C,A              ;Put error code & flags in register C
00386      LD      A,@ERROR         ;Call the system error displayer
00387      RST      28H            ;Call the @ERROR svc
00388
00389      ;      Because bit 7 is not set, the @ERROR svc will not return.
00390
00391      ;      Storage Declaration
00392
00393      SPACES: DEFM      ' '      ;ASCII Space char.for display formatting
00394      DEFB      3
00395      ARROW:   DEFM      '>'      ;Arrow for display shows data direction
00396      DEFB      3
00397      OK:      DEFM      10%25    ;Advance cursor 10 spaces without erasing
00398      DEFM      '[Ok]'          ;Used to indicate the Copy is complete
00399      DEFB      0DH             ;Terminated with an <Enter>
00400      MSG1:    DEFM      'Copy Filespec >'
00401      DEFB      3
00402      MSG2:    DEFM      'To Filespec >'
00403      DEFB      3
00404      FEXST:   DEFM      'Destination File Already Exists - Ok to Delete it (Y/N) ?'
00405      DEFB      3

```

Sample Program C, continued

```

00406  BADFIL:  DEFM    'Invalid Filename - Try Again'
00407          DEFB    0DH
00408  LOCMSG:  DEFM    ' 12345)'          ;This will be used in building the LOC
00409          ;Display will appear as (d) to (dddddd).
00410          DEFB    7%24                ;Backspace without erasing
00411          DEFB    3                    ;Etx, used to get the @DSPLY svc to stop
00412
00413  FILE1:   DEFS    32                    ;User Text Originally placed here
00414  FILE2:   DEFS    32                    ;Target Filename goes here
00415  FCB1:    DEFS    32                    ;32 bytes for the File Control Block
00416  FCB2:    DEFS    32                    ;32 bytes for the File Control Block
00417  COPY:    DEFS    32                    ;An extra copy of the target FCB goes here
00418  LRL:     DEFB    0                    ;The Logical Record Length of the source
00419          ;file will be stored here
00420  BUF1:    DEFS    256                   ;System buffer for File 1
00421  BUF2:    DEFS    256                   ;System buffer for File 2
00422  BUFFER:   DEFS    256                   ;Data buffer for both files
00423
00424          END      BEGIN                ;"begin" is the starting address

```

Sample Program D

```

Ln #           Source Line

000001 ;           This program will read a sector from the disk in Drive 0
000002 ;           and will write it to a disk in Drive 1.  The disk in Drive 1
000003 ;           must be formatted, but should not have anything important on
000004 ;           it.  This program makes an assumption that the directory is
000005 ;           located on cylinder 20 (x'14').
000006
000007 PSECT 3000H           ;This program begins at x'3000'.
000009
000010 ;           Define the equates for the SVCs that will be used.
000011
000012 @ABORT: EQU 21           ;Abort and return to TRSDOS
000013 @CKDRV: EQU 33           ;Test to see if a drive is ready
000014 @DCSTAT: EQU 40          ;Verify that a drive is defined in the DCT
000015 @ERROR: EQU 26           ;Display an error message
000016 @EXIT: EQU 22           ;Return to TRSDOS or the calling program
000017 @RDSEC: EQU 49           ;Read a sector
000018 @RDSSC: EQU 85           ;Read a system sector
000019 @WRSEC: EQU 53           ;Write a sector
000020 @WRSSC: EQU 54           ;Write a system sector
000021
000022 ;           Other Equates
000023
000024 SYSSEC: EQU 1400H        ;The system sector is Cylinder 20, Sector 0
000025 USRSEC: EQU 0000H        ;The regular sector is Cylinder 0, Sector 0
000026
000027 ;           First, test the target drive and make sure it is defined.
000028
000029 START: LD C,1           ;Select Drive 1
000030 LD A,@DCSTAT           ;Ask if the drive is listed in the DCT
000031 RST 28H               ;Call the @DCSTAT svc
000032 JR NZ,ERROR           ;If NZ, then the drive is not defined
000033                       ;and we will abort execution.
000034
000035 ;           Now, test and make sure the target drive contains a formatted
000036 ;           disk and is write-enabled.
000037
000038 LD C,1                 ;Select Drive 1
000039 LD A,@CKDRV            ;Test to see if the disk is formatted
000040                       ;and is write-enabled.  Note that the
000041                       ;disk must be formatted by TRSDOS 6.x
000042                       ;or by LDOS 5.1.x to be considered
000043                       ;"formatted" by this svc.
000044 RST 28H               ;Call the @CKDRV svc
000045 LD A,8                 ;This will become the error number if the
000046                       ;drive was not ready.  This is done
000047                       ;because the @CKDRV svc does not return error
000048                       ;codes.
000049 JR NZ,ERROR           ;The drive is not ready
000050 LD A,15                ;This will become the error number if the
000051                       ;drive is ready and is write-protected.
000052                       ;As above, this is done because @CKDRV does
000053                       ;not return error messages.
000054 JR C,ERROR            ;The disk is formatted, but it is
000055                       ;write-protected.  In either case, abort.
000056
000057 ;           Now that we know the target drive is ready, read a sector
000058 ;           from the source drive and write it to the target drive (Drive 1).
000059
000060 LD C,0                 ;Select Drive 0
000061 LD DE,USRSEC           ;Read the first sector on the disk,
000062                       ;Cylinder 0, Sector 0.
000063 LD HL,BUFF            ;Point to a buffer which will hold the sector
000064 LD A,@RDSEC           ;Read a non-system sector
000065 RST 28H               ;Call the @RDSEC svc
000066 JR NZ,ERROR           ;If NZ, an error occurred, so abort
000067

```

Sample Program D, continued

```

00068 ;      Now, write the sector to the target drive.
00069
00070 LD      C,1          ;Select Drive 1
00071 LD      DE,USRSEC    ;Write the sector to Cylinder 0, Sector 0
00072                      ;on Drive 1
00073 LD      HL,BUFF      ;Point to the buffer containing the sector
00074 LD      A,@WRSEC     ;Write the sector to disk
00075 RST     28H          ;Call the @WRSEC svc
00076 JR      NZ,ERROR    ;If NZ, an error occurred, so abort
00077
00078 ;      Now we will read a system sector from Drive 0 and write it on
00079 ;      drive 1.  The difference between a system sector and a non-system
00080 ;      sector is that the Data Address Marks (DAM) are different.  These
00081 ;      were written to the disk when it was formatted.  TRSDOS 6.x uses
00082 ;      these as an extra check to make sure that a write of user data
00083 ;      does not accidentally get placed over a sector containing system
00084 ;      data.  All of the sectors in the directory cylinder are marked
00085 ;      as system sectors.
00086
00087 LD      C,0          ;Select Drive 0
00088 LD      DE,SYSSEC    ;Read Cylinder 20, Sector 0
00089 LD      HL,BUFF      ;Store the sector at this address
00090 LD      A,@RDSSC     ;Read a system sector
00091 RST     28H          ;Call the @RDSSC svc
00092 JR      NZ,ERROR    ;An error occurred, so abort
00093
00094 ;      Now write the sector to the target drive as a system sector.
00095 ;      There is no requirement that a sector must be placed at the
00096 ;      same cylinder and sector location as it was read from, but
00097 ;      for simplicity, we are doing that.
00098
00099 LD      C,1          ;Select Drive 1
00100 LD      DE,SYSSEC    ;Write Cylinder 20, Sector 0
00101 LD      HL,BUFF      ;Point to the data to be written
00102 LD      A,@WRSSC     ;Write a system sector
00103 RST     28H          ;Call the @WRSSC svc
00104 JR      NZ,ERROR    ;An error occurred, so abort
00105
00106 LD      A,@EXIT      ;Return to TRSDOS or the calling program
00107 RST     28H          ;Call the @EXIT svc
00108
00109 ;      This routine displays an error message if anything goes wrong.
00110 ;      Note that @CKDRV does not return an error message, so @ERROR
00111 ;      cannot be used for it without some manipulation.
00112
00113 ERROR:  OR      0C0H   ;Set bit 7
00114 LD      C,A         ;Load error number into register C
00115 LD      A,@ERROR    ;This will display the error message
00116                      ;and return to the calling program
00117 RST     28H          ;Call the @ERROR svc
00118
00119 LD      A,@ABORT     ;Now, force an abort.  This will return
00120 ;to TRSDOS Ready and will abort any
00121 ;JCL file that is currently executing
00122 RST     28H          ;Call the @ABORT svc
00123
00124 BUFF:  DEFS     256   ;256-byte buffer to store the sector that
00125                      ;is read and then written
00126
00127 END      START

```

Sample Program E

