TEXAS INSTRUMENTS

TM 990

TM 990/303A Floppy- Disk Controller

MICROPROCESSOR SERIES™

User's Guide

TABLE OF CONTENTS

TABLE OF CONTENTS

APPENDICES

LIST OF ILLUSTRATIONS

LIST OF ILLUSTRATIONS

LIST OF TABLES

SECTION 1

INTRODUCTION

1.1 GENERAL

This manual covers the installation, operation, and theory of operation of the TM 990/303A floppy disk controller module, shown in Figure 1-1. Figure 1-2 is a block diagram of the board. This board provides a controlling interface between a microcomputer such as the TM 990/101MA and the following disk drives:

- Shugart model SA 400 (mini)
- Shugart model SA 800 (standard)
- CDC model 9404B (standard)
- Qume model DT-8 (standard)

Note that the TM 990/303A is used with only these models (this does not include other models in a series such as the Shugart 801). The TM 990/303A can be used with the TM 990/100M, TM 990/100MA, or TM 990/101MA microcomputer modules. However, because of buffering on the TM 990/100M and TM 990/100MA boards, the controller cannot do DMA (direct memory access) with the memory on these boards, only with an expansion memory board used with the TM 990/100M or TM 990/100MA.

If the TM 990/303A is to be used with the TM 990/101M module, the PCB must be modified to the equivalent of the TM 990/101MA microcomputer module. All TM 990/101M modules returned to the factory for repair will automatically be updated.

The PCB part number will show if the module has been modified to the correct revision level. This part number is found on the conductor side of the board. A part number of 994725-1 A (B) or 994725-1 B (or higher letter) indicates a PCB that has been modified to the proper revision level. A part number such as 994725-1 A indicates a PCB not modified to the proper revision level.

For users of any of the three disk drives, Table 1-1 illustrates the diskette formats used with their corresponding disk drives.

TABLE 1-1. DISKETTE FORMATS AND CORRESPONDING DISK DRIVES

Diskette Formats	Disk Drive Formats			
	CDC 9404B	SA 400	SA 800	Qume DT-8
IBM Single Density/1 Side	X	X	X	X
IBM Double Density/1 Side		X	X	X
TI Double Density/1 Side			X	X
IBM Single Density/2 Sides				X
IBM Double Density/2 Sides				X
TI Double Density/2 Sides				X

1.2 FEATURES

The TM 990/303A floppy disk controller has the following features:

- Formats supported:

  - IBM single density format
  - IBM double density format
  - TI Digital Systems Group (DSG) double density format (TILINE floppy controller)

- Disk sizes: Standard or mini

- Disk sides:

  - One side only on Shugart and CDC
  - Two sides on Qume DataTrak 8

- Number of disk drives (daisy chained): Four maximum for standard size, and three maximum for mini size.

- Recording methods:

  - Single density frequency modulation (FM)
  - Double density modified frequency modulation (MFM)

- Data format:

  - IBM 3740 compatible
  - TI FS 990 compatible

- System interface:

  - CRU (controller initialization)
  - DMA transfer (data and commands)

- Three LED's indicate controller status

- Bootstrap load feature can be used to initialize system from diskette.

- Controller firmware (see below) provided on two TMS 2716's (2 K words); controller firmware EPROM space expandable to 4 K words by using two TMS 2532's.

Software on the controller includes the following features:

- Seventeen commands including controller self test, read and write to/from diskette and host memory, read to and write from controller and host RAM, bootstrap load from diskette software, format diskette, execute program in controller memory, read status of specified drive.

- Command completion interrupt to host (interrupt level jumper selectable); completion status reported to host.

- Controller call through interrupt via CRU.

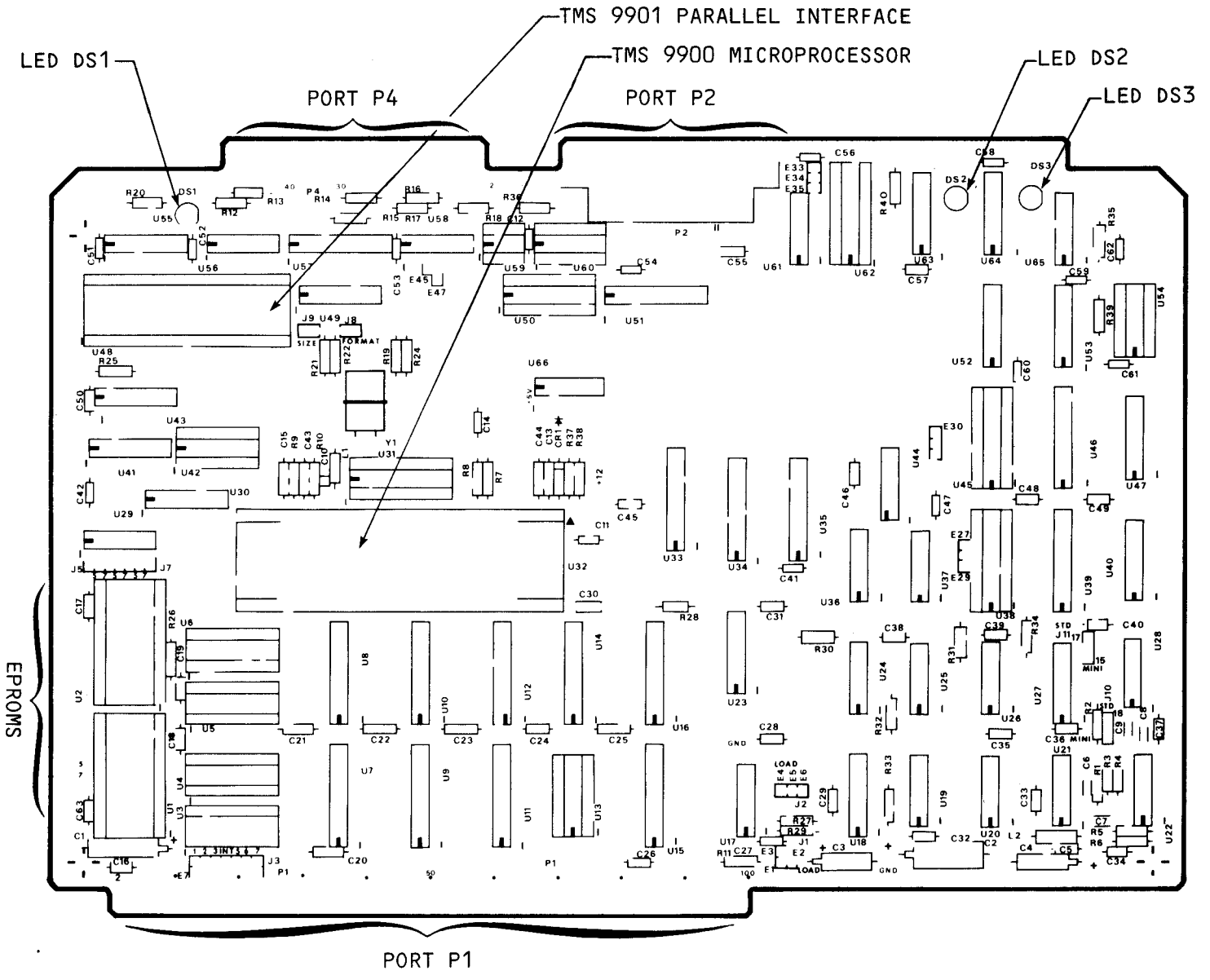


FIGURE 1-1. TM 990/303A FLOPPY DISK CONTROLLER MODULE

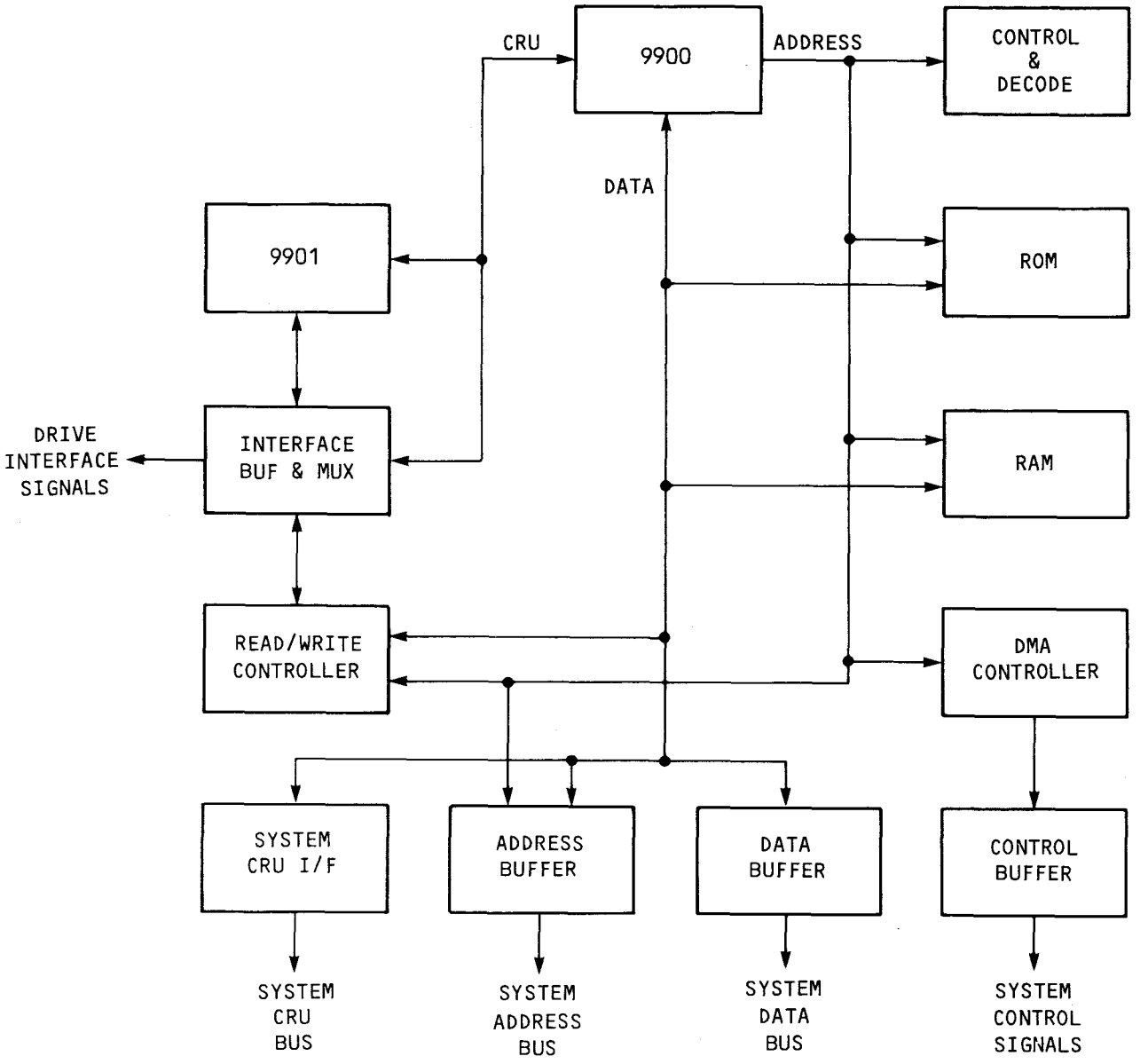


FIGURE 1-2. TM 990/303A BLOCK DIAGRAM

1.3 MANUAL ORGANIZATION

This manual is organized as follows:

- Section 1: General information and specifications of the TM 990/303A
- Section 2: How to install the TM 990/303A including required peripheral equipment, cabling, disk drive requirements, system configuration
- Section 3: Bringing up the system, demonstration software, bootload at powerup, data formats and disk drive parameters
- Section 4: Theory of operation including circuit descriptions, timing diagrams
- Appendices containing auxiliary data including logic diagrams, list of materials, cable pinouts, disk format tables, etc.

1.4 TYPICAL SYSTEM CONFIGURATION

In Figure 1-3, the TM 990/303A board is included in a card cage with a TM 990/101MA microcomputer and a TM 990/201 memory expansion board. The system terminal connects to P2 of the microcomputer board. Two standard-sized floppy disk drives are connected to the floppy controller by a TM 990/527 cable; if a full four-drive standard-size capability is desired, the user can provide his own cable using the data in Appendix F. (Up to three model 400 mini-sized floppy drives can be connected in a system using model 400 drives; these can be connected using the TM 990/535 cable.)

Note that the disk drives are "daisy-chained"; that is, they are connected by a single cable having connectors for each disk unit.

In a typical system, the bootstrap load feature of the TM 990/303A would be used to load into host RAM the routines to initialize the system. An initial load routine would be used to bring in other system tasks from the diskette to host RAM. These routines could include a file manager, device service routines to drive peripherals, as well as other system software requirements. The disk controller could then cause execution of a task in host memory that would start dedicated system functions.

CAUTION

Before applying power to the system carefully follow the installation procedures specified in Section 2 of this manual. When using a TM 990/5xx card cage, the etch on the backplane between lines 95 and 96 must be open (cut) in each slot that the TM 990/303A board is installed. Follow the procedure shown in section 2-6 and Figure 2-2. This applies to <u>all</u> TM 990/303A boards installed (<u>not</u> just to two or more boards).