```

Ln #           Source Line

000001 ;           This program displays the filenames of the disk in
000002 ;           Drive 0 three different ways.
000003
000004 PSECT    3000H           ;Program begins at x'3000'
000005
000006
000007 ;           First, declare the equates for the SVCs we intend to use.
000008 ;           This is not mandatory, but it makes the program easier to follow.
000009
000010 @CMNDI: EQU    24           ;Execute a TRSDOS command and return
000011 ;to TRSDOS Ready
000012 @CMNDR: EQU    25           ;Execute a TRSDOS command and return
000013 ;to the calling program
000014 @DODIR: EQU    34           ;Display visible filenames on the
000015 ;specified disk drive
000016
000017
000018 ;           First, pass a "DIR :0" command to the system. TRSDOS will
000019 ;           execute this command and then return to this program.
000020
000021 START: LD      HL,DIR0      ;Point at command we want to execute
000022 LD      A,@CMNDR           ;Execute the specified command and return
000023 RST     28H                ;Call the @CMNDR svc
000024
000025 ;           You may have noticed that the DIR displayed the files, but that
000026 ;           they were not sorted alphabetically. This is because the DIR
000027 ;           command will not use memory above x'3000' when it is invoked with
000028 ;           a @CMNDR svc. This prevents the DIR command from performing a
000029 ;           sort of the filenames.
000030
000031
000032 ;           Now do a directory command using the @DODIR svc.
000033
000034 LD      B,0                ;Use Function 0 which displays all
000035 ;visible files in the directory.
000036 LD      C,0                ;Put source drive number in register C
000037 LD      A,@DODIR           ;The filenames will be read from the
000038 ;directory and displayed in the
000039 ;order they appear in the directory.
000040 RST     28H                ;Call the @DODIR svc
000041
000042
000043 ;           Now pass a "DIR :0" command to the system. This time
000044 ;           the command will be executed and then TRSDOS will not return
000045 ;           to this program, but will return to TRSDOS Ready.
000046
000047 LD      HL,DIR0            ;Point at the command we want performed
000048 LD      A,@CMNDI           ;and execute it, but don't return to
000049 ;this program.
000050 RST     28H                ;Call the @CMNDI svc
000051 ;This svc returns to TRSDOS Ready.
000052
000053 ;           Note that when the library command DIR is performed this time,
000054 ;           the display of files is sorted. This is because DIR determines
000055 ;           that it was invoked with a @CMNDI svc, and it will not return
000056 ;           to the calling program. Therefore, DIR is free to use the
000057 ;           memory above x'3000' to perform the sort of the filenames in
000058 ;           the directory.
000059
000060
000061 ;           Constants
000062
000063 DIR0:  DEFM    'DIR :0'     ;This command is passed to TRSDOS
000064 ;via the @CMNDR and @CMNDI SVCs.
000065 DEFB    0DH                ;It must be terminated with an <ENTER>.
000066
000067 END      START

```

Sample Program F

Ln #	Source Line
00001	;
00002	This program adds to the system task scheduler a task
00003	which displays the date and a running count of the number
00004	of times the task has been executed.
00005	For simplicity, the program tries to use task slot 0.
00006	If it is already in use, it assumes that the task using that
00007	slot is this program, and it kills the task. It then tries to
00008	recover the memory used by the task in high memory.
00009	If the task slot is not in use, the task is placed in high memory,
00010	and the address of the task is passed to the task scheduler.
00011	The first time you run this program it adds the task, and the
00012	next time you run this program, it removes the task.
00013	PSECT 3000H ;This program starts at x'3000'
00015	;
00016	First, declare the equates for the SVCs we intend to use.
00017	This is not mandatory, but it makes the program easier to follow.
00018	;
00019	@ADTSK: EQU 29 ;Add a task entry to the scheduler
00020	@CKTSK: EQU 28 ;Check to see if a task slot is in use
00021	@DATE: EQU 18 ;Return the date in ASCII format
00022	@DSPLY: EQU 10 ;Display a message
00023	@EXIT: EQU 22 ;Return to TRSDOS Ready or the caller
00024	@GTMOD: EQU 83 ;Locate a memory module
00025	@HEXDEC: EQU 97 ;Convert a binary value to decimal ASCII
00026	@HIGH\$: EQU 100 ;Read or modify HIGH\$ or LOW\$
00027	@RMTSK: EQU 30 ;Remove a task entry from the scheduler
00028	@VDCTL: EQU 15 ;Perform video operations
00029	@WHERE: EQU 7 ;Find out where the program counter is
00030	;
00031	when this SVC is executed. This is
00032	useful in relocatable code that must
00033	make absolute address references to
00034	call subroutines or modify data.
00035	;
00036	Below we will define a macro to simulate a call relative
00037	instruction. Since the task must be able to run no matter
00038	where it is placed, it must use relative jumps and calls.
00039	The Z80 instruction set has a jump relative (JR), but does
00040	not have a call relative instruction. This can be simulated
00041	using the @WHERE SVC, which returns the address of the caller
00042	in a register. This address can be adjusted and placed on
00043	the stack as a return address. Then a jump relative can be used
00044	to reach the subroutine.
00045	;
00046	CALLR: MACRO #1 ;#1 will be the address you want to call
00047	PUSH HL ;Save the registers we damage
00048	PUSH BC ;Save it
00049	PUSH AF ;Save it
00050	LD A,@WHERE ;Get our current address
00051	RST 28H ;Call the @WHERE svc
00052	LD BC,3+1+1+1+1+2 ;Get the lengths of the instructions after
00053	the SVC. This will allow the subroutine
00054	to return to the correct address.
00055	ADD HL,BC ;Add that offset to where we are
00056	POP AF ;Put stack back
00057	POP BC ;Restore registers
00058	EX (SP),HL ;Put return address on stack and restore HL
00059	JR #1 ;Jump to the subroutine
00060	ENDM ;End of the macro
00061	;
00062	;
00063	This is the main program. It loads at x'3000'. It decides
00064	if it needs to add or remove the task in the scheduler tables.
00065	If it adds the task, it moves a copy to the top of memory and
00066	protects it, and adds a task entry to the scheduler.
00067	If it is removing a task, it kills the entry in the scheduler

Sample Program F, continued

```

00068 ;      tables, and then attempts to recover the memory used by the task.
00069
00070 BEGIN: LD      C,0           ;First, we will test slot 0
00071        LD      A,@CKTSK      ;to see if anyone is using it
00072        RST     28H           ;Call the @CKTSK svc
00073        JR      NZ,KILLIT     ;There is a task using slot 0, kill it
00074
00075 ;      At this point, we want to add a task to high memory.
00076 ;      First we find the value for HIGH$ and put a copy of the
00077 ;      task there. Then we protect the task by moving HIGH$ below
00078 ;      the new task.
00079
00080        LD      HL,0           ;First, get the value of HIGH$
00081        LD      B,H           ;Read HIGH$
00082        LD      A,@HIGH$
00083        RST     28H           ;Call the @HIGH$ svc
00084        LD      (ENDADD),HL    ;Save this value as the last address
00085                                ;that the task will be stored in once it
00086                                ;is moved to high memory
00087
00088        LD      DE,HL          ;Put that value here
00089        LD      HL,MODEND-1    ;Point at the end of the module
00090        LD      BC,MODEND-MODULE ;Move the module from where it is
00091                                ;right now to a position below HIGH$
00092        LDDR                                ;Do the copy
00093
00094        LD      HL,DE          ;Now protect the module using HIGH$
00095        LD      B,0           ;Update HIGH$
00096        LD      A,@HIGH$
00097        RST     28H           ;Call the @HIGH$ svc
00098
00099 ;      Now we need to load the TCB entry in the module with the address
00100 ;      of the first instruction to be executed.
00101
00102        LD      IX,HL          ;IX now points at memory header
00103        LD      BC,ENTRY-MODULE+1 ;Get the offset into the module
00104                                ;of the first instruction
00105        ADD     HL,BC          ;HL now contains the actual starting address
00106        LD      (IX+(1+MODTCB-MODULE)),L ;Store LSB of the address
00107        LD      (IX+1+(1+MODTCB-MODULE)),H ;Store MSB of the address
00108
00109 ;      Now the task is ready to run. We now add the entry to the task
00110 ;      scheduler table.
00111
00112        LD      BC,MODTCB-MODULE+1 ;Get offset into the
00113                                ;module of the TCB word
00114        PUSH    IX            ;Get a copy of the base address
00115        POP     HL            ;Put base address here
00116        ADD     HL,BC         ;Now HL points at TCB address
00117        LD      DE,HL         ;Put that value in DE
00118        LD      C,0           ;Add this entry to task slot 0
00119        LD      A,@ADTSK      ;Add this task, to be run every 266.67 msec
00120        RST     28H           ;Call the @ADTSK svc
00121
00122 ;      The main program has now done its work and can exit.
00123
00124        LD      HL,ADDED       ;Point at a message saying what was done
00125        LD      A,@DSPLY       ;and print it
00126        RST     28H           ;Call the @DSPLY svc
00127
00128        LD      A,@EXIT        ;Now exit
00129        RST     28H           ;Call the @EXIT svc
00130
00131 ;      This SVC does not return.
00132
00133
00134 ;      This part of the code removes the task from the scheduler
00135 ;      tables and then attempts to recover the memory that was used

```


Sample Program F, continued

```

00136 ;      by the task in high memory.  If another high memory module
00137 ;      was added AFTER this task was added, then the memory that
00138 ;      was used by this task cannot be recovered.
00139
00140 KILLIT: LD      C,0           ;We want to remove the task in slot 0
00141         LD      A,@RMTSK
00142         RST     28H          ;Call the @RMTSK svc
00143
00144 ;      At this point, the task is no longer called by the operating
00145 ;      system.  Now we want to determine if we can
00146 ;      reclaim the memory it was using.
00147
00148         LD      DE,MODNAM     ;Point at the name of the module
00149         LD      A,@GTMOD      ;Look for a module with that name
00150         RST     28H          ;Call the @GTMOD svc
00151         JR      NZ,CANT       ;If NZ is set, then we killed some other
00152                               ;task that was using slot 0.  Oops.
00153                               ;In that case, just stop and don't do any
00154                               ;more damage.
00155         LD      IX,HL         ;Set IX to point to the module.
00156         LD      B,0          ;Read the current value of HIGH$
00157         LD      HL,0          ;to see if this is the first program in
00158                               ;high memory
00159         LD      A,@HIGH$      ;If it is, then we can recover the space
00160         RST     28H          ;Call the @HIGH$ svc
00161         INC     HL            ;Move HIGH$ up by one byte
00162         PUSH    IX            ;Take the address of our module
00163         POP     DE            ;and store it here
00164         XOR     A             ;Compare these
00165         SBC     HL,DE         ;Are they the same?
00166         JR      NZ,CANT       ;No, the high memory module can't be removed
00167
00168 ;      At this point, we know it is ok to reclaim the memory used by the
00169 ;      high memory task.
00170
00171         LD      HL,(IX+2)     ;Read the end of module value out of the
00172                               ;header information
00173         LD      B,0           ;Update the HIGH$ value
00174         LD      A,@HIGH$
00175         RST     28H          ;Call the @HIGH$ svc
00176
00177         LD      HL,OK         ;Point to a message saying all is well
00178         LD      A,@DSPLY      ;and print it
00179         RST     28H          ;Call the @DSPLY svc
00180
00181         LD      A,@EXIT       ;Exit the main program
00182         RST     28H          ;Call the @EXIT svc
00183
00184
00185 ;      Here we will display a message saying we removed the task from
00186 ;      the scheduler table, but we cannot reclaim the memory that was
00187 ;      used.
00188
00189 CANT:   LD      HL,RECLM      ;Point to the message
00190         LD      A,@DSPLY      ;and display it
00191         RST     28H          ;Call the @DSPLY svc
00192
00193         LD      A,@EXIT       ;Now exit
00194         RST     28H          ;Call the @EXIT svc
00195
00196
00197 ;      Messages
00198
00199 ADDED:  DEFM    'Task placed in high memory and scheduled.'
00200         DEFB    0DH
00201 OK:     DEFM    'Task removed from scheduler table and memory reclaimed.'
00202         DEFB    0DH
00203 RECLM:  DEFM    'Task removed from scheduler table, but memory could not '

```

Sample Program F, continued

```

00204      DEFM      'be recovered.
00205      DEFB      0DH
00206
00207      ;          The Task begins at this point.  This part of the program loads
00208      ;          in low memory but is relocated to a point just below HIGH$.
00209
00210      ;          This is the Memory Header Block.  This block of data allows
00211      ;          the system to locate this module in memory by name,
00212      ;          using the @GTMOD svc.
00213
00214      MODULE: JR      ENTRY          ;Jump (relative) to the starting address
00215      ENDADD: DEFW    0              ;The highest address in the program.
00216      ;          ;This value is patched in before the program
00217      ;          ;is relocated.  This will be used
00218      ;          ;later in recovering the memory used by
00219      ;          ;this task.
00220      DEFB      MODTCB-MODNAM        ;Number of bytes in the name field below.
00221      MODNAM: DEFM    'UPTIME'       ;This is the name of the module and is
00222      ;          ;used to identify the module.
00223      MODTCB: DEFW    0              ;Actual address to start execution.  This
00224      ;          ;value is patched in after the program is
00225      ;          ;relocated.
00226      DEFW      0                  ;Spare system pointer - RESERVED
00227
00228      ;          This area contains data used by the task.  It is addressed using
00229      ;          the IX register which points to the task when it is executed.
00230
00231      COUNTER: DEFW    0              ;Count of how many times we have run
00232      DATBUF: DEFS     9              ;The date is stored here
00233
00234      ;          This is the actual task.
00235      ;          On entry to the task, IX points at the Task Control Block (TCB),
00236      ;          which in this program is the label 'MODTCB'.  All data is
00237      ;          referenced by indexing from that address.
00238
00239
00240      ENTRY:  PUSH     IY              ;Save this register.  It is not saved by
00241      ;          ;the Task Scheduler, and we use it.
00242      ;          ;Registers AF, BC, DE, and HL are saved
00243
00244      ;          Now we will read the current date.
00245
00246      LD        HL,IX                ;Get a copy of the index pointer
00247      LD        BC,DATBUF-MODTCB    ;Get the offset needed to access the date
00248      ADD       HL,BC                ;Now we have a pointer to the date
00249
00250      PUSH      IX                    ;Save the pointer to the start of the task
00251      PUSH      HL                    ;Save a copy of that pointer
00252      LD        A,@DATE              ;Ask the system what the date is
00253      RST       28H                  ;Call the @DATE svc
00254
00255      LD        (HL),0               ;Terminate the date string
00256
00257      POP       DE                    ;Put pointer to the date here
00258      PUSH      DE                    ;We will use this pointer later on
00259      LD        HL,0028H              ;Put the cursor on the top line,
00260      ;          ;specified in register HL
00261      ;          ;at the 41st position on the screen
00262      CALLR     WRITE                ;Write the message at the position
+      PUSH      HL                    ;Save the registers we damage
+      PUSH      BC                    ;Save it
+      PUSH      AF                    ;Save it
+      LD        A,@WHERE              ;Get our current address
+      RST       28H                  ;Call the @WHERE svc
+      LD        BC,3+1+1+1+1+2      ;Get the lengths of the instructions after
+      ;          ;the SVC.  This will allow the subroutine
+      ;          ;to return to the correct address.

```

Sample Program F, continued

```

+      ADD      HL,BC      ;Add that offset to where we are
+      POP      AF        ;Put stack back
+      POP      BC        ;Restore registers
+      EX       (SP),HL    ;Put return address on stack and restore HL
+      JR       WRITE     ;Jump to the subroutine
;      ;Note that the above was actually a macro
;      ;which performs a relative call.
00263
00264
00265
00266 ;      This part of the task displays a count of the number of times
00267 ;      the task has been executed.
00268
00269      POP      DE        ;Get the pointer to DATBUF back
00270      POP      IX        ;Get the pointer to the beginning of
00271                        ;this task
00272      PUSH     DE        ;Save the pointer to DATBUF again
00273      LD       BC,COUNTER-MODTCB ;Get the offset to our data
00274                        ;area
00275      LD       HL,IX      ;Put a copy of the base address in HL
00276      ADD      HL,BC      ;Add offset. Now HL points to COUNTER:
00277      LD       IY,HL      ;Put the pointer to COUNTER in IY
00278      LD       L,(IY)     ;Get LSB of the counter
00279      LD       H,(IY+1)   ;Get MSB of the counter
00280      INC      HL         ;Increment the number of times we have run
00281      LD       (IY),L     ;Store the LSB of the counter
00282      LD       (IY+1),H   ;Store the MSB of the counter
00283
00284      LD       A,@HEXDEC  ;Convert the count to decimal
00285      RST      28H        ;Call the @HEXDEC svc
00286
00287      XOR      A          ;Get a zero
00288      LD       (DE),A     ;Terminate the count string
00289
00290      POP      DE        ;Put pointer to date here
00291      LD       HL,0036H   ;Put the cursor on the top line,
00292                        ;specified in register HL
00293                        ;at the 55th position on the screen
00294      CALLR    WRITE     ;Write the message at the position
+      PUSH     HL        ;Save the registers we damage
+      PUSH     BC        ;Save it
+      PUSH     AF        ;Save it
+      LD       A,@WHERE   ;Get our current address
+      RST      28H        ;Call the @WHERE svc
+      LD       BC,3+1+1+1+1+2 ;Get the lengths of the instructions after
+                        ;the SVC. This will allow the subroutine
+                        ;to return to the correct address.
+
+      ADD      HL,BC      ;Add that offset to where we are
+      POP      AF        ;Put stack back
+      POP      BC        ;Restore registers
+      EX       (SP),HL    ;Put return address on stack and restore HL
+      JR       WRITE     ;Jump to the subroutine
;      ;Note that the above was actually a macro
;      ;which performs a relative call.
00295
00296
00297
00298 ;      Now we restore the IY register and return to the task scheduler.
00299
00300      POP      IY        ;Restore IY value
00301      RET          ;Return to the task scheduler
00302
00303
00304 ;      This routine places characters on the display using the @VDCTL
00305 ;      svc instead of @DSP or @DSPLY. This allows the cursor to
00306 ;      remain at its current position when we write to the screen.
00307 ;      This routine must be called using the relocatable call macro
00308 ;      CALLR.
00309
00310 WRITE: LD       B,2      ;Put character on the display
00311
00312 TSKLP: LD       A,(DE)   ;Get a character to display

```

Sample Program F, continued

00313	OR	A	;Is it time to stop putting this on
00314			;the display?
00315	RET	Z	;Yes, return to the caller
00316	PUSH	HL	;Save the registers, as the SVC will
00317	PUSH	DE	;alter the contents
00318	PUSH	BC	
00319	LD	C,A	;Put the character here
00320	LD	A,@VDCTL	;Put character on screen at specified position
00321	RST	28H	;Call the @VDCTL svc
00322	POP	BC	;Restore registers
00323	POP	DE	
00324	POP	HL	
00325	INC	L	;Advance display position
00326	INC	DE	;Point to next character to display
00327	JR	TSKLP	;Loop till date is completely displayed
00328			
00329	MODEND: END	BEGIN	;End of task and main program

Sample Program G

```

000001 ;      This program is a sample Extended Command Interpreter.  You
000002 ;      may make the ECI as large or small as you require.  You may
000003 ;      use allof main memory, or you can restrict yourself to the
000004 ;      system overlay area (x'2600' to x'2FFF').
000005 ;      To pass a command to the normal system interpreter for
000006 ;      processing, use the @CMNDI svc.  TRSDOS executes the command
000007 ;      and reloads the ECI.  If you want to have multiple entry
000008 ;      points, Bits 2 - 0 in EFLAG$ are in Register A on entry
000009 ;      (in Bits 6 - 4), or you may read EFLAG$ yourself.
000010 ;      EFLAG$ is totally dedicated to the ECI, and may contain any
000011 ;      non-zero value.  If EFLAG$ contains a zero, TRSDOS uses its
000012 ;      own interpreter.  Other programs that want to activate an ECI,
000013 ;      should set the EFLAG$ to a non-zero value and execute a @EXIT
000014 ;      svc.
000015
000016 ;      To install an ECI, use the command:
000017 ;          COPY filename SYS13/SYS.LSIDOS:d (C=N)
000018 ;      If you omit the C=N option, the SYS13 file loses it's "SYS"
000019 ;      status and you will receive 'Error 07' messages when you try
000020 ;      to use it as a ECI.
000021
000022 ;      When SYS1 (the normal command interpreter) has completed it's
000023 ;      normal housekeeping and is about to display the "TRSDOS Ready"
000024 ;      prompt, it checks EFLAG$.  If EFLAG$ contains a non-zero
000025 ;      value, TRSDOS loads and executes the Extended Command
000026 ;      Interpreter.
000027 ;      To execute this program, type <*><Enter>.
000028
000029 ;      This program checks EFLAG$ to see if it is zero.  If so, it
000030 ;      sets it to a non-zero value.  This causes this program to be
000031 ;      used instead of the normal interpreter when you execute an
000032 ;      @EXIT or @ABORT SVC.  (@CMNDI and @CMNDR invoke the TRSDOS
000033 ;      interpreter.)  If EFLAG$ is non-zero, the ECI displays a few
000034 ;      prompts and the names of all visible /CMD files on logical
000035 ;      Drive 0.
000036 ;      The operator may then type the name of a program to execute.
000037
000038 ;      If you press <Break>, this program sets EFLAG$ to 0, executes
000039 ;      an @EXIT SVC and returns to TRSDOS Ready.
000040
000041 ;      By pressing a number, 0 through 7, you can specify the drive
000042 ;      that TRSDOS searches.  This program stores this value in
000043 ;      EFLAG$.  Each time this program is invoked, it reads the value
000044 ;      from EFLAG$ and uses that drive.
000045
000046 ;      Note that if a drive is not enabled, not formatted, doesn't
000047 ;      exist, or contains no visible /CMD files, this program
000048 ;      redisplayes the prompt.
000049
000050      PRINT  SHORT,NOMAC
000051
000052      PSECT  3000H          ;This program starts at x'3000'
000053
000054 ;      Declare the equates for the SVCs used.
000055 ;      This is not mandatory, but it makes the program easier to
000056 ;      follow.
000057 @EXIT: EQU    22          ;Exit and return to TRSDOS
000058 @DSPLY: EQU    10         ;Display a string
000059 @FLAG$: EQU    101        ;Locate the system flag area
000060 @DODIR: EQU    34         ;Get the names of filenames
000061 @KEYIN: EQU    9          ;Accept a command and allow editing
000062 @CMNDI: EQU    24         ;Execute a command (using SYS1)
000063
000064 ;      On entry, determine if EFLAG$ is set to zero or not.  If it
000065 ;      is set to zero, this program is being started by typing
000066 ;      PROGRAM<Enter> or <*><Enter>.  In that case, set EFLAG$ to a
000067 ;      non-zero value so that in future, TRSDOS uses this interpreter
000068 ;      instead of it's own.

```

Sample Program G, continued