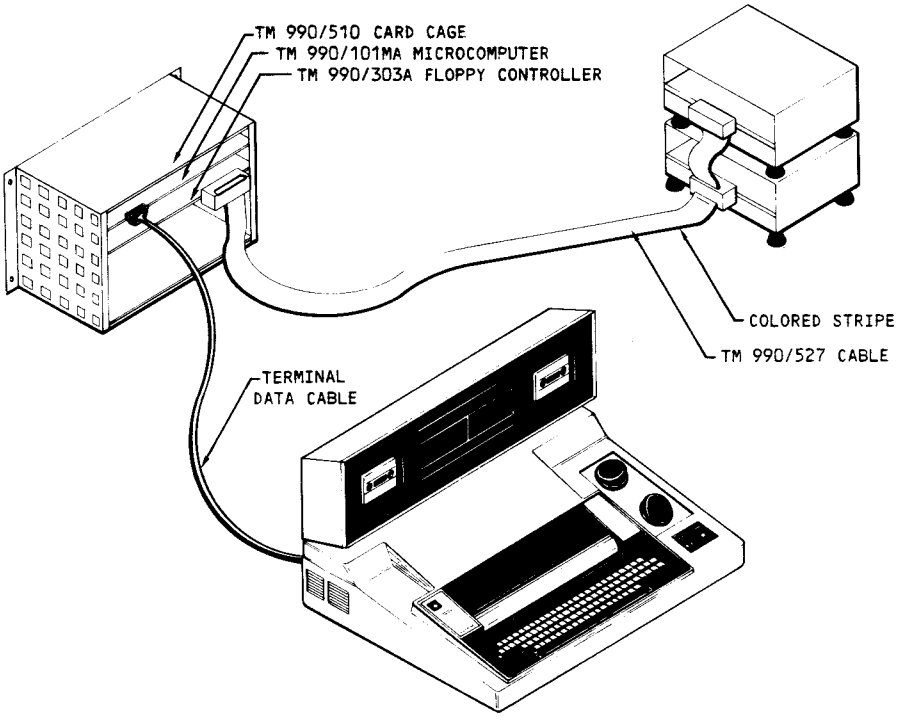


FIGURE 1-3. TYPICAL SYSTEM CONFIGURATION USING TWO MODEL 800 DISK DRIVES

1.5 POWER

1.5.1 TM 990/303A Power Requirements

The following are dc power requirements for the TM 990/303A:

	+5 Vdc		+12 Vdc		-12 Vdc		(voltage tol-	Unit
	Typ	Max	Typ	Max	Typ	Max	erance ±3%)	
TM 990/303A	2.1	3.0	0.1	0.2	0.04	0.2		A

During a powerup or a powerdown of either the disk controller or the disk drive (or both), data previously recorded on the diskette will not be destroyed due to controller action. Any operation in progress during a power sequence will not be completed.

If power is being supplied from separate power supplies, the system requires that -12V be turned on first and be turned off last. There is no required sequence in turning on the remaining voltages. This does not apply if the system uses only one power supply.

1.5.2 Disk Drive DC Power

	+5 Vdc		-5 Vdc		+12 Vdc		+24 Vdc		
	Typ	Max	Typ	Max	Typ	Max	Typ	Max	Unit
SA 400	0.5	0.7			0.9	1.8			A
SA 800	0.8	1.0	0.05	0.07			1.3	1.7	A
CDC 9404B	0.7						1.3		A
Qume DT-8									
- 1 drive	0.9	1.3					0.7	1.0	A
- 2 drives	1.6	2.2					0.8	1.2	A
- 3 drives	2.3	3.1					0.9	1.4	A
- 4 drives	3.0	4.0					1.0	1.6	A

1.6 ENVIRONMENT

Ambient Temperature

Operating: 0 to 50 degrees C (32 to 122 degrees F) at sea level
Storage: -40 to +100 degrees C (-40 to +212 degrees F)

Shock:

Shipping: 15 g applied to shipping container

Ambient Humidity:

Operating: 5 to 85 percent relative humidity without condensation
Storage: 5 to 95 percent without condensation

1.7 APPLICABLE DOCUMENTS

- TM 990/100MA Microcomputer User's Guide
- TM 990/101MA Microcomputer User's Guide
- TMS 9900 Microprocessor Data Manual
- TMS 9901 Programmable Systems Interface Data Manual
- TMS 9902 Asynchronous Communication Controller Data Manual
- TM 990/201/206 Expansion Memory Boards
- TM 990/203 Dynamic RAM Memory Expansion Module
- TM 990/527 Cable for Standard Disk Drives User's Guide
- TM 990/535 Cable for Mini Disk Drives User's Guide

SECTION 2

INSTALLATION AND OPERATION

2.1 GENERAL

This section covers the installation of the TM 990/303A Floppy Disk Controller and some demonstration software to illustrate controller operation.

CAUTION

1. Before applying power to your TM 990/303A board, properly set plugs and connectors as described in the following:
   - Jumper plugs at TM 990/303A board (section 2.4, Figure 2-1, Table 2-1)
   - Jumper plugs at Shugart 800 drive (section 2.5, Figure 2-2, Table 2-2)
   - Jumper plugs at Shugart 400 drive (section 2.5, Table 2-3)
   - Jumper plugs at CDC 9404B (section 2.5, Figure 2-3, Table 2-4)
   - Jumper plugs at Qume DataTrak 8 (section 2.5, Figure 2-4, Table 2-5)
   - Cable attachment to standard drive (section 2.7, Figures 2-7, 2-8), or
   - Cable attachment to mini drive (section 2.7, Figures 2-9, 2-10)
   - Cut backplane etch where controller board installed (see 2, below)

2. When using a TM 990/510 or /520 card cage, the etch on the backplane between lines 95 and 96 must be open (cut) in each slot that the TM 990/303A board is installed. Follow the procedure shown in section 2-6 and Figure 2-5. This applies to <u>all</u> TM 990/303A boards installed.

   Verify the system configuration using the check list in section 2.8 before applying power. Improper settings may harm equipment.

2.2 UNPACKING

Find the following:
- TM 990/303A Floppy Disk Controller Board
- TM 990/303A Floppy Disk Drive User's Guide
- Warranty Card

Remove the TM 990/303A module from its protective packing. Report any discrepancies to your supplier.

2.3 REQUIRED EQUIPMENT

This disk controller board is recommended for use only with the following disk drives:
- Shugart SA 800 (standard)
- Shugart SA 400 (mini)
- CDC 94904B (standard)
- Qume DataTrak 8 (standard)

A power supply must be supplied to meet the requirements of:
- Disk drive(s)
- TM 990/303A board as specified in section 1.5.
- TM 990/101MA, TM 990/100MA, or TM 990/100M microcomputer board.
- If installed, an expansion memory board (necessary for TM 990/100M or TM 990/100MA system) or other auxiliary board.

A TM 990/510 (four slots), TM 990/520 (eight slots), or equivalent card cage should be used to provide signal and power bussing to the microcomputer, disk controller, and (if used) expansion memory board. The microcomputer should be placed in the top of the cage (slot 1) with other boards below it.

A terminal (and proper terminal cabling) is required such as the Texas Instruments 733 or 743 for user interaction.

A connecting cable such as the TM 990/527 (two standard diskette drives) or TM 990/535 (three mini drives) is required between the disk controller and the floppy disk drive(s). Appendix F lists disk drive pinouts. An expansion RAM memory board will provide additional storage and work space for the host system. Suggested memory boards include the TM 990/203 dynamic memory board or the TM 990/206 static RAM board. Because memory on the TM 990/100M and the TM 990/100MA boards is buffered, it cannot be used for DMA.

2.4 JUMPERS ON TM 990/303A BOARD

Controller jumper settings are listed in Table 2-1; controller jumper locations are shown in Figure 2-1. The "Position" column contains the word or symbol as stenciled on the board that corresponds to the particular jumper setting desired. The term "Out" means the jumper is not installed; "In" means the jumper is installed. Note that jumper J9 (upper left of Figure 2-1) is not used. This jumper is reserved for future use in the designation of disk size. Jumper J8 specifies default disk format (IBM single density or TI double), but this can be changed by a command (see note below).

NOTE

> The Define Drive command (command 10_{16} in Table 3-2) must specify the same standard or mini size as jumnpered at pins J10 and J11. Failure to match the software designation (via the Define Drive command) and the hardware settings at J10 and J11 will result in incorrect performance. This command does not have to conform to jumper J8 format setting which prescribes the default value only at a powerup or bootload; this default can be changed by the Define Drive command.

2.5 JUMPERS ON DISK DRIVE

Follow the manufacturer's instructions for setting jumpers on the respective disk drive printed circuit board. Suggested disk drive jumper settings are listed in Table 2-2 (Figure 2-2) for the Shugart model 800, Table 2-3 for the Shugart model 400, Table 2-4 (Figure 2-3) for the CDC model 9404B, and Table 2-5 (Figure 2-4) for the Qume DT-8.

On the model 800, the user must select a unique drive number (DS1 to DS4) for each drive to correspond to the disk drive ID (00 to 11) in word 2 of the Command List (section 3.4.3.4). ID 00 corresponds to DS1, ID 01 to DS2, etc. If the drive is the last (or only) drive on the cable, install terminator jumpers on T1, T2, T3, T4, T5, and T6.

If a single model 400 drive is used, it is shipped from the factory properly jumpered. If two or three drives are used, open the MX shunts, and leave closed the applicable drive-select shunt (DS1-DS3) while the other two drive select shunts are cut (opened). Only the last disk drive on the cable will have the termination resistor pack installed; remove it from the others.

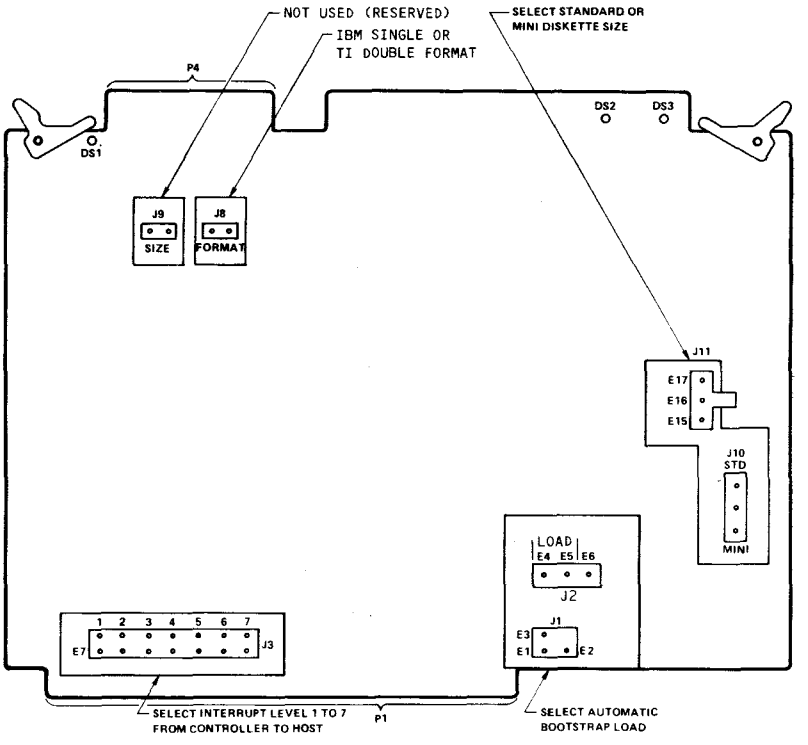


FIGURE 2-1. JUMPER LOCATIONS ON TM 990/303A BOARD

TABLE 2-1. TM 990/303A JUMPER SETTINGS

Function	Jumper	Pin[1]	Position	As Shipped?
Select Bootstrap Load				
Automatic Bootstrap Load	J1	E1-E2	LOAD	
	J2	E5-E4	LOAD	
No Automatic Bootstrap Load	J1	E1-E3	--	Yes
	J2	E5-E6	--	Yes
Interrupt Level to Host	J3			
Interrupt Level 1		E55-E7	1	
Interrupt Level 2		E56-E8	2	Yes
Interrupt Level 3		E57-E9	3	
Interrupt Level 4		E58-E10	4	
Interrupt Level 5		E59-E11	5	
Interrupt Level 6		E60-E53	6	
Interrupt Level 7		E61-E54	7	
Select Default Format for				
Powerup Reset	J8			
IBM Single Density, Single Sided, 8 in.		In		Yes
TI Double Density, Single Sided, 8 in.		Out		
Select Disk Size[2]				
Standard Size	J10	E19-E18	STD	Yes
	J11	E16-E17	STD	Yes
Mini Size	J10	E19-E20	MINI	
	J11	E16-E15	MINI	

Note: 1. Out = jumper not installed; In = jumper installed.
2. Jumpers J4, J5, J6, and J7 not used with TMS 2716 as onboard memory
3. Jumper J9 is not used; provided for future use.

TABLE 2-2. SUGGESTED SHUGART SA 800 DISK DRIVE JUMPER SETTINGS

Jumper Name	Board Position	Termination Board Jumper In or Out	Intermediate Board Jumper In or Out
A	A	In	In
B	A	In	In
C (P18)	B	In	In
D	B	Out	Out
DC	B	In	In
DDS	C	Out	Out
DS	D	In	In
DS1	E	See Note 1	See Note 1
DS2	E	See Note 1	See Note 1
DS3	E	See Note 1	See Note 1
DS4	E	See Note 1	See Note 1
HL	D	Out	Out
T1	C	In	Out
T2	C	In	Out
T3	E	In	Out
T4	E	In	Out
T5	E	In	Out
T6	E	In	Out
X	F	Out	Out
Y	G	Out	Out
Z	G	In	In
800	H	In	In
801	H	Out	Out

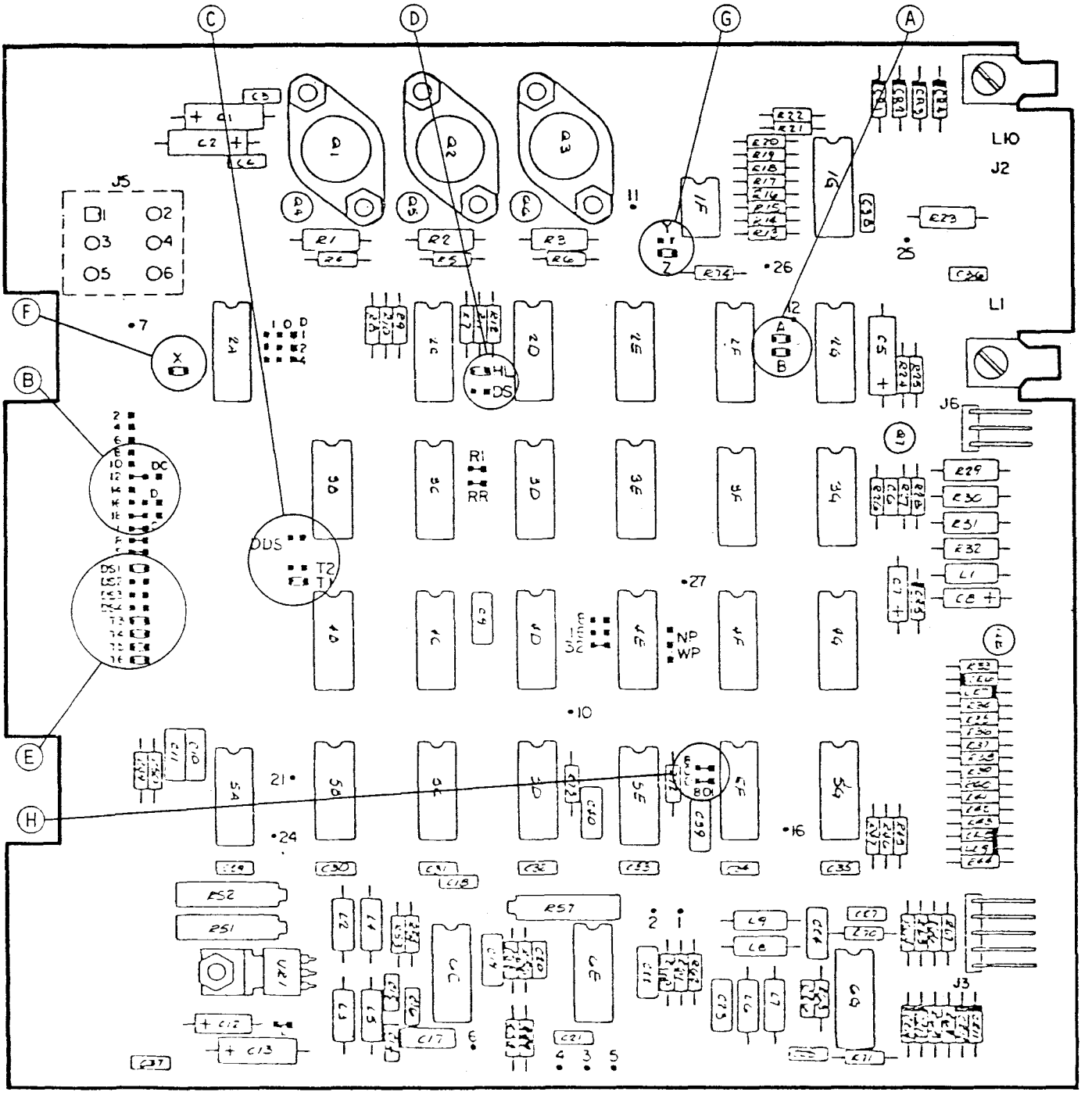
Note 1. Only one of the DS1 to DS4 (Drive Select 1 to 4) is jumpered to select a disk drive; this number (1 to 4) corresponds to the disk drive select number specified in the two LSBs of Command Word 2.