```

00069 ;      If EFLAG$ is non-zero, this initialization has already been
00070 ;      done and can be skipped.
00071
00072 BEGIN: LD      A,@FLAGS      ;Get the starting address of the flag
area
00073      RST      28H            ;Call the @FLAGS svc
00074
00075      LD      A,(IY+4)        ;Read the EFLAG$ (ECI flag)
00076      OR      A              ;Is it set to zero?
00077      JR      NZ,ECIRUN      ;Run the ECI
00078
00079      LD      A,8             ;Get a non-zero value. The value
00080      ;needs to be a non-zero value that
00081      ;does not set Bits 0, 1 or 2. The
00082      ;default drive # is kept in these bits.
00083      LD      (IY+4),A        ;Set the EFLAG$ to a non-zero value
00084      LD      HL,PROMPT      ;Explain how this works
00085      JR      ECIGO          ;Display message
00086
00087 ;      When the system is about to display
00088 ;      TRSDOS Ready, it executes this code instead.
00089
00090 ECIRUN: LD      HL,SPROMPT    ;Point at the prompt to use
00091 ECIGO:  LD      A,@DSPLY      ;Display the prompt
00092      RST      28H            ;Call the @DSPLY svc
00093
00094 ;      Display the names of all /CMD files
00095
00096      LD      A,(IY+4)        ;Get the EFLAG$
00097      AND      7              ;Delete all but the drive number field
00098      LD      C,A            ;Store the drive number for the svc
00099      LD      A,@DODIR        ;Do a directory display
00100      LD      B,2            ;Display visible, non-system files
00101      LD      HL,CMDTXT       ;that match "CMD" (stored at CMDTXT)
00102      RST      28H            ;Call the @DODIR svc
00103
00104 ;      Prompt for a filename or a function key.
00105
00106 ASK:   LD      HL,BUFFER      ;Point at text buffer
00107      LD      B,9             ;Allow up to 8 characters and <Enter>
00108      LD      C,0            ;Required by the svc
00109      LD      A,@KEYIN        ;Input text with edit capability
00110      RST      28H            ;Call the @KEYIN svc
00111
00112      JR      C,QUIT          ;The carry flag is set when the
00113      ;operator presses <BREAK>. Zero the
00114      ;EFLAG$ and exit to TRSDOS
00115
00116      LD      HL,BUFFER      ;Point at the start of the buffer
00117      LD      A,(HL)         ;Get the character
00118
00119      CP      0DH            ;Did they type anything?
00120      JR      Z,ASK          ;No, just repeat the prompt.
00121      ;If you want to redisplay the
00122      ;directory, change "ASK" to "ECIRUN".
00123
00124      SUB     '0'            ;Convert value to binary
00125      CP      7+1            ;Is the character a 0 - 7?
00126      JR      NC,NAME        ;Must be a filename
00127
00128 ;      The operator has typed 1 or more characters that start with
00129 ;      a number. This program assumes that the operator is defining
00130 ;      a new drive number and stores this value in EFLAG$ for
00131 ;      future use. TRSDOS does not alter this value.
00132 ;      The next time this program is run, EFLAG$ contains the
00133 ;      same value and this program knows what drive to scan.
00134
00135      LD      B,A            ;Save the drive number
00136      LD      A,(IY+4)        ;Get the EFLAG$

```

Sample Program G, continued

```

00137      AND      8          ;Delete the old drive number
00138      OR       B          ;Insert the new drive number
00139      LD       (IY+4),A    ;Save that value for future use
00140      JR       ECIRUN      ;Scan the new drive
00141
00142      ;      The operator pressed <Break>. Turn off the ECI and return to
00143      ;      TRSDOS.
00144      QUIT:     XOR      A          ;Get a zero
00145      LD       (IY+4),A    ;Set EFLAG$ to zero
00146      LD       HL,EPROPT   ;Point at the shutdown message
00147      LD       A,@DSPLY    ;And acknowledge the <Break>
00148      RST      28H        ;Call the @DSPLY svc
00149      LD       A,@EXIT     ;Return to TRSDOS Ready
00150      RST      28H        ;Call the @EXIT svc
00151
00152      ;      The operator entered what might be a filename or a library
00153      ;      command. Pass it to TRSDOS for processing. If there is an
00154      ;      error, TRSDOS is responsible for determining what the error is
00155      ;      and printing a message.
00156      ;      (HL already points at the start of the buffer.)
00157
00158      NAME:     LD       A,0DH    ;Look for this character
00159      FDIV:     CP       (HL)     ;In the command
00160      JR       Z,FOUND        ;Found the end of the filename
00161      INC      HL              ;Move character to next byte
00162      JR       FDIV          ;Find the divider (in this case, a 0DH)
00163
00164      ;      Found the end of a filename, and add the drive number from
EFLAG$.
00165      ;      Note that this program may not work properly if the operator
00166      ;      supplies a drive number as part of the filename.
00167
00168      FOUND:    LD       (HL),':'  ;Add a drive number to the filename
00169      INC      HL              ;Advance the pointer to the next byte
00170      LD       A,(IY+4)        ;Get the EFLAG$ value
00171      AND      7              ;Delete all but the drive number
00172      ADD      A,'0'          ;Convert the binary value to ASCII
00173      LD       (HL),A          ;Add that to the filename
00174      INC      HL              ;Advance the pointer to the next byte
00175      LD       (HL),0DH        ;Write a terminator on the end
00176      LD       HL,BUFFER      ;Point at the text entered
00177      LD       A,@CMNDI       ;Execute the command, but do not
00178      ;      return. Since this program is the
00179      ;      command processor at this time,TRSDOS
00180      ;      returns control to the beginning of
00181      ;      this module after executing the
00182      ;      command.
00183      RST      28H          ;Call the @CMNDI svc
00184
00185      ;      Messages and text storage
00186      PROMPT:   DEFM      '[Extended Command Interpreter Is Now Operational]'
00187      DEFB      0AH
00188      DEFB      0AH
00189      DEFM      'Press <BREAK> to use the normal interpreter,
00190      DEFB      0AH
00191      DEFM      'type <Number><ENTER> to change the default drive
00192      DEFB      0AH
00193      DEFM      'or type the name of the program to run and press
00194      DEFB      0DH          ;Terminate the display
00195
00196      SPROMPT:  DEFB      0AH
00197      DEFM      '[ECI On] <BREAK> to abort, n<ENTER> for new drive or
00198      DEFM      ' program<ENTER>'
00199      DEFB      0DH          ;Terminate the message
00200

```

Sample Program G, continued

```
00201 EPROMPT:DEFM '[Extended Command Interpreter Is Now Disabled]'  
00202         DEFB 0DH  
00203  
00204 CMDTXT: DEFM 'CMD'  
00205 BUFFER: DEFS 11           ;Allow for filename, drivespec and 0DH  
00206  
00207         END BEGIN           ;"BEGIN" is the starting address
```




9/ Technical Information on TRSDOS

Commands and Utilities

TRSDOS commands and utilities are covered extensively in the *Disk System Owner's Manual*. This section presents additional information of a technical nature on several of the commands and utilities.

Changing the Step Rate

The step rate is the rate at which the drive head moves from cylinder to cylinder. You can change the step rate for any drive by using one of the commands described below.

To set the step rate for a particular drive, use the following command:

SYSTEM (DRIVE = *drive*, STEP = *number*)

drive is any drive enabled in the system. *number* can be 0, 1, 2, or 3 and represents one of the following step rates in milliseconds:

0 = 6 milliseconds
1 = 12 milliseconds
2 = 20 milliseconds
3 = 30 milliseconds

Unless it is SYSGENed, the step value you select remains in effect for the specified drive only until the system is re-booted or turned off. If you use the SYSGEN command while the step value is in effect, then this step rate is written to the configuration file (CONFIG/SYS) on the disk in the drive specified by the SYSGEN command.

On a new TRSDOS disk, the step rate is set to 12 milliseconds.

To set the default bootstrap step rate used with the FORMAT utility, use the following command:

SYSTEM (BSTEP = *number*)

number is 0, 1, 2, or 3, which correspond to 6, 12, 20, and 30 milliseconds, respectively.

The value you select for *number* is stored in the system information sector on the disk in Drive 0. (On a new TRSDOS disk, the bootstrap step rate is set to 12 milliseconds.)

If you switch Drive 0 disks or change the logical Drive 0 with the SYSTEM (SYSTEM) command, the default value is taken off the new Drive 0 disk if you format a disk.

You can change the bootstrap step rate for a particular FORMAT operation if you do not want to use the default. Specify the new value for STEP on the FORMAT command line as follows:

FORMAT :*drive* (STEP = *number*)

drive is the drive to be used for the FORMAT. *number* is 0, 1, 2, or 3, which correspond to 6, 12, 20, and 30 milliseconds, respectively.

The step rate is important only if you will be using the disk in Drive 0 to start up the system. Keep in mind that too low a step rate may keep the disk from booting.

Changing the WAIT Value

The WAIT parameter compensates for hardware incompatibility between certain disk drives. The only time you should use it is when *all* tracks above a certain point during a FORMAT operation are shown as locked out when the FORMAT is verified.

The value assigned to WAIT signifies the amount of time between the arrival of the drive head at the location for a read or write, and the actual start of the read or write.

If you want to change the WAIT value, specify the new value on the FORMAT command line as follows:

FORMAT :drive (WAIT = number)

number is a value between 5000 and 50000. The exact value depends on the particular disk drive you are using. We recommend that you use a value around 25000 at first. Adjust this value higher if tracks are still locked out, or lower until the bottom limit is determined.

Logging in a Diskette

LOG is a utility program that logs in the directory track, number of sides, and density of a diskette. The syntax is:

LOG :drive

drive is any drive currently enabled in the system.

The LOG utility provides a way to log in diskette information and update the drive's Drive Code Table (DCT). It performs the same log-in function as the DEVICE library command, except for a single drive rather than all drives. It also provides a way to swap the Drive 0 diskette for a double-sided diskette.

The LOG :0 command prompts you to switch the Drive 0 diskette. You must use this command when switching between double- and single-sided diskettes in Drive 0. Otherwise, it is not needed.

Example

If you want to switch disks in Drive 0, type:

LOG :0 **ENTER**

The system prompts you with the message:

Exchange disks and hit <ENTER>

Remove the current disk from Drive 0 and insert the new system disk. When you press **ENTER**, information about the new disk is entered to the system.

Printing Graphics Characters

If your printer is capable of directly reproducing the TRS-80 graphics characters, you can use the SYSTEM (GRAPHIC) command. Once you have issued this command, any graphics characters on the screen will be sent to the line printer during a screen print. (Pressing **CTRL** **C** causes the contents of the video display to be printed on the printer.)

Do not use this command unless your printer is capable of directly reproducing the TRS-80 graphics characters.

Changing the Clock Rate

The system normally runs at the fast clock rate of 4 megahertz.

A slow mode of 2 megahertz is available, and may be necessary for real time-dependent programs. (This slow rate is the same as the Model III clock rate.)

To switch to the slow rate, enter the following command:

SYSTEM (SLOW)

To switch back to the fast rate, enter:

SYSTEM (FAST)



Appendix A/TRSDOS Error Messages

If the computer displays one of the messages listed in this appendix, an operating system error occurred. Any other error message may refer to an application program error, and you should check your application program manual for an explanation.

When an error message is displayed:

- Try the operation several times.
- Look up operating system errors below and take any recommended actions. (See your application program manual for explanations of application program errors.)
- Try using other diskettes.
- Reset the computer and try the operation again.
- Check all the power connections.
- Check all interconnections.
- Remove all diskettes from drives, turn off the computer, wait 15 seconds, and turn it on again.
- If you try all these remedies and still get an error message, contact a Radio Shack Service Center.

Note: If there is more than one thing wrong, the computer might wait until you correct the first error before displaying the second error message.

This list of error messages is alphabetical, with the binary and hexadecimal error numbers in parentheses. Following it is a quick reference list of the messages arranged in numerical order.

Attempted to read locked/deleted data record (Error 7, X'07')

In a system that supports a "deleted record" data address mark, an attempt was made to read a deleted sector. TRSDOS currently does not use the deleted sector data address mark. Check for an error in your application program.

Attempted to read system data record (Error 6, X'06')

An attempt was made to read a directory cylinder sector without using the directory read routines. Directory cylinder sectors are written with a data address mark that differs from the data sector's data address mark. Check for an error in your application program.

Data record not found during read (Error 5, X'05')

The sector number for the read operation is not on the cylinder being referenced. Either the disk is flawed, you requested an incorrect number, or the cylinder is improperly formatted. Try the operation again. If it fails, use another disk. Reformatting the old disk should lock out the flaw.

Data record not found during write (Error 13, X'0D')

The sector number requested for the write operation cannot be found on the cylinder being referenced. Either the disk is flawed, you requested an incorrect number, or the cylinder is improperly formatted. Try the operation again. If it fails, use another disk.

Device in use (Error 39, X'27')

A request was made to REMOVE a device (delete it from the Device Control Block tables) while it was in use. RESET the device in use before removing it.

Device not available (Error 8, X'08')

A reference was made for a logical device that cannot be found in the Device Control Block. Probably, your device specification was wrong or the device peripheral was not ready. Use the DEVICE command to display all devices available to the system.

Directory full — can't extend file (Error 30, X'1E')

A file has all extent fields of its last directory record in use and must find a spare directory slot but none is available. (See the "Directory Records" section.) Copy the disk's files to a newly formatted diskette to reduce file fragmentation. You may use backup by class or backup reconstruct to reduce fragmentation.

Directory read error (Error 17, X'11')

A disk error occurred during a directory read. The problem may be media, hardware, or program failure. Move the disk to another drive and try the operation again.

Directory write error (Error 18, X'12')

A disk error occurred during a directory write to disk. The directory may no longer be reliable. If the problem recurs, use a different diskette.

Disk space full (Error 27, X'1B')

While a file was being written, all available disk space was used. The disk contains only a partial copy of the file. Write the file to a diskette that has more available space. Then, REMOVE the partial copy to recover disk space.

End of file encountered (Error 28, X'1C')

You tried to read past the end of file pointer. Use the DIR command to check the size of the file. This error also occurs when you use the @PEOF supervisor call to successfully position to the end of a file. Check for an error in your application program.

Extended error (Error 63)

An error has occurred and the extended error code is in the HL register pair.

File access denied (Error 25, X'19')

You specified a password for a file that is not password protected or you specified the wrong password for a file that is password protected.

File already open (Error 41, X'29')

You tried to open a file for UPDATE level or higher, and the file already is open with this access level or higher. This forces a change to READ access protection. Use the RESET library command to close the file.

File not in directory (Error 24, X'18')

The specified filespec cannot be found in the directory. Check the spelling of the filespec.

File not open (Error 38, X'26')

You requested an I/O operation on an unopened file. Open the file before access.

GAT read error (Error 20, X'14')

A disk error occurred during the reading of the Granule Allocation Table. The problem may be media, hardware, or program failure. Move the diskette to another drive and try the operation again.

GAT write error (Error 21, X'15')

A disk error occurred during the writing of the Granule Allocation Table. The GAT may no longer be reliable. If the problem recurs, use a different drive or different diskette.

HIT read error (Error 22, X'16')

A disk error occurred during the reading of the Hash Index Table. The problem may be media, hardware, or program failure. Move the diskette to another drive and try the operation again.

HIT write error (Error 23, X'17')

A disk error occurred during the writing of the Hash Index Table. The HIT may no longer be reliable. If the problem recurs, use a different drive or different diskette.

Illegal access attempted to protected file (Error 37, X'25')

The USER password was given for access to a file, but the requested access required the OWNER password. (See the ATTRIB library command in your *Disk System Owner's Manual*.)

Illegal drive number (Error 32, X'20')

The specified disk drive is not included in your system or is not ready for access (no diskette, non-TRSDOS diskette, drive door open, and so on). See the DEVICE command in your *Disk System Owner's Manual*.

Illegal file name (Error 19, X'13')

The specified filespec does not meet TRSDOS filespec requirements. See your *Disk System Owner's Manual* for proper filespec syntax.