TABLE 2-3. SUGGESTED SHUGART SA 400 DISK DRIVE SIGNAL PLATFORM SETTINGS

Name	Termination Board Connection	Intermediate Board Connection
MH	Open	Open
MX	Open	Open
DS3	See Note 1	See Note 1
DS2	See Note 1	See Note 1
DS1	See Note 1	See Note 1
HL	Closed	Closed
Resistor Pack at 1E	Installed	Removed

Notes 1. Only one of the DS1 to DS3 (Drive Select 1 to 3) is shorted to select a disk drive; this number (1 to 3) corresponds to the disk drive select number specified in the two LSBs of Command Word 2.

2. The program shunt can be replaced with a DIP switch (AMP 435626-4) for ease in programming.



Jumper Plug Installed as Shipped

Test Point

FIGURE 2-2. JUMPER LOCATIONS ON THE SHUGART SA 800 DISK DRIVE

TABLE 2-4. SUGGESTED CDC 9404B DISK DRIVE JUMPER SETTINGS

Jumper	Board Position	Termination Board Jumper In or Out	Intermediate Board Jumper In or Out
W1/W5	A	See Note 1	See Note 1
W2/W5	A	See Note 1	See Note 1
W3/W5	A	See Note 1	See Note 1
W4/W5	A	See Note 1	See Note 1
Terminator Pack Beckman R220/330	B	In	Out

NOTE 1

Drive designators W1 to W4 correspond to drive select numbers 1 to 4. One of these is jumpered to W5 to designate the disk drive number which corresponds to the disk drive select number in the two LSBs of Command Word 2. As shipped from the factory, W1 is jumpered to W5 to designate the drive as drive 1, selected by a 00_2 in bits 14 and 15 of Command Word 2. For example:

Disk Drive Select Number	Install Jumper
1	W1/W5
2	W2/W5
3	W3/W5
4	W4/W5

To select drive 2, 3, or 4, the user must first remove the soldered jumper installed at the factory between W1and W5 (to select drive 1). It is recommended that the user substitute a four-switch standard DIP package (AMP 435166-2 or 435626-1, 7000 series) between W5 and W1 to W4. This allows the user to easily designate the drive number by setting one of the four switches to ON.

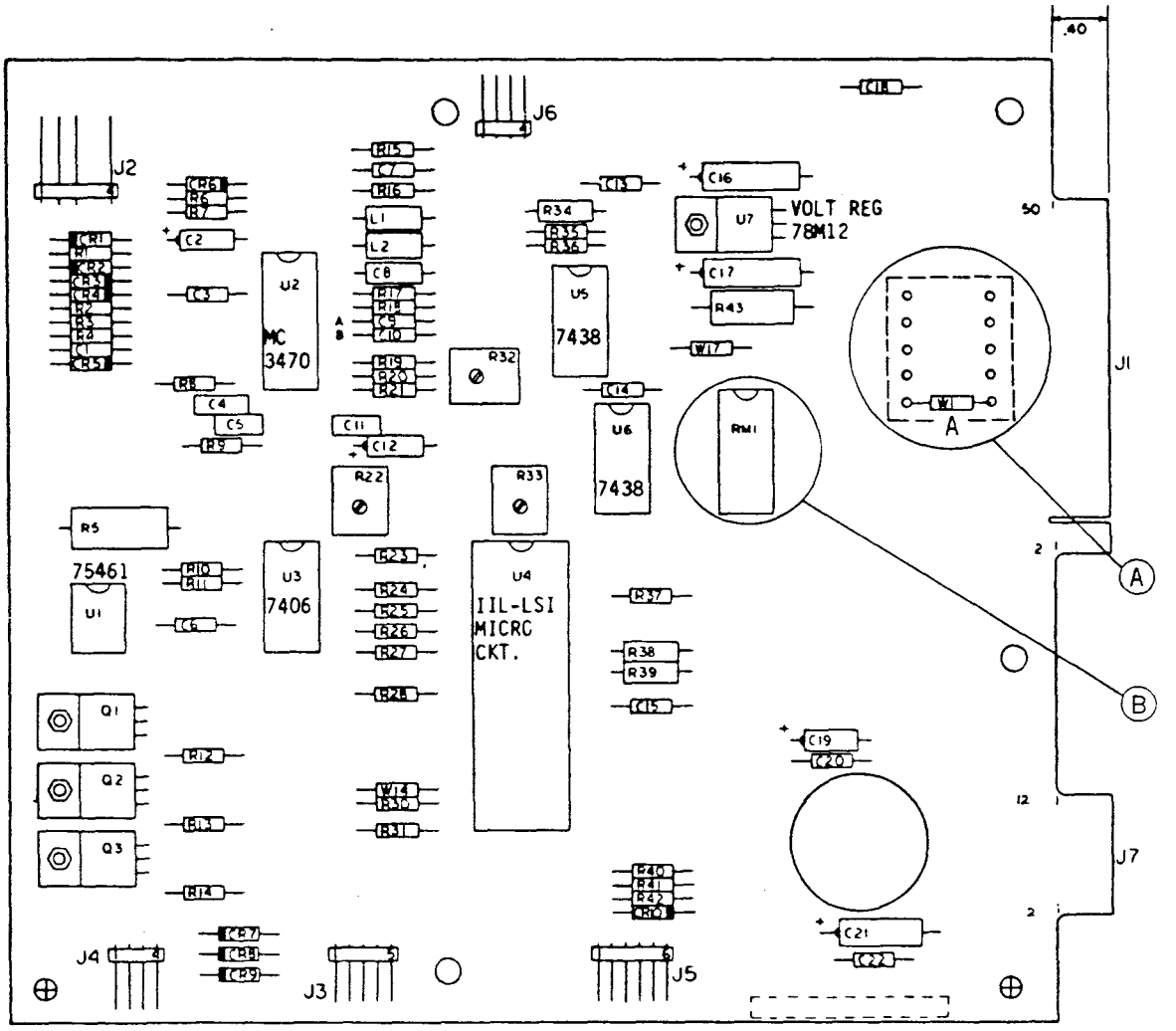


FIGURE 2-3. JUMPER LOCATIONS ON THE CDC 9404B DISK DRIVE

TABLE 2-5. SUGGESTED QUME DT-8 DISK DRIVE JUMPER SETTINGS

Jumper	Description	Board Position	Termination Board In or Out	Intermediate Board In or Out
DS1	Drive select 1	D	See Note 1	See Note 1
DS2	Drive select 2	D	See Note 1	See Note 1
DS3	Drive select 3	D	See Note 1	See Note 1
DS4	Drive select 4	D	See Note 1	See Note 1
RR	Radial ready	J	In	In
R1	Radial index and sector	J	In	In
R	Shunt option for Ready Output	H	In	In
2S	Two-sided status output	B	In	In
I	Index output	H	In	In
DC	Disk change option	B	In	In
HL	Stepper power from Head Load	H	Out	Out
DS	Stepper power from Drive Select	G	Out	Out
WP	Inhibit write when write protected	I	In	In
NP	Allow write when write protected	I	Out	Out
DL	Door lock latch option	G	Out	Out
B	Radial head load	H	In	In
X	Daisy chain head load	H	Out	Out
A	Radial Select	H	In	In
Z	In-use from Drive Select	H	Out	Out
Y	In-use from Head Load	G	In	In
S1	Side select option using direction	C	Out	Out
S2	Standard side select input	C	In	In
S3	Side select option using Drive Select	C	Out	Out
C	Head load	B	In	In
D	In use	B	Out	Out
W	See note 2	C	Out	Out
SS	See note 2		In	In
D1	Binary method of drive select	E	Out	Out
D2	Binary method of drive select	E	Out	Out
D4	Binary method of drive select	E	Out	Out
DDS	Binary method of drive select	F	Out	Out
B1-B4	Two double side drive select	D	Out	Out
1TM	Termination resistor packs	A	In	Out
2TM	Termination resistor packs	A	In	Out

NOTES

1. Only one of DS1 to DS4 is installed to designate the drive number. This number (DS1 to DS4) corresponds to the disk drive select number specified in the LSB's of Command Word 2 (e.g., to designate a disk drive as number 1, jumper DS1).

2. Traces W and SS are not on some Qume DT-8 models that have been modified for use with TI options. These board require the following modifications before being used with the TM 990/303A:

   a. Reconnect trace C which has been opened.
   b. Remove a wire from 2G pin 8 to 1D pin 5 (no head load latch)
   c. Remove a wire from J1-18 pad (c) to 1D pin 3 (no head load in use)
   d. Remove a wire from 3F pin 10 to 3F pin 7.