Illegal logical file number (Error 16, X'10')

A bad Directory Entry Code (DEC) was found in the File Control Block (FCB). This usually indicates that your program has altered the FCB improperly. Check for an error in your application program.

Load file format error (Error 34, X'22')

An attempt was made to load a file that cannot be loaded by the system loader. The file was probably a data file or a BASIC program file.

Lost data during read (Error 3, X'03')

During a sector read, the CPU did not accept a byte from the Floppy Disk Controller (FDC) data register in the time allotted. The byte was lost. This may indicate a hardware problem with the drive. Move the diskette to another drive and try again. If the error recurs, try another diskette.

Lost data during write (Error 11, X'0B')

During a sector write, the CPU did not transfer a byte to the Floppy Disk Controller (FDC) in the time allotted. The byte was lost; it was not transferred to the disk. This may indicate a hardware problem with the drive. Move the diskette to another drive and try again. If the error recurs, try another diskette.

LRL open fault (Error 42, X'2A')

The logical record length specified when the file was opened is different than the LRL used when the file was created. COPY the file to another file that has the specified LRL.

No device space available (Error 33, X'21')

You tried to SET a driver or filter and all of the Device Control Blocks were in use. Use the DEVICE command to see if any non-system devices can be removed to provide more space. This error also occurs on a "global" request to initialize a new file (that is, no drive was specified), if no file can be created.

No directory space available (Error 26, X'1A')

You tried to open a new file and no space was left in the directory. Use a different disk or REMOVE some files that you no longer need.

No error (Error 0)

The @ERROR supervisor call was called without any error condition being detected. A return code of zero indicates no error. Check for an error in your application program.

Parameter error (Error 44, X'2C')

(Under Version 6.2 only) An error occurred while executing a command line or utility because a parameter that does not exist was specified. Check the spelling of the parameter name, value, or abbreviation.

Parity error during header read (Error 1, X'01')

During a sector I/O request, the system could not read the sector header successfully. If this error occurs repeatedly, the problem is probably media or hardware failure. Try the operation again, using a different drive or diskette.

Parity error during header write (Error 9, X'09')

During a sector write, the system could not write the sector header satisfactorily. If this error occurs repeatedly, the problem is probably media or hardware failure. Try the operation again, using a different drive or diskette.

Parity error during read (Error 4, X'04')

An error occurred during a sector read. Its probable cause is media failure or a dirty or faulty disk drive. Try the operation again, using a different drive or diskette.

Parity error during write (Error 12, X'0C')

An error occurred during a sector write operation. Its probable cause is media failure or a dirty or faulty disk drive. Try the operation again, using a different drive or diskette.

Program not found (Error 31, X'1F')

The file cannot be loaded because it is not in the directory. Either the filespec was misspelled or the disk that contains the file was not loaded.

Protected system device (Error 40, X'28')

You cannot REMOVE any of the following devices: *KI, *DO, *PR, *JL, *SI, *SO. If you try, you get this error message.

Record number out of range (Error 29, X'1D')

A request to read a record within a random access file (see the @POSN supervisor call) provided a record number that was beyond the end of the file. Correct the record number or try again using another copy of the file.

Seek error during read (Error 2, X'02')

During a read sector disk I/O request, the cylinder that should contain the sector was not found within the time allotted. (The time is set by the step rate specified in the Drive Code Table.) Either the cylinder is not formatted or it is no longer readable, or the step rate is too low for the hardware to respond. You can set an appropriate step rate using the SYSTEM library command. The problem may also be caused by media or hardware failure. In this case, try the operation again, using a different drive or diskette.

Seek error during write (Error 10, X'0A')

During a sector write, the cylinder that should contain the sector was not found within the time allotted. (The time is set by the step rate specified in the Drive Code Table.) Either the cylinder is not formatted or it is no longer readable, or the step rate is too low for the hardware to respond. You can set an appropriate step rate using the SYSTEM library command. The problem may also be caused by media or hardware failure. In this case, try the operation again, using a different drive or diskette.

— Unknown error code

The @ERROR supervisor call was called with an error number that is not defined. Check for an error in your application program.

Write fault on disk drive (Error 14, X'0E')

An error occurred during a write operation. This probably indicates a hardware problem. Try a different diskette or drive. If the problem continues, contact a Radio Shack Service Center.

Write protected disk (Error 15, X'0F')

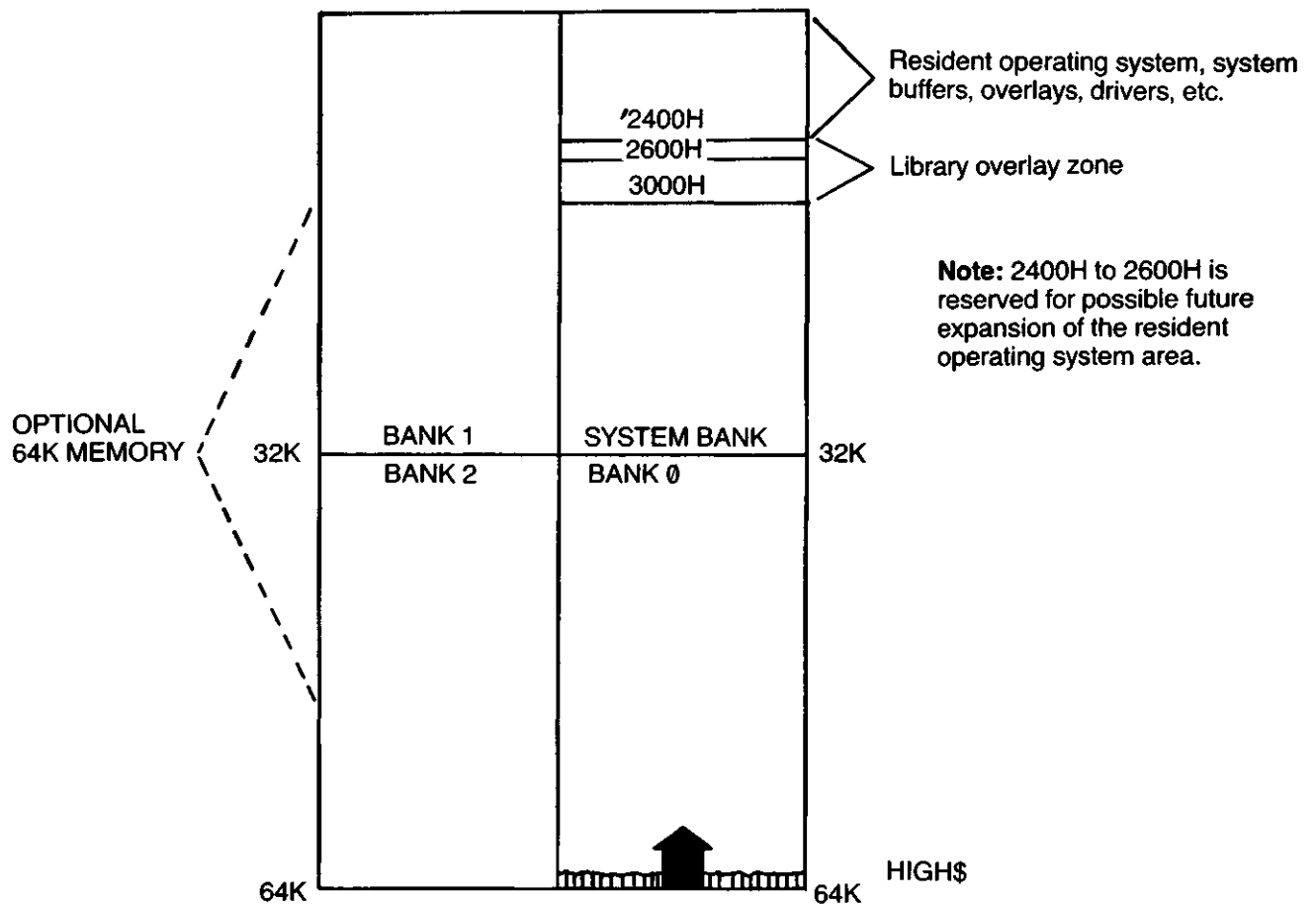
You tried to write to a drive that has a write-protected diskette or is software write-protected. Remove the write-protect tab, if the diskette has one. If it does not, use the DEVICE command to see if the drive is set as write protected. If it is, you can use the SYSTEM library command with the (WP = OFF) parameter to write enable the drive. If the problem recurs, use a different drive or different diskette.

Numerical List of Error Messages

Decimal	Hex	Message
0	X'00'	No Error
1	X'01'	Parity error during header read
2	X'02'	Seek error during read
3	X'03'	Lost data during read
4	X'04'	Parity error during read
5	X'05'	Data record not found during read
6	X'06'	Attempted to read system data record
7	X'07'	Attempted to read locked/deleted data record
8	X'08'	Device not available
9	X'09'	Parity error during header write
10	X'0A'	Seek error during write
11	X'0B'	Lost data during write
12	X'0C'	Parity error during write
13	X'0D'	Data record not found during write
14	X'0E'	Write fault on disk drive
15	X'0F'	Write protected disk
16	X'10'	Illegal logical file number
17	X'11'	Directory read error
18	X'12'	Directory write error
19	X'13'	Illegal file name
20	X'14'	GAT read error
21	X'15'	GAT write error
22	X'16'	HIT read error
23	X'17'	HIT write error
24	X'18'	File not in directory
25	X'19'	File access denied
26	X'1A'	No directory space available
27	X'1B'	Disk space full
28	X'1C'	End of file encountered
29	X'1D'	Record number out of range
30	X'1E'	Directory full—can't extend file
31	X'1F'	Program not found
32	X'20'	Illegal drive number
33	X'21'	No device space available
34	X'22'	Load file format error
37	X'25'	Illegal access attempted to protected file
38	X'26'	File not open
39	X'27'	Device in use
40	X'28'	Protected system device

41	X'29'	File already open
42	X'2A'	LRL open fault
43	X'2B'	SVC parameter error
44	X'2C'	Parameter error
63	X'3F'	Extended error
—		Unknown error code

Appendix B/Memory Map



All software must observe HIGH\$.

User software which does not allow TRSDOS library commands to be executed during run time may use memory from 2600H to HIGH\$.

User software which allows for library commands during execution must reside in and use memory only between 3000H and HIGH\$.

TRSDOS provides all functions and storage through supervisor calls. No address or entry point below 3000H is documented by Radio Shack.



Appendix C/Character Codes

Text, control functions, and graphics are represented in the computer by codes. The character codes range from zero through 255.

Codes one through 31 normally represent certain control functions. For example, code 13 represents a carriage return or "end of line." These same codes also represent special characters. To display the special character that corresponds to a particular code (1-31), precede the code with a code zero.

Codes 32 through 127 represent the text characters — all those letters, numbers, and other characters that are commonly used to represent textual information.

Codes 128 through 191, when output to the video display, represent 64 graphics characters.

Codes 192 through 255, when output to the video display, represent either space compression codes or special characters, as determined by software.

ASCII Character Set

Code		ASCII Abbrev.	Keyboard	Video Display
Dec.	Hex.			
0	00	NUL	CTRL @	Treat next character as displayable; if in the range 1-31, a special character is displayed (see list of special characters later in this Appendix).
1	01	SOH	CTRL A	
2	02	STX	CTRL B	
3	03	ETX	CTRL C	
4	04	EOT	CTRL D	
5	05	ENQ	CTRL E	
6	06	ACK	CTRL F	
7	07	BEL	CTRL G	
8	08	BS	CTRL H	Backspace and erase
9	09	HT	CTRL I	
10	0A	LF	CTRL J	Move cursor to start of next line
11	0B	VT	CTRL K	
12	0C	FF	CTRL L	
13	0D	CR	ENTER	Move cursor to start of next line
14	0E	SO	CTRL M	Turn cursor on
15	0F	SI	CTRL N	Turn cursor off
16	10	DLE	CTRL O	Enable reverse video and set high bit routine on*
17	11	DC1	CTRL P	Set reverse video high bit routine off*
18	12	DC2	CTRL Q	
19	13	DC3	CTRL R	
20	14	DC4	CTRL S	
21	15	NAK	CTRL T	Swap space compression/special characters
22	16	SYN	CTRL U	Swap special/alternate characters
23	17	ETB	CTRL V	Set to 40 characters per line
24	18	CAN	SHIFT CTRL Q	Backspace without erasing
25	19	EM	CTRL X	
26	1A	SUB	SHIFT CTRL P	Advance cursor
27	1B	ESC	CTRL Y	
28	1C	FS	SHIFT CTRL N	Move cursor down
29	1D	GS	CTRL Z	
30	1E	RS	SHIFT CTRL O	Move cursor up
			CTRL [
			CTRL /	Move cursor to upper left corner. Disable reverse video and set high bit routine off.* Set to 80 characters per line.
			CTRL ENTER	Erase line and start over
			CTRL .	
			CTRL ;	Erase to end of line

*When the high bit routine is on, characters 128 through 191 are displayed as standard ASCII characters in reverse video.

Code		ASCII Abbrev.	Keyboard	Video Display
Dec.	Hex.			
31	1F	VS SPA	SHIFT CLEAR	Erase to end of display
32	20		SPACEBAR	(blank)
33	21		1	!
34	22		"	"
35	23		#	#
36	24		\$	\$
37	25		%	%
38	26		&	&
39	27		'	'
40	28		((
41	29))
42	2A		*	*
43	2B		+	+
44	2C		,	,
45	2D		-	-
46	2E		.	.
47	2F		/	/
48	30		0	0
49	31		1	1
50	32		2	2
51	33		3	3
52	34		4	4
53	35		5	5
54	36		6	6
55	37		7	7
56	38		8	8
57	39		9	9
58	3A		:	:
59	3B		;	;
60	3C		<	<
61	3D		=	=
62	3E		>	>
63	3F		?	?
64	40		@	@
65	41		SHIFT A	A
66	42		SHIFT B	B
67	43		SHIFT C	C
68	44		SHIFT D	D
69	45		SHIFT E	E
70	46		SHIFT F	F
71	47		SHIFT G	G
72	48		SHIFT H	H
73	49		SHIFT I	I
74	4A		SHIFT J	J
75	4B		SHIFT K	K
76	4C		SHIFT L	L
77	4D		SHIFT M	M
78	4E		SHIFT N	N
79	4F		SHIFT O	O
80	50		SHIFT P	P
81	51		SHIFT Q	Q
82	52		SHIFT R	R
83	53		SHIFT S	S
84	54		SHIFT T	T
85	55		SHIFT U	U
86	56		SHIFT V	V
87	57		SHIFT W	W
88	58		SHIFT X	X
89	59		SHIFT Y	Y

Code		ASCII Abbrev.	Keyboard	Video Display
Dec.	Hex.			
90	5A		SHIFT Z	Z
91	5B		CLEAR ,	
92	5C		CLEAR /	\
93	5D		CLEAR .	
94	5E		CLEAR :	^
95	5F		CLEAR ENTER	—
96	60		SHIFT @	
97	61		A	a
98	62		B	b
99	63		C	c
100	64		D	d
101	65		E	e
102	66		F	f
103	67		G	g
104	68		H	h
105	69		I	i
106	6A		J	j
107	6B		K	k
108	6C		L	l
109	6D		M	m
110	6E		N	n
111	6F		O	o
112	70		P	p
113	71		Q	q
114	72		R	r
115	73		S	s
116	74		T	t
117	75		U	u
118	76		V	v
119	77		W	w
120	78		X	x
121	79		Y	y
122	7A		Z	z
123	7B		CLEAR SHIFT ,	{
124	7C		CLEAR SHIFT /	
125	7D		CLEAR SHIFT .	}
126	7E		CLEAR SHIFT :	~
127	7F	DEL	CLEAR SHIFT ENTER	±

Extended (non-ASCII) Character Set

Code		Keyboard	Video Display
Dec.	Hex.		
128	80	BREAK	
129	81	F1	
		CLEAR CTRL A	
130	82	F2	
		CLEAR CTRL B	
131	83	F3	
		CLEAR CTRL C	
132	84	CLEAR CTRL D	
133	85	CLEAR CTRL E	
134	86	CLEAR CTRL F	
135	87	CLEAR CTRL G	
136	88	CLEAR CTRL H	
137	89	CLEAR CTRL I	
138	8A	CLEAR CTRL J	
139	8B	CLEAR CTRL K	
140	8C	CLEAR CTRL L	
141	8D	CLEAR CTRL M	
142	8E	CLEAR CTRL N	
143	8F	CLEAR CTRL O	
144	90	CLEAR CTRL P	
145	91	SHIFT F1	
		CLEAR CTRL Q	
146	92	SHIFT F2	
		CLEAR CTRL R	
147	93	SHIFT F3	
		CLEAR CTRL S	
148	94	CLEAR CTRL T	
149	95	CLEAR CTRL U	
150	96	CLEAR CTRL V	
151	97	CLEAR CTRL W	
152	98	CLEAR CTRL X	
153	99	CLEAR CTRL Y	
154	9A	CLEAR CTRL Z	
155	9B	CLEAR SHIFT ➡	
156	9C		
157	9D		
158	9E		
159	9F		
160	A0	CLEAR SPACE	
161	A1	CLEAR SHIFT 1	
162	A2	CLEAR SHIFT 2	
163	A3	CLEAR SHIFT 3	
164	A4	CLEAR SHIFT 4	
165	A5	CLEAR SHIFT 5	
166	A6	CLEAR SHIFT 6	
167	A7	CLEAR SHIFT 7	
168	A8	CLEAR SHIFT 8	
169	A9	CLEAR SHIFT 9	
170	AA	CLEAR SHIFT :	
171	AB		
172	AC		
173	AD	CLEAR -	
174	AE		
175	AF		
176	B0	CLEAR 0	
177	B1	CLEAR 1	
178	B2	CLEAR 2	

See graphics character table in this Appendix.

Code		Keyboard	Video Display
Dec.	Hex.		
179	B3	CLEAR 3	See graphics character table in this Appendix.
180	B4	CLEAR 4	
181	B5	CLEAR 5	
182	B6	CLEAR 6	
183	B7	CLEAR 7	
184	B8	CLEAR 8	
185	B9	CLEAR 9	
186	BA	CLEAR :	
187	BB		
188	BC		
189	BD	CLEAR SHIFT -	See list of special characters in this Appendix.
190	BE		
191	BF		
192	C0	CLEAR @*	
193	C1	CLEAR A**	
194	C2	CLEAR B**	
195	C3	CLEAR C**	
196	C4	CLEAR D**	
197	C5	CLEAR E**	
198	C6	CLEAR F**	
199	C7	CLEAR G**	
200	C8	CLEAR H**	
201	C9	CLEAR I**	
202	CA	CLEAR J**	
203	CB	CLEAR K**	
204	CC	CLEAR L**	
205	CD	CLEAR M**	
206	CE	CLEAR N**	
207	CF	CLEAR O**	
208	D0	CLEAR P**	
209	D1	CLEAR Q**	
210	D2	CLEAR R**	
211	D3	CLEAR S**	
212	D4	CLEAR T**	
213	D5	CLEAR U**	
214	D6	CLEAR V**	
215	D7	CLEAR W**	
216	D8	CLEAR X**	
217	D9	CLEAR Y**	
218	DA	CLEAR Z**	
219	DB		
220	DC		
221	DD		
222	DE		
223	DF		
224	E0	CLEAR SHIFT @	
225	E1	CLEAR SHIFT A	
226	E2	CLEAR SHIFT B	
227	E3	CLEAR SHIFT C	
228	E4	CLEAR SHIFT D	
229	E5	CLEAR SHIFT E	
230	E6	CLEAR SHIFT F	
231	E7	CLEAR SHIFT G	
232	E8	CLEAR SHIFT H	
233	E9	CLEAR SHIFT I	
234	EA	CLEAR SHIFT J	

*Empties the type-ahead buffer.

**Used by Keystroke Multiply, if KSM is active.







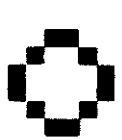
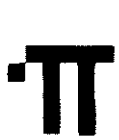
Code		Keyboard	Video Display
Dec.	Hex.		
235	EB	CLEAR SHIFT K	See list of special characters in this Appendix.
236	EC	CLEAR SHIFT L	
237	ED	CLEAR SHIFT M	
238	EE	CLEAR SHIFT N	
239	EF	CLEAR SHIFT O	
240	F0	CLEAR SHIFT P	
241	F1	CLEAR SHIFT Q	
242	F2	CLEAR SHIFT R	
243	F3	CLEAR SHIFT S	
244	F4	CLEAR SHIFT T	
245	F5	CLEAR SHIFT U	
246	F6	CLEAR SHIFT V	
247	F7	CLEAR SHIFT W	
248	F8	CLEAR SHIFT X	
249	F9	CLEAR SHIFT Y	
250	FA	CLEAR SHIFT Z	
253	FD		
254	FE		
255	FF		

Graphics Characters (Codes 128-191)





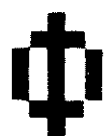
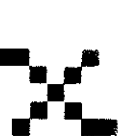


128	129	130	131	132	133	134	135
136	137	138	139	140	141	142	143
144	145	146	147	148	149	150	151
152	153	154	155	156	157	158	159
160	161	162	163	164	165	166	167
168	169	170	171	172	173	174	175
176	177	178	179	180	181	182	183
184	185	186	187	188	189	190	191

Special Characters (0-31, 192-255)







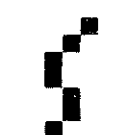

							
0	1	2	3	4	5	6	7
							
8	9	10	11	12	13	14	15
							
16	17	18	19	20	21	22	23
							
24	25	26	27	28	29	30	31
							
192	193	194	195	196	197	198	199
							
200	201	202	203	204	205	206	207









208 209 210 211 212 213 214 215

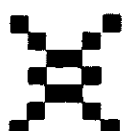

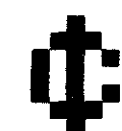




216 217 218 219 220 221 222 223





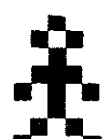
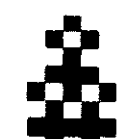
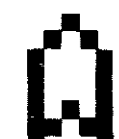
224 225 226 227 228 229 230 231

232 233 234 235 236 237 238 239

240 241 242 243 244 245 246 247

248 249 250 251 252 253 254 255

Appendix D/Keyboard Code Map

The keyboard code map shows the code that TRSDOS returns for each key, in each of the modes: control, shift, unshift, clear and control, clear and shift, clear and unshift.

For example, pressing **CLEAR**, **SHIFT**, and **1** at the same time returns the code X'A1'.

A program executing under TRSDOS — for example, BASIC — may translate some of these codes into other values. Consult the program's documentation for details.

BREAK Key Handling

The **BREAK** key (X'80') is handled in different ways, depending on the settings of three system functions. The table below shows what happens for each combination of settings.

Break Enabled	Break Vector Set	Type-Ahead Enabled	
Y	N	Y	If characters are in the type-ahead buffer, then the buffer is emptied.* If the type-ahead buffer is empty, then a BREAK character (X'80') is placed in the buffer.*
Y	N	N	A BREAK character (X'80') is placed in the buffer.
Y	Y	Y	The type-ahead buffer is emptied of its contents (if any), and control is transferred to the address in the BREAK vector (see @BREAK SVC).*
Y	Y	N	Control is transferred to the address in the BREAK vector (see @BREAK SVC).
N	X	X	No action is taken and characters in the type-ahead buffer are not affected.

*Because the **BREAK** key is checked for more frequently than other keys on the keyboard, it is possible for **BREAK** to be pressed after another key on the keyboard and yet be detected first.

Y means that the function is on or enabled

N means that the function is off or disabled

X means that the state of the function has no effect

Break is enabled with the SYSTEM (BREAK = ON) command (this is the default condition).

The break vector is set using the @BREAK SVC (normally off).

Type-ahead is enabled using the SYSTEM (TYPE = ON) command (this is the default condition).

B1	31	B2	32	B3	33	B4	34	B5	35	B6	36	B7	37	B8	38	B9	39	B0	30	BA	††	AD	2D	80	80															
A1	!	21	A2	"	22	A3	#	23	A4	\$	24	A5	%	25	A6	&	26	A7	'	27	A8	(28	A9)	29	A0	0	†	AA	:	2A	BD	=	3D	80	B	R	†††	
B1	31	B2	32	B3	33	B4	34	B5	35	B6	36	B7	37	B8	38	B9	39	B0	30	BA	3A	AD	2D	80	80	K	80													
8B	0B	91	11	97	17	85	05	92	12	94	14	99	19	95	15	89	09	8F	0F	90	10	0	0	88	08	89	09													
9B	↑	1B	Q	51	F7	W	57	E5	E	45	F2	R	52	F4	T	54	F9	Y	59	F5	U	55	E9	I	49	EF	O	4F	F0	P	50	E0	@	60	98	←	18	99	→	19
8B	0B	D1	71	D7	77	C5	65	D2	72	D4	74	D9	79	D5	75	C9	69	CF	6F	D0	70	C0	40	88	08	89	09													
8A	0A	81	01	93	13	84	04	86	06	87	07	88	08	8A	0A	8B	0B	8C	0C	1E	1E	8D	0D	CLEAR		1F														
9A	↓	1A	A	E1	41	F3	S	53	E4	D	44	E6	F	46	E7	G	47	E8	H	48	EA	J	4A	EB	K	4B	EC	L	4C	7E	;	2B	7F	ENTER	1D	0D				
8A	0A	C1	61	D3	73	C4	64	C6	66	C7	67	C8	68	CA	6A	CB	6B	CC	6C	5E	3B	5F	0D																	
SHIFT		9A	1A	98	18	83	03	96	16	82	02	8E	0E	8D	0D	1B	1B	1D	1D	1C	1C	SHIFT																		
		FA	Z	5A	X	E3	C	43	F6	V	E2	B	42	EE	N	4E	ED	M	4D	7B	<	3C	7D	>	3E	7C	?	3F												
		DA	7A	D8	78	C3	63	D6	76	C2	62	CE	6E	CD	6D	5B	2C	7D	.	3E	7C	/	3F																	
		C T R L		A0		A0		A0												00		C A P S																		
																				20																				
																				20																				

The keys may be positioned differently on your keyboard. However, they produce the same codes.

LEGEND:

Clear and Control	•	•	Control
-------------------	---	---	---------

Clear and Left Shift	•	•	Shift
Clear and Unshift	•	•	Unshift

Note: Pressing CONTROL, SHIFT, and @ at the same time generates an EOF (end of file) -- X'1C' with NZ return flag.

Whenever pressing CLEAR, SHIFT, and another key at the same time, be sure to use the left SHIFT key – not the right SHIFT key.

† Pressing SHIFT and Ø at the same time (or CAPS alone) turns the CAPS mode on or off.

†† Pressing CONTROL and : at the same time causes a screen print.

††† Pressing SHIFT and BREAK at the same time reselects the last drive.

Codes for these keys are the same as for the main keyboard.

81	81	82	82	83	83
91 F1 91	91	92 F2 92	92	93 F3 93	93
81	81	82	82	83	83
7		8		9	
4		5		6	
1		2		3	
∅		•		ENT	

Appendix E/Programmable SVCs

(Under Version 6.2 only)

SVC numbers 124 through 127 are reserved for programmer installable SVCs. To install an SVC the programmer must write the routine to execute when the SVC is called.

The routine should be written as high memory module if it is to be available at all times. If you execute a SYSGEN command when a programmable SVC is defined, the address of the routine is saved in the SYSGEN file and restored each time the system is configured. If the routine is a high memory module, the routine is saved and restored as well. This makes the SVC always available. For more information on high memory modules, see Memory Header and Sample Program F.

To install an SVC, the program must access the SVC table. The SVC table contains 128 two-byte positions, a two-byte position for each usable SVC. Each position in the table contains the address of the routine to execute when the SVC is called.

To access the SVC table, execute the @FLAGS SVC (SVC 101). IY + 26 contains the MSB of the SVC table start address. The LSB of the SVC table address is always 0 because the SVC table always begins on a page boundary.

Store the address of the routine to be executed at the SVC *number times 2* byte in the table. For example, if you are installing SVC 126, store the address of the routine at byte 252 in the table. Addresses are stored in LSB-MSB format.

When the SVC is executed, control is transferred to the address in the table. On entry to your SVC, Register A contains the same value as Register C. All other registers retain the values they had when the RST 28 SVC instruction was executed.

To exit the SVC, execute a RET instruction. The program should save and restore any registers used by the SVC.

Initially, SVCs 124 through 127 display an error message when they are executed. When installing an SVC you should save the original address at that location in the table and restore it when you remove the SVC.

These program lines insert a new SVC into the system SVC table, save the previous value of the table, and reinsert that value before execution ends. You could check the existing value to see if the address is above X'2600'. If it is, the SVC is already assigned and should not be used at this time.

This code inserts SVC 126, called MYSVC:

LD	A,@FLAGS	;Locate start of SVC table
RST	28H	;Execute @FLAGS SVC
LD	H,(IY + 26)	;Get MSB of address
LD	L,126*2	;Want to use SVC 126
LD	(OSVC126A),HL	;Save address of SVC entry
LD	E,(HL)	;Get current SVC address
INC	HL	
LD	D,(HL)	
LD	(OSVC126V),DE	;Save the old value
DEC	HL	
LD	DE,MYSVC	;Get address of routine for
		;SVC 126
LD	(HL),E	;Insert new SVC address into
		;table
INC	HL	

LD (HL),D

.
. Code that uses MYSVC (SVC 126)
.
.

This code removes SVC 126:

LD	HL,(OSVC126A)	;Get address of SVC entry
LD	DE,(OSVC126V)	;Get original value
LD	(HL),E	;Insert original SVC address
INC	HL	
LD	(HL),D	

Appendix F/Using SYS13/SYS

(Under Version 6.2 only)

With TRSDOS Version 6.2, you can create an Extended Command Interpreter (ECI) or an Immediate Execution Program (IEP). TRSDOS can store either an ECI or IEP in the SYS13 file. Both programs cannot be present at the same time.

At the TRSDOS Ready prompt when you type ☐ **ENTER**, TRSDOS executes the program stored in SYS13/SYS. Because TRSDOS recognizes the program as a system file, TRSDOS includes the file when creating backups and loads the program faster.

If you want to write additional commands for TRSDOS, you can write an interpreter to execute these commands. Your ECI can also execute TRSDOS commands by using the @CMNDI SVC to pass a command to the TRSDOS interpreter.

If EFLAG\$ contains a non-zero value, TRSDOS executes the program in SYS13/SYS. If EFLAG\$ contains a zero, TRSDOS uses its own command interpreter.

Sample Program G is an example of an ECI. It is important to note that your ECI must be executable by pressing ☐ **ENTER** at the TRSDOS Ready prompt.

An ECI can use all of memory or you can restrict it to use the system overlay area (X'2600' to X'2FFF').

To implement an IEP or ECI, use the following syntax:

```
COPY filespec SYS13/SYS.LSIDOS:drive (C = N) ENTER
```

filespec can be any executable (/CMD) program file. *drive* specifies the destination drive. The destination drive must contain an original SYS13/SYS file.

Example

```
COPY SCRIPSIT/CMD:1 SYS13/SYS.LDI:0 (C = N)
```

TRSDOS copies SCRIPSIT/CMD from Drive 1 to SYS13/SYS in Drive 0. At the TRSDOS Ready prompt, when you press ☐ **ENTER**, TRSDOS executes SCRIPSIT.



Index

Subject	Page	Subject	Page
@ABORT	48	interfacing to device drivers	42-44
Access		Cylinder	
device	9-10	highest numbered	12
drive	11-21	number of	18
file	4	position, current	12
@ADTSK	49	starting	25
Alien disk controller	12	@DATE	67
Allocation		@DCINIT	68
dynamic	3	@DCRES	69
information	12, 25	@DCSTAT	70
methods of	3	DEBUG	6
pre-	3	@DEBUG	71
unit of	2	@DECHEX	72
ASCII codes	202-04	Density, double and single	1, 11, 18
Background tasks, invoking	33-34	Device	
@BANK	37-39	access	9-10
Bank switching	36-39	handling	27
@BKSP	52	NIL	9
BOOT/SYS	5	Device Control Block (DCB)	9
BREAK		Device driver	7, 8, 13
detection	29-32, 53	address	9
key handling	211	COM	43-44
@BREAK	53	@CTL interfacing to	42-44
Byte I/O	40-42	keyboard	43
Characters		printer	43
ASCII	202-04	templates	40-42
codes	201-10	video	43
graphics	205-06, 208	Devspec	9
special	206-07, 209-10	Directory	
@CHNIO	54	location on disk	2, 12
@CKDRV	55	primary and extended entries	14
@CKBRKC	55	16, 20	
@CKEOF	56	record, locating a	20
@CKTSK	57	records (DIREC)	13-16
Clock rate, changing	192	sectors, number of	14
@CLOSE	60	Directory Entry Code (DEC)	18-19
@CLS	61	20, 24	
@CMNDI	63	@DIRRD	73
@CMNDR	64	DIR/SYS	5
Codes		@DIRWR	74
ASCII	202-04	Disk, diskette	
character	201-10	controller	12
error	197	double-sided	11-12, 17, 18
graphics	205-06, 208	files	13-14
keyboard	211-12	floppy	1
return	28	formatting	17, 18
special character	206-07, 209-10	hard	2
Converting to TRSDOS Version 6	27-28	I/O table	13
CREATED files	15	minimum configuration	7-8
@CTL	40-42, 65-66	name	18

Index

Subject	Page	Subject	Page
organization	1-2	contents of	16-18
single-sided	11-12, 17, 18	Graphics	
space, available	2	characters, printing	190
@DIV8	75	codes	205-06, 208
@DIV16	76	@GTDCB	91
@DODIR	77-78	@GTDCT	92
Drive		@GTMOD	93
access	11-22	Guidelines, programming	27-44
address	12	Hash code	15, 18
floppy	1, 11	Hash Index Table (HIT)	
hard	2, 11	location on disk	2
size	11	explanation of	18-19
Drive Code Table DCT	11-13	@HDFMT	94
Driver — see Device driver		@HEXDEC	95
@DSP	79	@HEX8	96
@DSPLY	80	@HEX16	97
End of File (EOF)	15	@HIGH\$	98
Ending Record Number (ERN)	16, 25	@ICNFG, interfacing to	32-33
ENTER detection	29-32	Immediate Execution Program	215
Error		@INIT	99
codes and messages	193-197	Initialization configuration	
dictionary	6	vector	32-33
@ERROR	81	Interrupt tasks	34-36
@EXIT	82	@IPL	100
Extended Command Interpreter	84, 215	Job Control Language (JCL)	6, 28
@FEXT	83	@KBD	101
File		@KEY	102
access	4	Keyboard codes	211-12
descriptions, TRSDOS	5-8	@KEYIN	103
modification	15	KFLAG\$	29
File Control Block (FCB)	23	@KITSK, interfacing to	33-34
Files		@KLTSK	104
CREATED	15	Library commands	28
device driver	7	technical information on	189-91
filter	7	@LOAD	105
system (/SYS)	5-6, 7-8, 19	@LOC	106
utility	7	@LOF	107
Filter templates	40-42	LOG utility	190
Filters	7, 8, 40-42	@LOGGER	108
example of	42	Logical Record Length (LRL)	15, 24
FLAGS	28, 84-86	@LOGOT	109
@FNAME	87	Memory banks — see RAM banks	
@FSPEC	89	Memory header	10, 27
@GET	40-42, 90	Memory map	199
Gran, granule		Minimum configuration disk	7
allocation information	25	Modification date	15
definition	2, 17	@MSG	110
per track	1-2, 12	@MUL8	111
Granule Allocation Table (GAT)		@MUL16	112
location on disk	2	Next Record Number (NRN)	24

Index

Subject	Page	Subject	Page
NIL device	9	C	168
@OPEN	113	D	175
Overlays, system	5-6, 19	E	177
@PARAM	114-15	F	178
Password		G	187
for TRSDOS files	8	Sectors	
protection levels	14, 24	per cylinder	14, 19
@PAUSE	116	per granule	1-2, 12
PAUSE detection	29-32	@SEEK	138
@PEOF	117	@SEEKSC	139
@POSN	118	@SKIP	140
@PRINT	119	@SLCT	141
Printing Graphics Characters	190	@SOUND	142
Programming Guidelines	27-44	Special Character Codes	206-07, 209-10
Protection Levels	14, 24, 27	Stack handling	28
@PRT	120	Step rate	11
@PUT	40-42, 121	changing	189
RAM Banks		@STEPI	143
switching	36-39	Supervisor calls (SVCs)	
use of	50-51	calling procedure	45
@RAMDIR	122	lists of	46-47, 155-57, 158-59
@RDHDR	123	program entry and	
@RDSEC	124	return conditions	45
@RDSSC	125	sample programs using	160-183
@RDTRK	126	using	45-183
@READ	127	SYS files	5-6, 7-8, 19
Record		System	
length	3-4, 15, 24	files	5-6, 7-8, 19
logical and physical	3-4	overlays	5-6, 19
numbers	4	Task	
processing	4	interrupt level, adding	49
spanning	3-4	slots	34, 35, 49
@REMOV	128	Task Control Block (TCB)	34, 35, 49
@RENAM	129	Vector Table (TCBVT)	34, 35
Restart Vectors (RSTs)	29	Task processor, interfacing to	34-36
Return Code (RC)	28	@TIME	144
@REW	130	TRSDOS	
@RMTSK	131	converting to Version 6	27-28
@RPTSK	132	error messages and codes	193-97
@RREAD	133	file descriptions	5-8
RS-232		technical information on	
initializing	32	commands and utilities	189-91
COM driver for	43-44	TYPE code	23
@RSLCT	134	@VDCTL	145-46
@RSTOR	135	@VER	147
@RUN	136	Version, operating system	17
@RWRIT	137	Visibility	14
Sample Programs	160-83	@VRSEC	148
A	161	WAIT value, changing	190
B	163	@WEOF	149

Index

Subject	Page	Subject	Page
@WHERE	150	@WRSEC	152
@WRITE	151	@WRSSC	153
Write Protect	9	@WRTRK	154

Index

Subject	Page	Subject	Page

RADIO SHACK, A DIVISION OF TANDY CORPORATION

U.S.A.: FORT WORTH, TEXAS 76102
CANADA: BARRIE, ONTARIO L4M 4W5

TANDY CORPORATION

AUSTRALIA

**91 KURRAJONG AVENUE
MOUNT DRUITT, N.S.W. 2770**

BELGIUM

**PARC INDUSTRIEL
5140 NANINNE (NAMUR)**

U. K.

**BILSTON ROAD WEDNESBURY
WEST MIDLANDS WS10 7JN**

S-L/3-85

Printed in U.S.A.