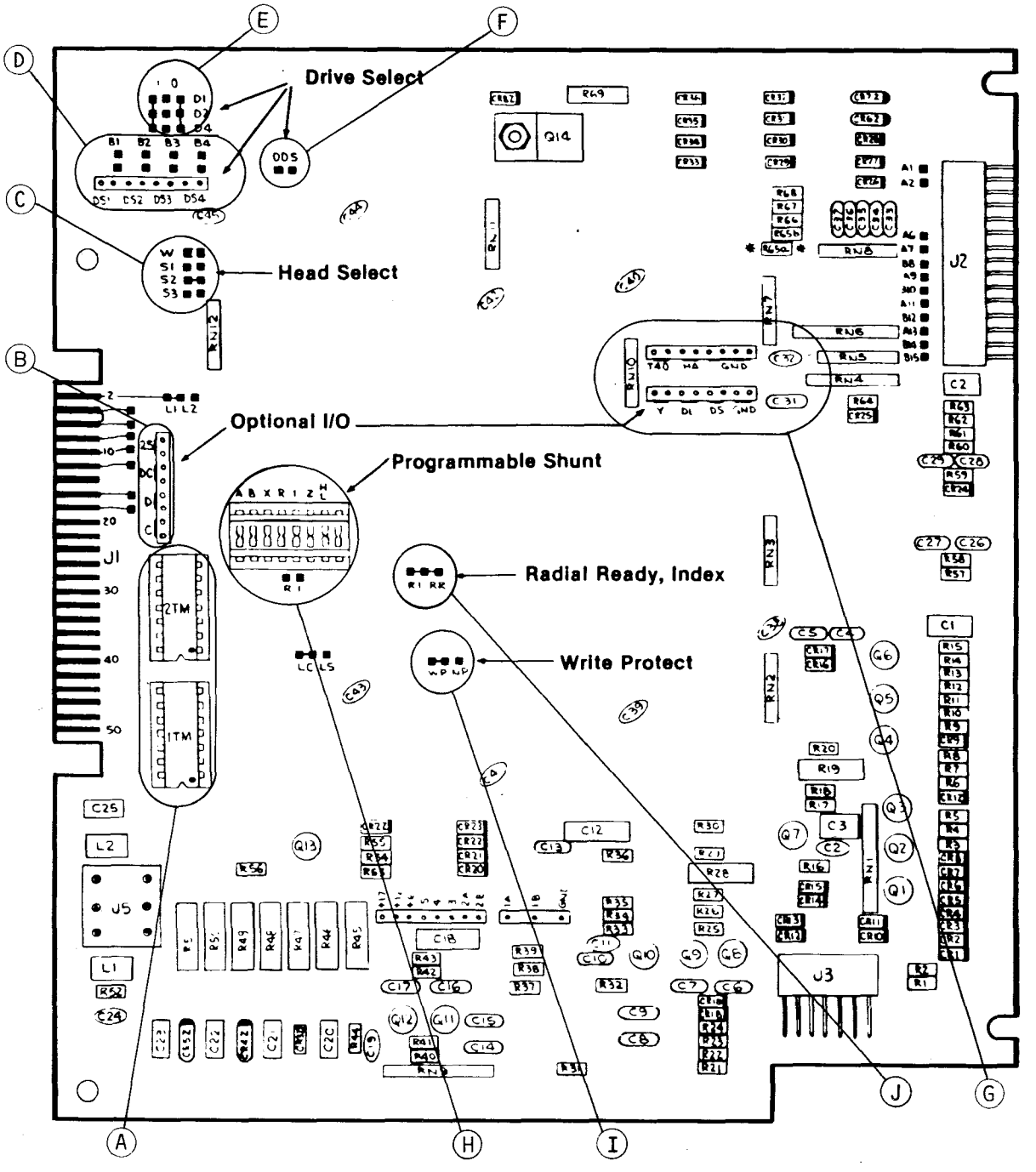


FIGURE 2-4. JUMPER LOCATIONS ON THE QUME DT-8 DISK DRIVE

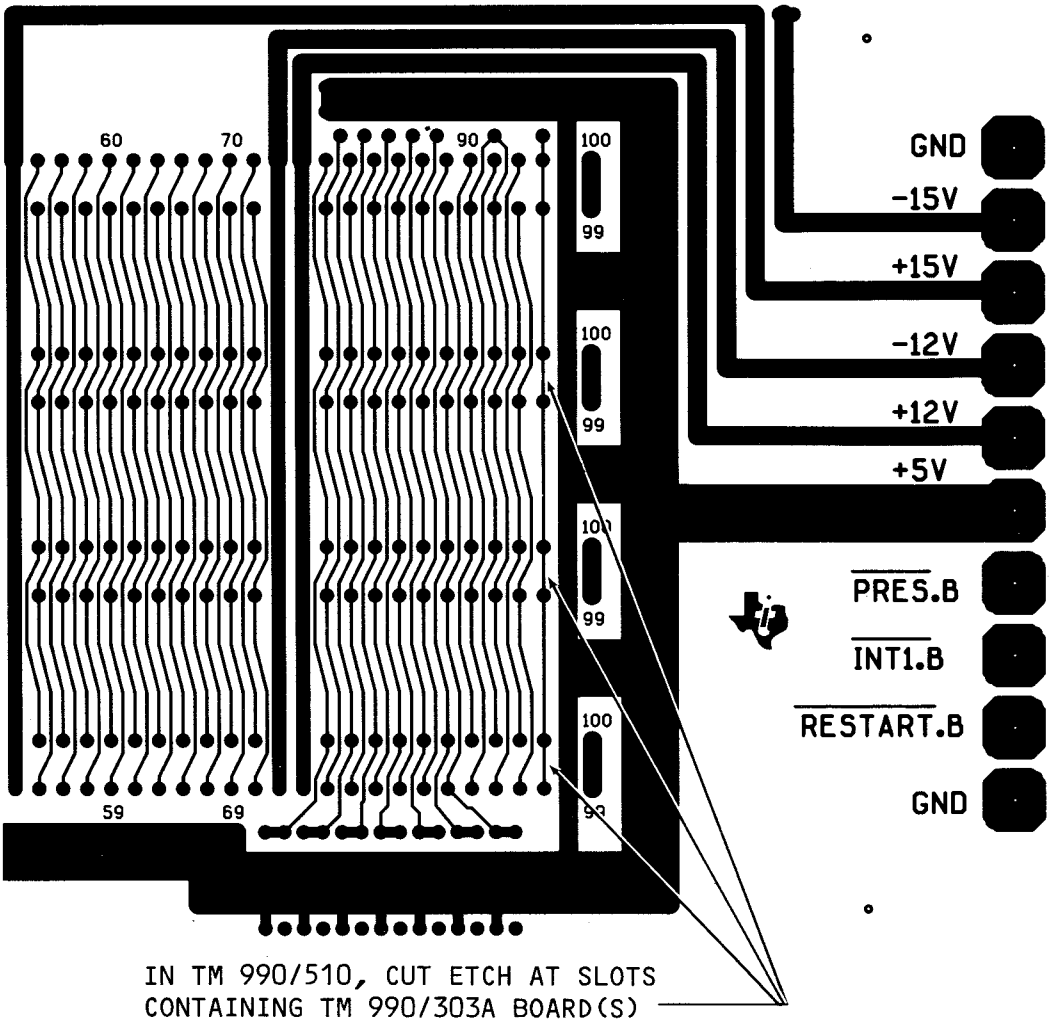


FIGURE 2-5. LOCATION OF SOLDER BRIDGE BETWEEN PINS 96 AND 95 OF MOTHERBOARD

2.6 BOARD INSTALLATION

Turn power off before installation of cards into the card cage. Install the microcomputer board at the top of the cage with other boards beneath. Where the TM 990/303A boards are installed, cut the etch on the card cage backplane between lines 95 and 96 as shown in Figure 2-5. This is the same as for multicontroller systems as explained in section 2.10. Do not apply power until cards are properly installed and cables connected as specified in section 2.7. With power applied, the controller will execute a self-test with LEDs DS2 and DS3 on and DS1 off; when the test is successfully completed, these two LEDs will extinguish and DS3 will remain illuminated. Board status, as shown in the LEDs, is explained in section 2.9.

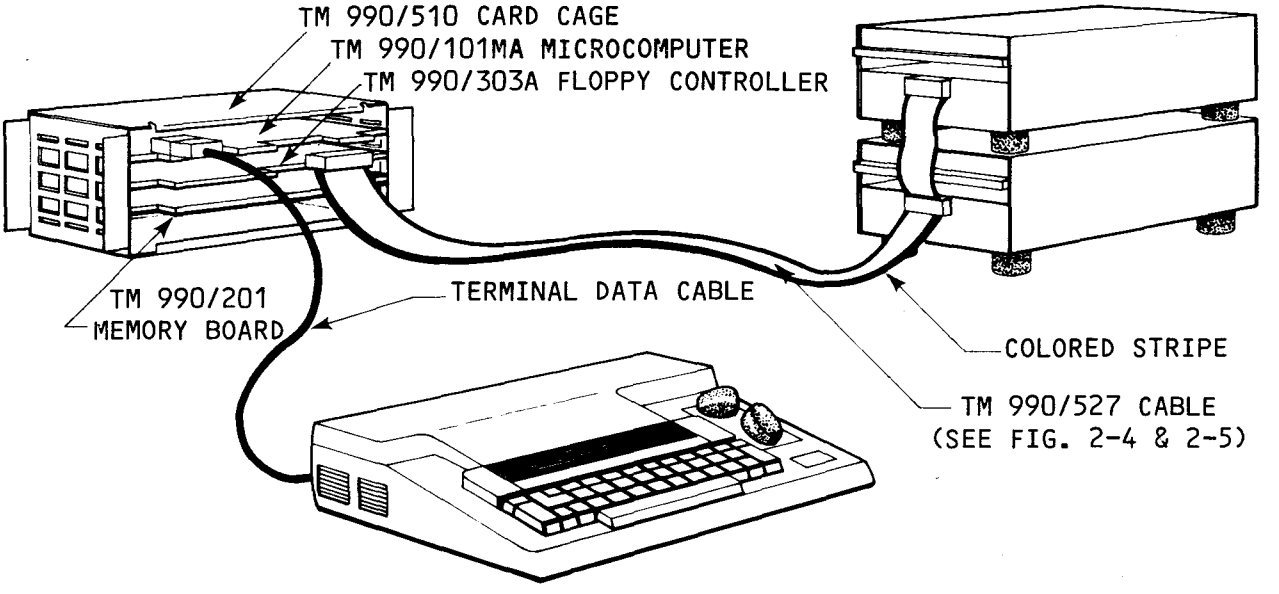


FIGURE 2-6. SYSTEM INTERCONNECTIONS USING TM 990/527 CABLE (STANDARD SIZE)

2.7 CABLING

Figure 2-6 shows a typical system configuration using a standard size drive and TM 990/527 cable. Detail connections using the this cable are shown in Figures 2-7 and 2-8.

- Connect the data cable from the terminal to connector P2 of the microcomputer board.

- The TM 990/527 cable has three connectors on it. The end with the two connectors closest together contains the connectors to the two disk drives. The single connector at the other end goes to the controller board, and is attached with the colored stripe to the left of connector P4 as shown in Figure 2-7. As shown in the figure, pin 2 and the colored stripe are at the left of the connector when properly installed. The colored stripe encloses the wire to pin 1.

- Connect one or both of the disk drive cable connectors to the back of the disk drive(s). Consult the manufacturer's installation instruction for proper orientation of pin 1 on the disk drive connector. Pin 1 on the TM 990/527 cable is designated by a diamond engraved into the connector close to the side near the colored stripe. See Figure 2-8 for orientation. If pin 1 is on top of the edge connector at the disk drive, the diamond on the connector must also be on top, oriented with that pin.

Figures 2-9 and 2-10 show similar cabling for model 400 mini drives using the TM 990/535 cable and its interface card. Note that the card acts as the interface between the 50-pin connector P4 on the TM 990/303A board and the 34-wire cable.

NOTE

If you want to make you own cable, be aware that the connector plugs of various vendors, including TI, do not necessarily use the numbering schemes on the board edge connector. ALWAYS refer to the board edge when wiring a connector.

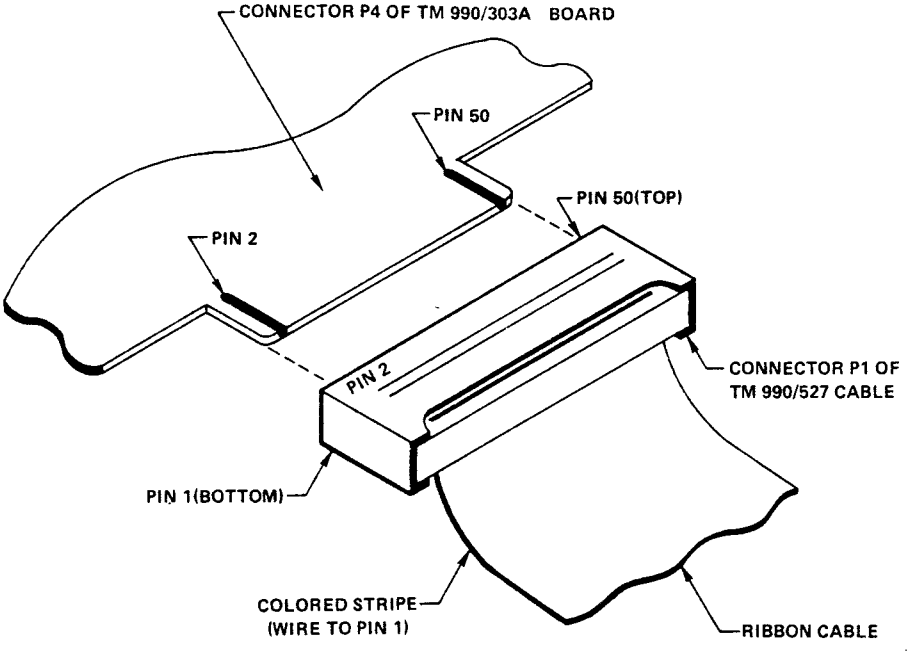


FIGURE 2-7. CONNECTING TM 990/527 DISK DRIVE CABLE TO P4 OF TM 990/303A BOARD

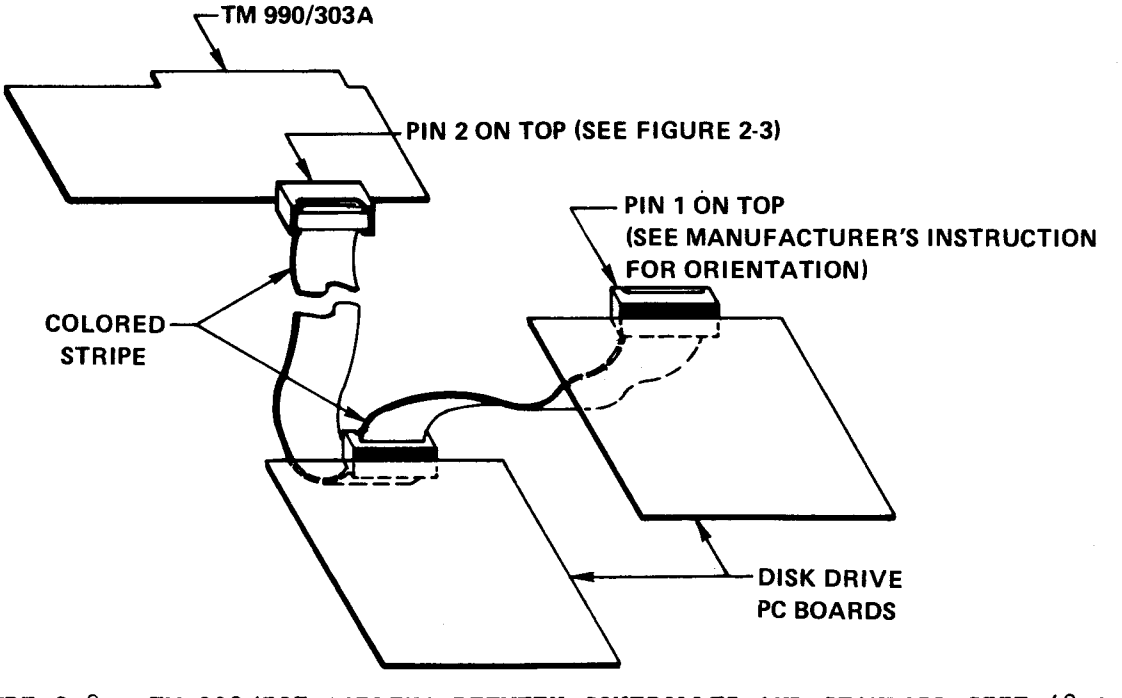


FIGURE 2-8. TM 990/527 CABLING BETWEEN CONTROLLER AND STANDARD SIZE (8 in.) DISK DRIVES

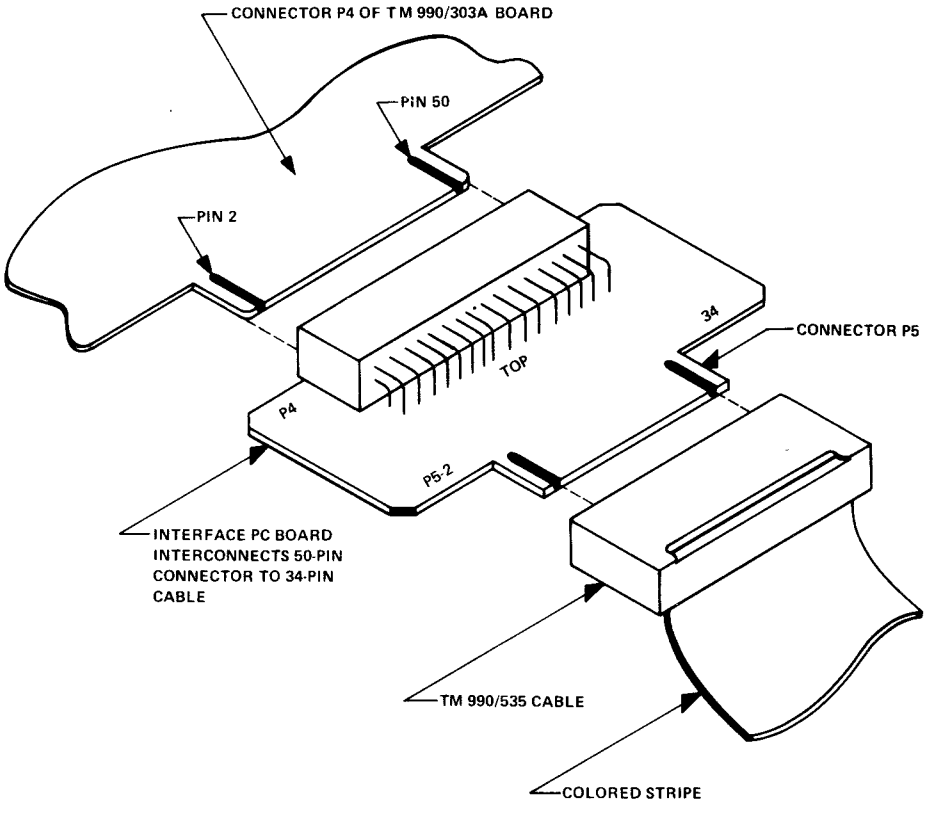


FIGURE 2-9. CONNECTING TM 990/535 DISK DRIVE CABLE TO P4 OF TM 990/303A BOARD

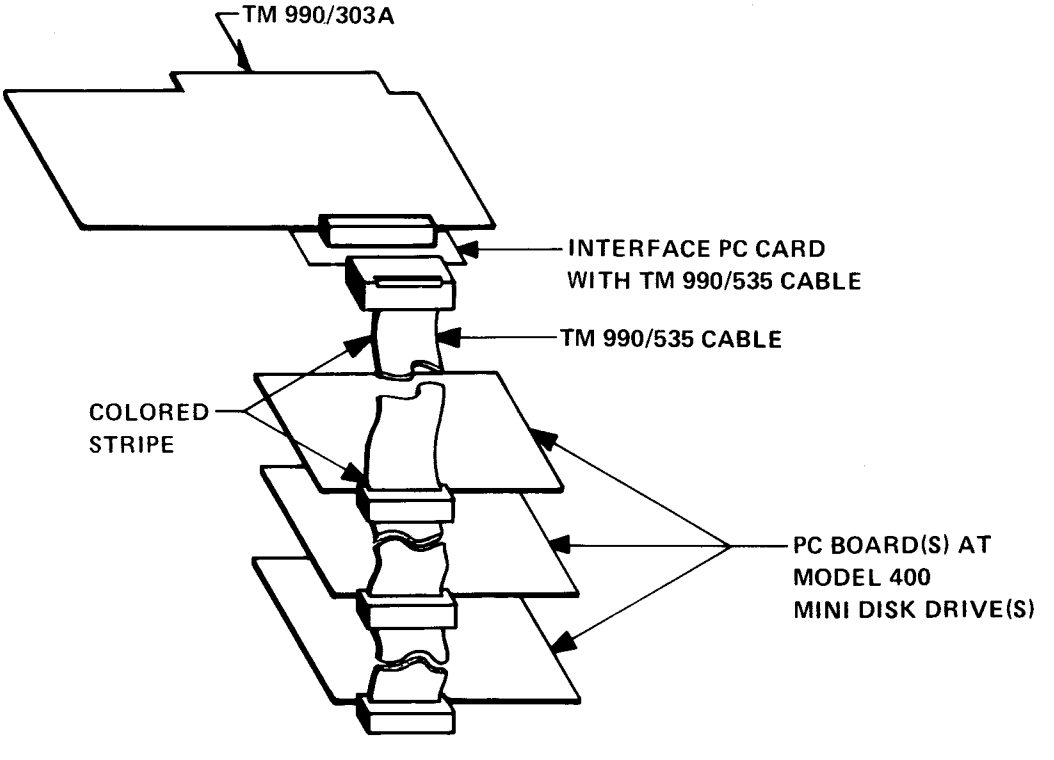


FIGURE 2-10. TM 990/535 CABLING BETWEEN CONTROLLER AND MODEL 400 DISK DRIVES

2.8 SYSTEM CHECK AND POWER APPLICATION

Do not apply power until the PC cards (TM 990/303A as well as the disk drive) are properly jumpered and installed and cables are connected as specified in this section. Before applying power, use the following check list to verify proper installation according to applicable paragraphs, figures, and tables:

☐ Jumper plugs at TM 990/303A board (section 2.4, Figure 2-1, Table 2-1)

☐ Jumper plugs at disk drive (section 2.5, and one of
Table 2-2, Figure 2-2 for Shugart 800, or
Table 2-3 for Shugart 400, or
Table 2-4, Figure 2-3 for CDC 9404B, or
Table 2-5, Figure 2-4 for Qume DT-8 disk drive).

☐ Backplane etch is cut at controller board slot, pins 95/96 (section 2.6, Figure 2-5).

☐ Cable attachment to standard drive (section 2.7, Figures 2-7, 2-8), <u>or</u> cable attachment to mini drive (section 2.7, Figures 2-9, 2-10)

Cable attachments should be as shown in the applicable figure. When system installation has been checked and verified, apply power. If power is being supplied from separate power supplies, the system requires that -12V be turned on first and be turned off last. There is no required sequence in turning on the remaining voltages. This does <u>not</u> apply if the system uses only one power supply.

With power applied, the controller will execute a self-test. LEDs DS2 and DS3 will go on; when test is complete, these will go off and DS1 will remain on indicating no error. LED interpretation is shown in section 2.9.

2.9 ONBOARD LED ERROR CHECK

Placement of the three LEDs on the TM 990/303A is shown in Figure 1-1 (DS3 to DS1, left to right as seen from the front of the card cage). These lights reflect board status as shown below.

DS3	DS2	DS1	Condition/Examples
N/A	N/A	On	No error condition
Off	Off	Off	No power
Off	On	Off	Self-test error (clock, memory)
On	Off	Off	Disk drive error (write protect, not ready)
On	On	Off	Controller error (CRC, overrun, I/O)

2.10 TWO OR MORE TM 990/303A BOARDS IN A SYSTEM

More than one TM 990/303A board can be installed in a card cage and share the bus on the motherboard. However, several items must be considered:

- Access to the bus must be arbitrated
- Each board must have a unique CRU address space for communicating with the host microcomputer

Access through the common bus will be arbitrated by the GRANTIN.B- signal at P1-96 and the GRANTOUT.B- signal at P1-95 as shown on page 4 of the schematics in Appendix D. Through these two signals, the bus is arbitrated so that the board in the highest position in the chassis (closest to the microcomputer board) has priority for the bus over the board beneath it in the chassis.

The solder pattern on the motherboard of the card cage has to be opened between the pins of 96 (top) and 95 (bottom) for the slots containing the TM 990/303A boards. The location of these lines is shown in Figure 2-5. This allows the upper board to obtain bus control by presenting his GRANTOUT.B signal (via pin 95 on the lower side of the TM 990/303A board) to the lower board's GRANTIN.B- line (pin 96 on the upper side of the board). This lets the controller board with the higher priority (higher slot position) to suspend bus access by the lower-priority board.

Communication via the CRU is covered in detail in Section 3. As stated above, each board will require a unique CRU base address for communication with the host micromputer. This CRU address is selected by bits programmed into the 74S287 PROM at socket U13. This socket must contain a PROM programmed with a pattern so that each board has a unique CRU address. For example (as shown in Figure 3-3 of the next section), the TM 990/303A board is shipped so that it occupies the 32 bits of CRU address space between hardware base address 100_{16} and 120_{16}. The 74S287 PROM at U13 must be programmed so that its DO1 output has a low value on it when driven by address lines A3.B to A3.10. The address line value to enable a low value at DO1 on one TM 990/303A must be different from the address line value driving the other U13 PROMs of the other TM 990/303As. As shipped, the U13 PROM outputs a low at DO1 when address lines A3.B to A10.B are all zeroes except for A5.B (a one). Instructions to reprogram U13 are provided in Appendix E of this manual.

2.11 DEMONSTRATION PROGRAM

With the system properly connected and powered up, the program in Figure 2-11 can be entered into host memory using TIBUG. (This program cannot be run on the TM 990/100M/100MA microcomputers as DMA cannot be accomplished to the RAM on these microcomputer boards.) This program causes the disk controller to read the message "Congratulations it works" from host memory, write this message to disk, read the message back from disk, and write the message to the system terminal. This program assumes:

- The disk format is IBM single density and follows the format block as shown in Appendix A, page A-2 for that format. This format is called out in the define drive block of code at source lines 0051 to 0058; note that the assembled values for this block match the values shown on page A-2. If another format is used, the define format values for the particular format are presented in Appendix A; these values can be substituted into source lines 0051 to 0058.

- The system is jumpered for the disk size and format desired; the TM 990/303A board is shipped jumpered for IBM single density, standard size.

- TIBUG is in EPROM in lower memory and RAM is in upper memory on the microcomputer. The program is loaded beginning at $FC20_{16}$ and requires RAM between $FC00_{16}$ and $FDA2_{16}$, as assembled.

- The disk drive accessed is DS1 which should be the jumpered ID at the disk drive PC board.

CAUTION

Data will be written to and read from the second track, first sector on the diskette (take care that data present on the diskette used will not be destroyed).

The program can be entered line-by-line by using the memory (M) inspect/change command of TIBUG. The second column contains the memory addresses; the third column cotains the object codes to insert at those addresses.

If assembled, the program uses an AORG directive to load the program beginning at $FC20_{16}$ (with workspace pointer at $FC00_{16}$), and it uses an entry vector label on the END assembler directive. These allow the program to be loaded with the TIBUG L command without a bias specified, and then executed immediately with the TIBUG E command as shown below:

```
?L
FC20 DFNC
?E
CONGRATULATIONS. IT WORKS!

?
```

Program entry is at memory address $FC82_{16}$; this value must be in the program counter before re-executing the program. Note that the program checks the Operation Complete bit of the Command List to determine completion of the Command List. It also enters a timeout loop in case of failure to complete the command and set the Operation Complete bit. The use of a timeout routine is necessary as a regular programming practice to avoid hangups in case the command is not completed and the Operation Complete bit is not set.

```
0001                    IDT  'FLOPDEMO'
0003 FC20               AORG >FC20          PROGRAM LOAD ADDRESS
0004                *** *** *** *** *** *** *** *** *** *** *** *** *** *** *** ***
0005                ** THIS PROGRAM IS A DEMONSTRATION OF THE TM 990/303A FLOPPY
0006                ** DISK CONTROLLER WITH A TM 990/101MA BOARD. THIS PROGRAM
0007                ** ASSUMES THE FOLLOWING BUT CAN BE CHANGED TO USE OTHER
0008                ** FORMATS:
0009                ** 1. THE CONTROLLER IS CABLED TO ONE OF THE STANDARD
0010                **    SIZED DISK DRIVES SUITABLE FOR THIS CONTROLLER WITH
0011                **    THE DRIVE JUMPERED AS DS1.
0012                ** 2. NO INTERRUPTS TO THE HOST ARE GENERATED FOR COMMAND
0013                **    COMPLETION.
0014                ** 3. THIS PROGRAM CAN BE RE-EXECUTED CONTINUOUSLY WITH THE
0015                **    USER MAKING EXPERIMENTAL MODIFICATIONS AS DESIRED. IF
0016                **    OPERATION IS UNSUCCESSFUL -- CHECK JUMPERS ON CONTROLL
0017                **    AND ON DISK DRIVE TO VERIFY FORMAT COMPATIBILITY.
0018                ** 4. DISK FORMAT IS IBM STANDARD SIZE, AND THE CONTROLLER
0019                **    BOARD IS JUMPERED THUSLY AT J10, & J11.
0020                **    SINGLE DENSITY IS SPECIFIED IN THE DISK DRIVE FORMAT
0021                **    BLOCK BEGINNING AT MEMORY ADDRESS >FC34 (2D COLUMN).
0022                **    THE USER CAN CHANGE THE FORMAT PARAMETERS IN THIS
0023                **    BLOCK IF DESIRED (E.G., FOR MINI FORMAT), BUT JUMPER
0024                **    ACCORDINGLY.
0025                ** 5. IT IS SUGGESTED THAT A NEW DISKETTE BE USED FOR THIS
0026                **    DEMO PROGRAM. ONLY ONE TRACK (SECOND TRACK) WILL
0027                **    BE FORMATTED; THE FORMAT USED WILL BE IBM STANDARD.
0028                ** 6. A TM 990/101MA BOARD IS USED WITH TIBUG IN LOWER MEM-
0029                **    ORY, RAM IN UPPER MEMORY, OR A MEMORY BOARD WITH RAM
0030                **    IN UPPER MEMORY AND A TM 990/101MA BOARD.
0031                ** 7. A CHECK IS MADE FOR ERRORS IN COMMAND EXECUTION;
0032                **    HOWEVER, NO ERROR CORRECTION ROUTINES ARE PROVIDED.
0033                ** 8. DATA WILL BE WRITTEN TO THE FIRST SECTOR OF THE SECOND
0034                **    TRACK ON THE DISKETTE. TAKE CARE THAT DATA WILL NOT
0035                **    BE DESTROYED AT THIS ADDRESS.
0036                ** 10/25/79  J.J.WALSH
0037                ** 08/10/79  M.L.STRAIN
0038                *** *** *** *** *** *** *** *** *** *** *** *** *** *** *** ***
0039                ** *  *  *  *  *  *  *  *  *  *  *  *  *  *  *  *  *  *  *  *
0040                ** EQUATES, COMMAND LISTS, AND COMMAND LIST DATA
0041                ** *  *  *  *  *  *  *  *  *  *  *  *  *  *  *  *  *  *  *  *
0042                * EQUATE MNEMONICS
0043        0008    COMND  EQU  8            DISPLACEMENT ON CRU FOR COMMAND BIT
0044        000A    CUE    EQU  10           DISPLACEMENT ON CRU FOR CUE BIT
0045        000B    ACCEPT EQU  11           DISPLACEMENT ON CRU FOR BUSY BIT
0046        000C    BUSY   EQU  12           DISPLACEMENT ON CRU FOR ACCEPT BIT
0047                * ADDRESS BYTES FOR FIRST COMMAND LIST
0048 FC20   000F    CMLST1 DATA >000F        EXTENDED ADDR. ALL ONES FOR MEM BRD
0049 FC22   FC34           DATA CLIST1       BYTES 2 AND 3
0050                * DEFINE DISK FORMAT BLOCK FOR IBM SINGLE DENSITY
0051 FC24   0101    FORMAT DATA >0101        DISK SIZE, NO. OF SURFACES
0052 FC26   004D           DATA 77           NUMBER OF TRACKS
0053 FC28   03E8           DATA 1000         HEAD STEP TIME X 10 US
0054 FC2A   05DC           DATA 1500         STEP SETTLING TIME X 10 US
0055 FC2C   0DAC           DATA 3500         HEAD LOAD TIME X 10 US
0056 FC2E   03E8           DATA 1000         HEAD UNLOAD TIMEOUT (MS)
0057 FC30   0000           DATA 0            DENSITY, SYNC TYPE, INTERLACE FACTR
0058 FC32   6880           DATA >6880        SEC/TR
```

FIGURE 2-11. DEMO PROGRAM TO READ TO/WRITE FROM DISK (SHEET 1 OF 6)

```
0060                * FIRST COMMAND LIST -- DEFINE DISK FORMAT
0061 FC34 0000      CLIST1 DATA 0          WORD 0, CLEAR FLAGS
0062 FC36 0000             DATA 0          WORD 1, CLEAR FLAGS
0063 FC38 1000             DATA >1000      WORD 2, DEFINE DISK FORMAT COMMAND
0064 FC3A 0000             DATA 0          WORD 3, NOT NEEDED
0065 FC3C 0000             DATA 0          WORD 4, NOT NEEDED
0066 FC3E 0000             DATA 0          WORD 5, NOT NEEDED
0067 FC40 000F             DATA >000F      WORD 6, ADDRESS OF
0068 FC42 FC24             DATA FORMAT     WORD 7, FORMAT PARAMETERS ADDR
0069 FC44 800F             DATA >800F      WORD 8, CHAIN TO NEXT CMD LIST
0070 FC46 FC48             DATA CLIST2     WORD 9, NEXT CMD LIST ADDRESS
0071                * SECOND COMMAND LIST -- FORMAT SECOND TRACK OF SIDE 0
0072 FC48 0000      CLIST2 DATA 0          WORD 0, CLEAR FLAGS
0073 FC4A 0000             DATA 0          WORD 1, CLEAR FLAGS
0074 FC4C 0900             DATA >0900      WORD 2, FORMAT TRACK COMMAND
0075 FC4E 8001             DATA >8001      WORD 3, FORMAT SECOND TRACK (01)
0076 FC50 0001             DATA >0001      WORD 4, SURFACE/SECTOR ADDRESS
0077 FC52 0000             DATA 0          WORD 5, NOT NEEDED
0078 FC54 000F             DATA >000F      WORD 6, FIRST BYTE OF THE --
0079 FC56 FD7E             DATA PATERN     WORD 7, ADDRESS OF FORMAT PATTERN
0080 FC58 800F             DATA >800F      WORD 8, CHAIN TO NEXT CMD ADDRESS
0081 FC5A FC5C             DATA CLIST3     WORD 9, NEXT CMD LIST ADDRESS
0082                * THIRD COMMAND LIST -- WRITE TO DISKETTE
0083                * PHYSICAL STORAGE MODE USED IN WORDS 3 AND 4
0084 FC5C 0000      CLIST3 DATA 0          WORD 0, CLEAR FLAGS
0085 FC5E 0000             DATA 0          WORD 1, CLEAR FLAGS
0086 FC60 0400             DATA >0400      WORD 2, WRITE TO DISK DS1 COMMAND
0087 FC62 8001             DATA >8001      WORD 3, (& 4) TRACK ADDRESS
0088 FC64 0001             DATA >0001      WORD 4, SURFACE/SECTOR ADDRESS
0089 FC66 0080             DATA >80        WORD 5, WRITE >80 BYTES (1 SECTOR)
0090 FC68 000F             DATA >000F      WORD 6, ADDRESS OF
0091 FC6A FD5C             DATA MESG       WORD 7, ADDRESS OF MESSAGE IN HOST
0092 FC6C 800F             DATA >800F      WORD 8, CHAIN TO NEXT CMD LIST
0093 FC6E FC70             DATA CLIST4     WORD 9, NEXT CMD LIST ADDRESS
0094                * FOURTH COMMAND LIST -- READ FROM DISKETTE
0095                * MASS STORAGE MODE USED IN WORDS 3 AND 4
0096 FC70 0000      CLIST4 DATA 0          WORD 0, CLEAR FLAGS
0097 FC72 0000             DATA 0          WORD 1, CLEAR FLAGS
0098 FC74 0300             DATA >0300      WORD 2, READ FROM DISK DS1 COMMAND
0099 FC76 0000             DATA 0          WORD 3, (& 4) TRACK, SURACE AND
0100 FC78 0D00             DATA >0D00      WORD 4, SECTOR ADDR - MASS STORGE
0101 FC7A 0080             DATA >80        WORD 5, READ THE SECTOR
0102 FC7C 0000             DATA 0          WORD 6, NEW ADDRESS OF
0103 FC7E FD80             DATA FNLMSG     WORD 7, FINAL ADDRESS OF MESSAGE
0104 FC80 0000             DATA 0          WORD 8, NO CHAINING
```

FIGURE 2-11. DEMO PROGRAM TO READ TO/WRITE FROM DISK (SHEET 2 OF 6)

```
0106                ** *  *  *  *  *  *  *  *  *  *  *  *  *  *  *  *  *  *  *
0107                ** TASK AREA
0108                ** *  *  *  *  *  *  *  *  *  *  *  *  *  *  *  *  *  *  *
0109                ** ROUTINE TO SEND 3 COMMAND LIST ADDRESS BYTES
0110                ** TO DISK CONTROLLER MODULE THROUGH CRU
0111                **
0112 FC82 02E0      START  LWPI >FC00        DEFINE WORKSPACE POINTER
     FC84 FC00
0113 FC86 020C             LI   R12,>210     CRU SOFTWR BASE ADDR
     FC88 0210
0114                * INITIALIZE CONTROLLER THROUGH CRU
0115 FC8A 1D0E             SBO  14           RESET CONTROLLER
0116 FC8C 1E0E             SBZ  14           RELEASE RESET, OPERATE
0117                * TEST CRU CONDITIONS; IF ACCEPT OR BUSY NOT ZERO,
0118                * USE RESET TO CLEAR THESE (IN ACTUAL PRACTICE,
0119                * USE LOOP UNTIL BITS BECOME SET)
0120 FC8E 1F0B      ACEP1  TB   ACCEPT        ACCEPT = ZERO?
0121 FC90 13FE             JEQ  ACEP1        NO, LOOP UNTIL ACCEPT = 0
0122 FC92 1F0C      BUSY1  TB   BUSY          YES, BUSY = ZERO?
0123 FC94 13FE             JEQ  BUSY1        NO, LOOP UNTIL BUSY = 0
0124                * SET UP ADDRESS OF FIRST COMND LIST IN R2
0125 FC96 0202             LI   R2,CMLST1+1  MSB'S IN SECOND BYTE
     FC98 FC21
0126                * FOR FIRST ADDRESS BYTE, COMMAND BIT = 1
0127 FC9A 1D08             SBO  COMND        COMMAND BIT A ONE
0128 FC9C 3232             LDCR *R2+,8       ADDR BYTE TO CRU
0129 FC9E 1D0A             SBO  CUE          CAUSE INTERRUPT
0130 FCA0 1F0B      ACCEP1 TB   ACCEPT        CONTR RECV BYTE? (ACCEPT=1?)
0131 FCA2 16FE             JNE  ACCEP1       NO. LOOP UNTIL RECVD
0132 FCA4 1E0A             SBZ  CUE          YES, ACKNOWLEDGE THIS
0133 FCA6 1F0B      ACCEP2 TB   ACCEPT        ACCEPT = 0?
0134 FCA8 13FE             JEQ  ACCEP2       NO, LOOP UNTIL ACCEPT=0
0135 FCAA 1E08             SBZ  COMND        COMMAND=0 FOR BYTES 2 & 3
0136                * SEND SECOND ADDRESS BYTE
0137 FCAC 3232             LDCR *R2+,8       ADDR BYTE TO CRU
0138 FCAE 1D0A             SBO  CUE          CAUSE INTERRUPT
0139 FCB0 1F0B      ACCEP3 TB   ACCEPT        CONTR RECV BYTE? (ACCEPT=1?)
0140 FCB2 16FE             JNE  ACCEP3       NO. LOOP UNTIL RECVD
0141 FCB4 1E0A             SBZ  CUE          YES, ACKNOWLEDGE THIS
0142 FCB6 1F0B      ACCEP4 TB   ACCEPT        ACCEPT = 0?
0143 FCB8 13FE             JEQ  ACCEP4       NO, LOOP UNTIL ACCEPT=0
0144                * SEND THIRD ADDRESS BYTE
0145 FCBA 3212             LDCR *R2,8        ADDR BYTE TO CRU
0146 FCBC 1D0A             SBO  CUE          CAUSE INTERRUPT
0147 FCBE 1F0B      ACCEP5 TB   ACCEPT        CONTR RECV BYTE? (ACCEPT=1?)
0148 FCC0 16FE             JNE  ACCEP5       NO. LOOP UNTIL RECVD
0149 FCC2 1E0A             SBZ  CUE          YES, ACKNOWLEDGE THIS
0150 FCC4 1F0B      ACCEP6 TB   ACCEPT        ACCEPT = 0?
0151 FCC6 13FE             JEQ  ACCEP6       NO, LOOP UNTIL ACCEPT=0
0152                ** THIRD BYTE OF FIRST ADDRESS SENT
0153                ** SET UP COMMAND LIST 1 NUMBER IN ERROR MESSAGE
0154 FCC8 0205             LI   R5,>2031     ASCII SPACE AND 1 IN R5
     FCCA 2031
0155 FCCC C805             MOV  R5,@WORD     MOVE TO MESSAGE
     FCCE FD56
```

FIGURE 2-11. DEMO PROGRAM TO READ TO/WRITE FROM DISK (SHEET 3 OF 6)

```
                    ** CLEAR FIRST WORD OF EACH COMMAND LIST AND FINAL
                    ** MESSAGE LOCATION IN CASE THIS DEMO PROGRAM IS
                    ** EXECUTED AGAIN AND AGAIN
FCD0 04E0                 CLR  @CLIST1
FCD2 FC34
FCD4 04E0                 CLR  @CLIST2
FCD6 FC48
FCD8 04E0                 CLR  @CLIST3
FCDA FC5C
FCDC 04E0                 CLR  @CLIST4
FCDE FC70
FCE0 04E0                 CLR  @FNLMSG      ZERO PREVENTS MESSAGE WRITE
FCE2 FD80
                    ** SET UP FOR FIRST WORD OF EACH COMMAND LIST TO BE
                    ** BE READ BY TIMEOUT AND OPERATION COMPLETE ROUTINE.
                    ** INCREMENT NUMBER OF COMMAND LIST IN ERROR
                    ** (BIT 0 A ONE = COMMAND COMPLETE)
                    ** NO ERROR CORRECTION ROUTINE PROVIDED IN THIS DEMO PROGRAM
FCE4 0204                 LI   R4,CLIST1    SET UP 1ST LIST ADDR
FCE6 FC34
FCE8 06A0                 BL   @TIMOUT      GO TIME OUT
FCEA FD20
FCEC 05A0                 INC  @WORD        INCREMENT WORD IN ERROR MESSAGE
FCEE FD56
FCF0 0204                 LI   R4,CLIST2    SET UP 2ND LIST ADDR
FCF2 FC48
FCF4 06A0                 BL   @TIMOUT      GO TIME OUT
FCF6 FD20
FCF8 05A0                 INC  @WORD        INCREMENT WORD IN ERROR MESSAGE
FCFA FD56
FCFC 0204                 LI   R4,CLIST3    SET UP 3RD LIST ADDR
FCFE FC5C
FD00 06A0                 BL   @TIMOUT      GO TIME OUT
FD02 FD20
FD04 05A0                 INC  @WORD        INCREMENT WORD IN ERROR MESSAGE
FD06 FD56
FD08 0204                 LI   R4,CLIST4    SET UP 4TH LIST
FD0A FC70
FD0C 06A0                 BL   @TIMOUT      GO TIME OUT
FD0E FD20
                    ** MESSAGE WRITTEN TO DISKETTE
                    ** THEN READ FROM DISKETTE TO NEW HOST MEMORY LOCATION
                    ** NOW WRITE TO TERMINAL FROM NEW HOST ADDRESS
FD10 2FA0           WRITE  XOP  @FNLMSG,14   WRITE MESSAGE USING TIBUG
FD12 FD80
FD14 0460                 B    @>80         BRANCH TO TIBUG, MISSION COMPLETE
FD16 0080
                    ** ERROR ROUTINES IF DEMONSTRATION FAILS, SUGGEST THAT
                    ** USER CHECK JUMPER SETTINGS TO SEE THAT THEY AGREE
                    ** WITH THE FORMAT SELECTED (DEFAULT IS IBM STANDARD,
                    ** SINGLE DENSITY. CHECK BOTH CONTROLLER AND DISK DRIVE.
FD18 2FA0           ERROR  XOP  @ERRMSG,14   SEND ERROR MESSAGE
FD1A FD3E
FD1C 0460                 B    @>80         BRANCH TO TIBUG FOR USER INPUTS
FD1E 0080
```

FIGURE 2-11. DEMO PROGRAM TO READ TO/WRITE FROM DISK (SHEET 4 OF 6)

```
0193                ** COMMAND TIMEOUT ROUTINE READS FIRST WORD OF COMMAND
0194                ** LIST FOR COMPLETION AND ALSO RUNS TIMER IN CASE
0195                ** COMPLETION DOES NOT OCCUR WITHIN A SPECIFIED TIME;
0196                ** IF TIMEOUT OCCURS OR ERROR FOUND, ERROR MESSAGE
0197                ** WRITTEN OUT.
0198 FD20 0205      TIMOUT LI   R5,10          SET UP MASTER COUNTER
     FD22 000A
0199 FD24 0706      TIMOU1 SETO R6             SET UP INNER COUNTER
0200 FD26 0606      TIMOU2 DEC  R6             DECREMENT INNER COUNTER
0201 FD28 16FE             JNE  TIMOU2         IF NOT 0, KEEP LOOPING
0202 FD2A C0D4             MOV  *R4,R3         LOOK AT WORD 0 FOR COMPLETION
0203 FD2C 1605             JNE  TIMOU4         IF COMPLETE GO CHECK ERROR
0204 FD2E 0605             DEC  R5             NOT COMPLETE, DECREMENT COUNTER
0205 FD30 16F9             JNE  TIMOU1         IF NOT 0, GO THRU INNER LOOP
0206 FD32 020B      TIMOU3 LI   R11,ERROR      SET UP ERROR RETURN
     FD34 FD18
0207 FD36 045B             RT                  ERROR RETURN (*R11)
0208 FD38 0A13      TIMOU4 SLA  R3,1           DO WE HAVE ERRORS ?
0209 FD3A 16FB             JNE  TIMOU3         IF YES, GO TO ERROR EXIT
0210 FD3C 045B             RT                  NO-ERROR EXIT (*R11)
0211                ** *  *  *  *  *  *  *  *  *  *  *  *  *  *  *  *  *  *
0212                ** DEMO MESSAGES
0213                ** *  *  *  *  *  *  *  *  *  *  *  *  *  *  *  *  *  *
0214                ****** ERROR MESSAGE ******
0215 FD3E    0A     ERRMSG BYTE >0A,>0D        LINE FEED CARRIAGE RETURN
     FD3F    0D
0216 FD40    45             TEXT 'ERROR IN COMMAND LIST '
     FD41    52
     FD42    52
     FD43    4F
     FD44    52
     FD45    20
     FD46    49
     FD47    4E
     FD48    20
     FD49    43
     FD4A    4F
     FD4B    4D
     FD4C    4D
     FD4D    41
     FD4E    4E
     FD4F    44
     FD50    20
     FD51    4C
     FD52    49
     FD53    53
     FD54    54
     FD55    20
0217 FD56 0000      WORD   DATA 0              WORD NUMBER PLACED HERE
0218 FD58    0A             BYTE >0A,>0D,>07
     FD59    0D
     FD5A    07
0219 FD5B    00             BYTE 0              END-OF-MESSAGE DELIMITER
```

FIGURE 2-11. DEMO PROGRAM TO READ TO/WRITE FROM DISK (SHEET 5 OF 6)

```
 0221                ****** MESSAGE TO WRITE AND READ TO/FROM DISKETTE ******
 0222 FD5C                  EVEN
 0223 FD5C    0A    MESG    BYTE >0A,>0D       LINE FEED/CR
      FD5D    0D
 0224 FD5E    43            TEXT 'CONGRATULATIONS, IT WORKS!'
      FD5F    4F
      FD60    4E
      FD61    47
      FD62    52
      FD63    41
      FD64    54
      FD65    55
      FD66    4C
      FD67    41
      FD68    54
      FD69    49
      FD6A    4F
      FD6B    4E
      FD6C    53
      FD6D    2C
      FD6E    20
      FD6F    49
      FD70    54
      FD71    20
      FD72    57
      FD73    4F
      FD74    52
      FD75    4B
      FD76    53
      FD77    21
 0225 FD78    0A            BYTE >0A,>0D,>07,>07
      FD79    0D
      FD7A    07
      FD7B    07
 0226 FD7C    00            BYTE 0             END-OF-MESSAGE DELIMITER
 0227 FD7E  FFFF   PATERN DATA >FFFF          FORMAT SECTORS WITH ALL ONES
 0228       FC82   FNLMSG END  START          BEGINNING OF STORED MESSAGE
NO ERRORS,      NO WARNINGS
```

FIGURE 2-11. DEMO PROGRAM TO READ TO/WRITE FROM DISK (SHEET 6 OF 6)

SECTION 3

COMMUNICATING WITH THE TM 990/303A DISK CONTROLLER

3.1 GENERAL

This section describes the methods for communication between a host microcomputer and the TM 990/303A disk controller. This section is designed to help the user construct the device service routine (DSR) for handling the storage and retrieval of data to and from the TM 990/303A. Included are the following means of communication:

1. Initial contact through Communication Register Unit (CRU) (section 3.3)

2. Communication via Command List in host memory (section 3.4)

3. Interrupts to both controller (section 3.2) and host (section 3.5)

4. Bootstrap load at powerup (section 3.6)

Initial method of communication would be through the CRU (1 above) or bootstrap load at powerup (4 above). Using the CRU, explained in section 3.3, a 20-bit address is passed to the disk controller; this is the address in host memory of a ten-word block that defines a command for the controller to execute. This ten-word block is called the Command List and is used to transfer command data to the controller and return status and error data to the host. Command data to the controller includes the number of bytes to transfer, data addresses in host memory and diskette, and the chaining address of the next Command List to be executed. The controller accesses the Command List via direct memory access. The controller executes the command in the Command List and reports back completion status (successful completion, error completion, etc.) via the same Command List written back to host memory. Communications through the Command List in memory is explained in detail in section 3.4.

The disk controller also can indicate command completion (successful or otherwise) via a dedicated interrupt. The specified interrupt level must be jumpered at the disk controller, and the host is responsible for enabling the interrupt at the CRU interface, in the Command List, at the host TMS 9901 interface, and at the host microprocessor interrupt mask. Interrupts to the host are covered in section 3.5.

Some transactions through the Command List require the use of data placed in other host memory blocks. A series of Command Lists can be "chained" by giving the memory address of the next Command List in the last two words of the present Command List; thus, one beginning Command List address entered through the CRU can be used to start execution of a series of commands defined in a series of Command Lists, each list located in its own memory location. If one command in the chain is terminated unnaturally, execution of the present command and future commands in the chain is terminated.

If jumpered, a bootstrap load can be initiated at powerup. In this case, the disk controller receives the Command List from a sector on a formatted diskette. This Command List usually is used to initialize the system and is explained in section 3.4.

3.2 CONSIDERATIONS

1. If an error occurs during a command to the disk controller, the executing command is terminated. If the command was part of a "chain" of commands, termination of this command also terminates the other commands in the chain. ("Chaining" is where the address of the next Command List is obtained from the previous Command List, etc.) Thus, it is recommended that an interrupt be issued at the end of every command to allow an interrupt service routine to determine corrective action should operation be terminated other than successfully.

2. Interrupt service routines at the host as a result of command completion should consider the following:

   - Maintenance by the host microcomputer of a Command List address table pointer showing the address of the just-completed Command List so that errors can be monitored by the interrupt service routine.

   - Error handling routines.

   - Reenabling of interrupts at the host CRU, TMS 9901, and microprocessor as well as specifying interrrupts in the Command List.

   Interrupts and interrupt service routines are covered in more detail in section 3.5.

3. Take care when writing to controller memory (e.g. via the Write Controller Memory command) in that parts of controller memory are reserved for functions such as memory mapping. The only areas of controller RAM that can be written to are from address $FC00_{16}$ to $FFFF_{16}$. See Figure 3-1.

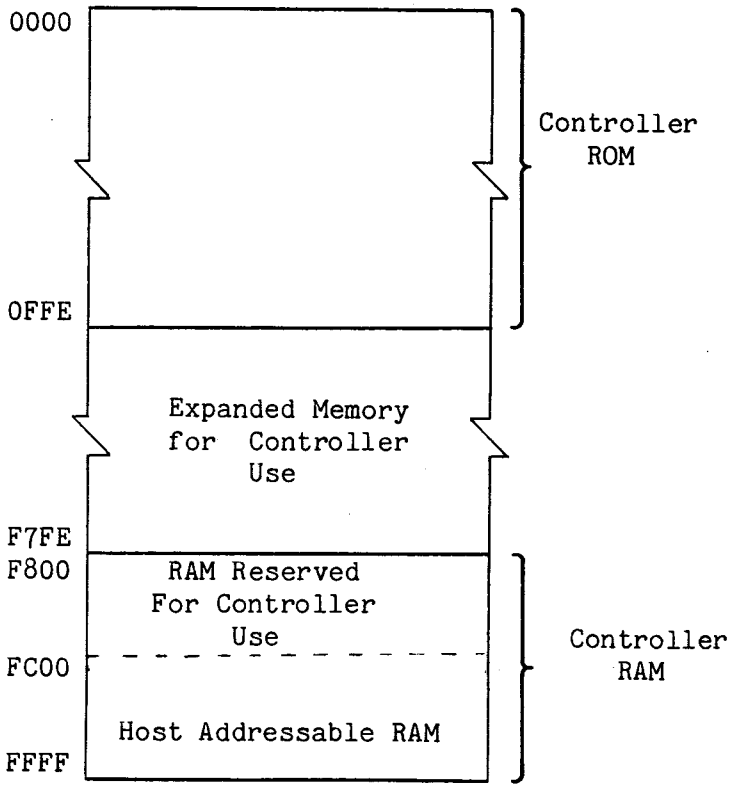


FIGURE 3-1. DISK CONTROLLER MEMORY MAP

4. Should the controller become "locked up," recovery would be thru writing a one to the RESET bit on the CRU; issuing a Reset Command would be ineffective while the controller is locked up.

5. Because the disk controller writes to an image of the Command List, setting error and status bits as dictated, the host should intiialize these bits to zeroes when building the list.

6. Completion of a command by the controller can be determined by the host by checking the Operation Complete bit of word 0 in the Command List. However, it is possible that the controller can inadvertently "lock up" during operation and never set the Operation Complete bit. Because of this, a timeout routine should also be used that would transfer host control to a recovery routine should the controller not complete its command in a proper time period. Such a timeout routine is shown in the demonstration program in Section 2 (Figure 2-11, starting at source line 193). In the routine, a continuous check is made of word 0 which has been initialized to all zeroes. If this state does not change (command completion will set one or more bits in the word) during the specified time period, host control is transferred so that a recovery can be made (instead of a lockup). See the next consideration below.

7. Code passed to the disk controller for controller execution must be thoroughly tested beforehand. The controller could become inadvertently "locked up" by writing to sensitive CRU or memory locations. (The host could check for a "locked up" situation via a timeout operation. Recovery could be through the RESET bit on the CRU interface.)

8. Care must be taken when passing code to the disk controller then commanding the controller to execute the code (Execute Controller Memory command). The passed code must not be position dependent in that relocatable portions are not relocated during the Write Controller Memory command such as that done by a relocating loader.

9. Do not read or write across 64 K memory boundaries (this consideration applies to systems using extended addressing). Instead, use multiple writes to write across such boundaries. For example, to write from diskette to host memory space $FC00_{16}$ to $101FE_{16}$, execute two writes -- one to $FC00_{16}$ to $FFFE_{16}$ and a second to 10000_{16} to $101FE_{16}$.

10. The controller can be used with different disk formats, and the user must be aware of changes in hardware as well as software when changing controller use from one format to another. For example, when changing from a standard-size diskette drive to a mini diskette drive, the user can specify the new format in software using the Define Drive command; however, he also must make jumper setting changes on the TM 990/303A board (i.e., change jumpers J10 and J11 from the STD setting to the MINI setting).

11. The onboard memory of the TM 990/100M or TM 990/100MA microcomputer cannot be accessed via DMA and <u>must not</u> be in the memory map of the expansion memory.

12. The TM 990/303A can be used with the TM 990/100M, TM 990/100MA, or TM 990/101MA microcomputer modules. If the TM 990/303A is to be used with the TM 990/101M module, the board PCB must be at revision B or later. Board revision level is shown on the non-component side following the PCB number

BD 994725-1. For example, PCB number BD 994725-1 A (B) or BD 994725-1 B are at the correct revision level; however, BD 994725-1A is <u>not</u> at the correct revision level. See section 1.1 for detail information.

13. Although each make of drive uses one of two diskette sizes (standard or mini), there are significant differences between the two. Therefore, it is not recommended to that a system configuration contain drives of different makes or models.

14. A full 65 K bytes of data cannot be transferred to or from the disk because the byte count is contained in a 16-bit word. The maximum length of data that can be transferred to or from the disk is $FF80_{16}$ for IBM single density, $FF00_{16}$ for IBM double density, and $FF60_{16}$ for TI double density.

3.3 COMMUNICATION THROUGH THE CRU (Software Base Address 210_{16})

Initial communication between the host and disk controller is through the CRU in which the disk controller is told the address of the Command List to be executed. The transfer of this address is via three data transfers in which

TO DISK CONTROLLER	CRU BIT ADDRESSED	FROM DISK CONTROLLER
LSB	HDW BASE + 0	0
↑	1	0
COMMAND	2	0
LIST	3	0
ADDRESS	4	0
	5	0
↓	6	0
MSB	7	0
COMMAND	8	0
0	9	0
CUE	10	0
0	11	ACCEPT
0	12	BUSY
INTERRUPT ENABLE	13	0
RESET	14	0
0	HDW BASE + 15	INTERRUPT ISSUED

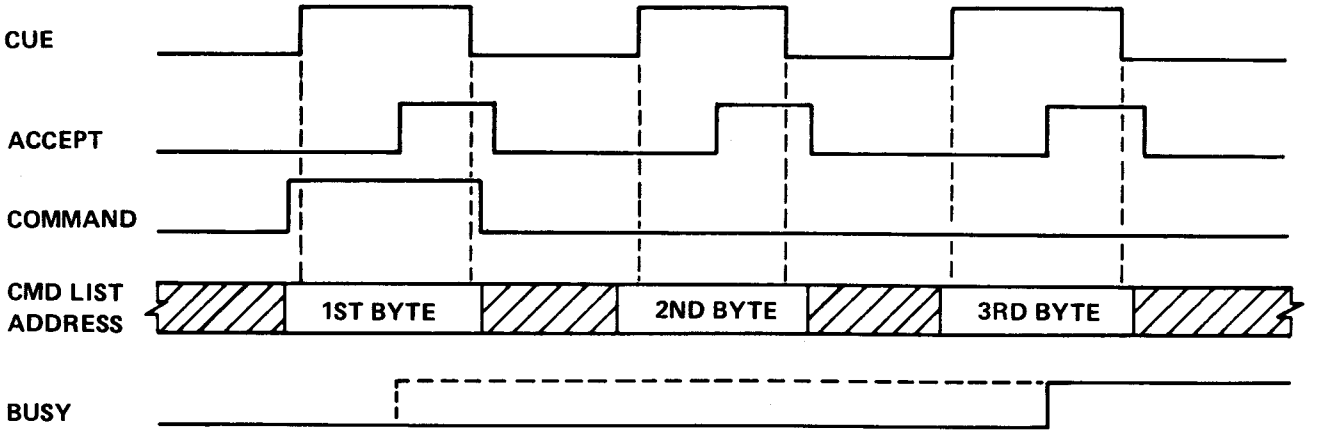


FIGURE 3-2. CRU INTERFACE AND TIMING

one address byte is sent to the controller in each transfer. Three bytes contain the Command List address (in the order sent):

- Four zeroes followed by the most significant four bits of the 20-bit address. If a memory board with extended addressing capability is used, set these bits to the value as mapped.
- The most-sigificant eight bits of the low-order 16 bits.
- The least-significant eight bits of the low-order 16 bits.

This communication is via 16 CRU bits starting at CRU software base address 0210_{16} as specified in a PROM at socket U13. (CRU software base address is the entire contents of R12.) The user can program his own PROM for an alternate address and insert it in this socket (this is described in Appendix E).

Figure 3-2 shows the signals, signal timing, and data bits that are output to and input from the disk controller over the CRU.

	To Disk Controller	CRU Bit and (Software, Hardware Base Addresses)		From Disk Controller
	LSB		(200, 100)	0
	↑			0
	Command			0
	List			0
	Address			0
				0
	↓			0
	MSB		(20E, 107)	0
	LSB	Hdwr Base + 0	(210, 108)	0
	↑	1		0
	Command	2		0
	List	3		0
	Address	4		0
		5		0
	↓	6		0
CRU	MSB	7		0
Bits	COMMAND	8		0
Used	0	9		0
	CUE	10		0
	0	11		ACCEPT
	0	12		BUSY
	INTERRRUPT ENABLE	13		0
	RESET	14		0
	0	Hdwr Base + 15	(22E, 117)	INTERRUPT ISSUED
	COMMAND		(230, 118)	0
	0			0
	CUE			0
	0			ACCEPT
	0			BUSY
	INTERRRUPT ENABLE			0
	RESET			0
	0		(23E, 11F)	INTERRUPT ISSUED

FIGURE 3-3. 32-BIT CRU INTERFACE BLOCK AS SHIPPED FROM FACTORY

HOST	DISK CONTROLLER
1. Initial Setup: a. Set CRU software base address in R12 b. Set counter of address bytes to 3*. c. Wait until BUSY & ACCEPT are zeroes	
2. Set COMMAND bit to one (next data byte is first of three bytes of address)	
3. Load data byte (address byte) onto CRU	
4. Set CUE bit to cause interrupt to disk controller (INT1-)	
	5. Is COMMAND bit a one? (1st byte is being sent?)
	6. If yes, set interrupt mask, set BUSY bit to a one, and set byte counter to 3*.
	7. Store 1st byte.
	8. Set ACCEPT bit to one.
9. Is ACCEPT bit a one? If no, wait.	
10. If yes, set CUE to zero. Set COMMAND bit to zero.	
	11. Is CUE a zero?
	12. If yes, set ACCEPT to zero.
	13. Decrement counter (to 2).
14. Is ACCEPT a zero? If no, wait.	
15. If yes, decrement counter (to 2).	
16. Load 2d byte on CRU; set CUE.	
	17. Is CUE a one?
	18. If yes, check COMMAND; if COMMAND is a one, return to step 6.
	19. If COMMAND a zero, store 2d byte.
	20. Set ACCEPT to one.
21. Is ACCEPT a one? If no, wait.	
22. If yes, set CUE to zero.	
	23. Is CUE a zero?
	24. If yes, set ACCEPT to zero.
	25. Decrement counter (to 1).
26. Is ACCEPT a zero? If no, wait.	
27. If yes, decrement counter (to 1).	
28. Load 3d byte on CRU, set CUE.	
	29. Is CUE a one?
	30. If yes, check COMMAND; if COMMAND is a one, return to step 6.
	31. If COMMAND a zero, store 3d byte.
	32. Set ACCEPT to one.
33. Is ACCEPT a one? If no, wait.	
34. If yes, set CUE to zero.	
	35. Is CUE a zero?
	36. If yes, set ACCEPT to zero
	37. Decrement counter (to 0).
	38. Counter = 0, so exit. BUSY bit is set.
39. Is ACCEPT a zero? If no, wait.	
40. If yes, decrement counter (to 0); exit.	

*NOTE: When (either) counter reaches zero, routine is exited.

FIGURE 3-4. COMMUNICATION BETWEEN THE HOST AND DISK CONTROLLER TO STORE COMMAND LIST ADDRESS THROUGH THE CRU

The CRU addressing scheme for each TM 990/303A takes up 32 bits of CRU address space; however, the user need use only 16 bits as explained earlier and in Figure 3-2. Figure 3-3 shows these 16 CRU bits in relation to the 32-bit block being addressed starting at CRU software base address 200_{16}. Note that because address line A11 is not used and A10 selects between the lower 8 CRU bits and upper 8 CRU bits, each 8-bit section is repeated; thus, the middle 16 bits can be used to have a contiguous 16-bit addressing scheme.

Figure 3-4 shows the "handshaking" between the disk controller and host to effect the transfer of the Command List Address. Figure 3-5 is example of code to make the transfer. In Figure 3-2, a software base address of 210_{16} is used (hardware base address 108_{16} as shown in the 32-bit CRU block in Figure 3-3). The middle 16 bits of the 32-bit CRU address block are used; the first and last 8 bits are reserved.

3.3.1 Output to Disk Controller Over CRU (Figure 3-2)

```
* EQUATE MNEMONICS
COMND   EQU  8       DISPLACEMENT ON CRU FOR COMMAND BIT
CUE     EQU  10      DISPLACEMENT ON CRU FOR CUE BIT
ACCEPT  EQU  11      DISPLACEMENT ON CRU FOR ACCEPT BIT
BUSY    EQU  12      DISPLACEMENT ON CRU FOR BUSY BIT
* BYTE STORAGE FOR COMMAND LIST ADDRESS
ADDR    BYTE >0F,>FE,00  3 ADDRESS BYTES (COMMAND LIST ADDRESS)
* LOAD CRU SOFTWARE BASE ADDRESS IN REGISTER 12
        LI   R12,>210
* SET UP COUNTER TO COUNT THREE BYTES
        LI   R1,3
* BUSY & ACCEPT MUST BE ZEROES BEFORE CONTINUING
ACCEP1  TB   ACCEPT  ACCEPT = ZERO?
        JEQ  ACCEP1  NO, LOOP
BUSY1   TB   BUSY    BUSY = ZERO?
        JEQ  BUSY1   NO, LOOP
* LOAD ADDRESS OF THREE BYTES OF COMMAND LIST ADDRESS
        LI   R2,ADDR
**
** ROUTINE TO SEND THREE ADDRESS BYTES
** OF COMMAND LIST THROUGH CRU
**
* FOR FIRST BYTE, SET COMMAND BIT TO 1 (MEANS 1ST BYTE BEING SENT)
        SBO  COMND
LDBYTE  LDCR *R2+,8  ADDR BYTE TO CRU
        SBO  CUE     CAUSE INTERRUPT TO DISK CONTROLLER
ACCEP2  TB   ACCEPT  DISK ACKNOWLEDGES BYTE RECEIVED????
        JNE  ACCEP2  NO, LOOP UNTIL CONTROLLER SETS ACCEPT TO ONE
        SBZ  CUE     YES, ACKNOWLEDGE CONTROLLER SETTING ACCEPT BIT
        SBZ  COMND   FIRST BYTE SENT, COMMAND = 0 LAST 2 BYTES
ACCEP3  TB   ACCEPT  CONTROLLER RETURNS ACCEPT BIT TO ZERO???
        JEQ  ACCEP3  NO, LOOP UNTIL CONTROLLER SETS IT TO ZERO
        DEC  R1      YES, THIRD BYTE SENT????  (R1 EQUALS ZERO?)
        JNE  LDBYTE  NO, LOOP, LOAD ANOTHER BYTE ON CRU
        .            YES, CONTINUE
        .
        .
```

FIGURE 3-5. PROGRAM TO PASS COMMAND LIST ADDRESS

3.3.1.1 Command List Address (Bits 0-7). This is the address in host memory of the Command List. The Command List must be located in host memory on an even byte boundry (LSB of the address a zero).

3.3.1.2 COMMAND Bit (Bit 8). Set the COMMAND bit to a logical one for transfer of the first byte of the Command List address, and set it to a logical zero for the second and third bytes of the Command List address. The COMMAND bit is valid only when the CUE bit (bit 10) is a logical one.

3.3.1.3 CUE Bit (Bit 10). Causes an interrupt to INT1- of the disk controller to initialize CRU data transfer. Checked as CRU bit during transfer of bytes two and three of Command List address over CRU.

3.3.1.4 INTERRUPT ENABLE Bit (Bit 13). Set this bit to a logical one to permit the disk controller hardware to issue interrupts to the host as directed by the Command List. Set this bit to a logical zero to clear the INTERRUPT ISSUED bit (bit 15). During interrupt-driven operation, this bit is normally set to a one to enable interrupts, then a zero to clear the INTERRUPT ISSUED bit (bit 15), then a one to re-enable interrupts. The host is also responsible for enabling interrupts at both the host TMS 9901 and microprocessor, for specifying in the Command List (word 2) that an interrupt is wanted at command completion, and for properly jumpering J3 for the correct interrupt level.

3.3.1.5 RESET Disk Controller Bit (Bit 14). Set this bit to a logical one to cause an unconditional reset of the disk controller (same as a powerup reset but not the result of the RESET switch toggled). This could be used to recover from a software "lockup" of the controller. After resetting the controller, set (toggle) RESET to a zero to allow normal operation to resume. RESET causes execution of the following:

1. Disable interrupts.
2. Turn off the Write gate.
3. Unload head from disk surface.
4. Set BUSY bit on CRU interface (bit 12) to logical one.
5. Set up to receive CUE interrupt at CRU.
6. Initialize workspace address for timer routine.
7. Clear status flags.
8. Indicate track position unknown.
9. Perform controller RAM test.
10. Perform checksum test on ROM contents.
11. Check accuracy of TMS 9901 timer.
12. Turn on LED DS1.
13. Set up CRU interface to receive commands.
14. Enable interrupts on controller.
15. Set BUSY bit on CRU interface (bit 12) to logical zero.
16. Enter an idle loop, wait for an interrupt (for example, interrupt on CRU specifying Command List address is at CRU).

3.3.2 Input From Disk Controller Over CRU (Figure 3-2)

3.3.2.1 ACCEPT Bit (Bit 11). A logical one sensed at this bit indicates that the disk controller has recognized the enabled CUE bit and has read the COMMAND bit and address bits. The ACCEPT bit can be set to a one only if the CUE bit is a one and can be set to a zero only if the CUE bit is a zero; this means that the ACCEPT bit can change state only if the CUE bit has already been changed to that state.

3.3.2.2 BUSY Bit (Bit 12). A logical one sensed at this bit indicates that the disk controller is currently executing a command and is unable to accept a new command. This bit will be a zero when the disk controller is not executing a command and is awaiting further command input. The BUSY bit should not be tested to determine if the disk controller has completed a command; instead, check the OPERATION COMPLETE bit in the first word of the command's Command List for this status. The BUSY bit should be a zero before addressing the CRU to transfer the Command List address.

3.3.2.3 INTERRUPT ISSUED Bit (Bit 15). A logical one sensed at this bit indicates that the disk controller has issued an interrupt. This bit is cleared by writing a zero to the INTERRUPT ENABLE bit (bit 13). Clearing this bit and re-enabling the interrupt should be part of the interrupt service routine (see paragraph 3.3.1.4).

3.4 COMMUNICATION THROUGH MEMORY (COMMAND LIST)

The Command List is another means of communication between the disk controller and the host microcomputer. This list is a ten-word block, shown in Figure 3-6, of system memory that is accessed by the host directly and by the disk controller via direct memory access. The address of this block is given to the disk controller via the CRU as explained in section 3.3 or via the last two words of the presently executing Command List ("chaining").

CAUTION

Do not place the Command List in ROM. It is important that the disk controller write back to the Command List showing errors, command completion, etc. The Command List must be in RAM.

A summary of the Command List (Figure 3-6) is as follows:

Word 0: Disk Controller Primary Status. This contains three data bits designating that the disk controller has completed its operation (OC, bit 0), or that at least one of several errors occurred (ER, bit 1), or that the controller issued an interrupt upon completion (IO, bit 2). There are also four bits explaining status of the errors incurred by the disk unit and a bit indicating the error indicator is in word 1; other error status bits are in Word 1. (Section 3.4.1.)

Word 1: Disk Controller Secondary Status. This contains 13 bits indicating disk status and disk-type data (e.g., number of sides, diskette size, diskette format) from the disk drive as well as errors incurred by the disk unit. When an error is reported in this word, the unit error bit (bit 15) in Word 0 is set. (Section 3.4.2.)

NOTE

The status bits (8 to 15) do not represent the format specified by the Define Format command (command 10). Instead, these bit values are the values read via hardware on connector P4 (from the disk drive) and at jumpers J8 and J9. Only J8 is monitored by the controller -- for the bootstrap load function. J9 is not monitored by the controller, but its setting will be reflected in bit 10 of word 1 (if jumpered, a one).

Word 2: Command and Disk Unit ID: This word contains an eight-bit code for a command to the disk controller, two bits identifying which disk controller is to answer the command, and a flag field specifying additional command data to the disk controller. (Section 3.4.3.)

Words 3 & 4: These two words contain the storage address of the diskette data addressed. (Section 3.4.4.)

Word 5: This word contains the number of bytes to be transferred. This will be an even number with bit 15 forced to zero. This transfer count must be a multiple of a sector length (Section 3.4.5).

Words 6 & 7: These two words contain the 20-bit memory address of (1) the data to be transferrred to disk or (2) the location of a list which is used by some commands. (Section 3.4.6.)

Words 8 & 9: These two words contain the 20-bit memory address of the next Command List address. (Section 3.4.7.)

NOTE

Because the disk controller writes into the Command List to indicate the status of command completion, all bits of the Command List should be initialized by the host to a proper value (i.e., set all status and error bits to zeroes).

Word 0: Primary Status

OC	ER	IO	0	0	0	0	0	0	DE	0	ID	OV	0	SE	UE
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15

Check if ER bit (1) is Set

OC = Operation Complete
ER = Error Occurred
IO = Interrupt Occurred
DE = Data Error
ID = Disk ID Error
OV = Overrun Error
SE = Data Field Search Error
UE = Unit Error

Word 1: Secondary Status

OL	0	WP	0	0	SI	ST	BC		DF	DS	WP	HD	DR	ND	SD
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15

Disk Drive — Individual Status — Drive Status (See below)

OL = Off Line
WP = Write Protect
SI = Seek Incomplete
ST = Self Test Error
BC = Bad Command

Drive Status Bits (reflect J8, J9 settings and inputs from disk drive via connector P4):

Bit 8: Spare (P4-24)
Bit 9: DF = Diskette Format; 0 = TI Double, 1 = IBM (J8)
Bit 10: DS = Diskette Size; 0 = Mini, 1 = Standard (J9[1])
Bit 11: WP = Diskette Write Protected?; 0 = No, 1 = Yes (P4-44)
Bit 12: HD = Head at Track 00?; 0 = No, 1 = Yes (P4-42)
[2]Bit 13: DR = Drive Ready?; 0 = No, 1 = yes (P4-22)
[2]Bit 14: ND = New Diskette Inserted?; 0 = No, 1 = Yes (P4-12)
Bit 15: SD = Sides on Diskette; 0 = 1 side, 1 = 2 sides (P4-10)

NOTES: 1. J9 not used by controller; if jumpered = 1, not jumpered = 0.
2. For mini drives, bits 13 (DRIVE READY) and 14 (NEW DISKETTE) are not used; these signals are not provided via connector P4 for mini drives (only standard drives).

Word 2: Command to Disk Drive

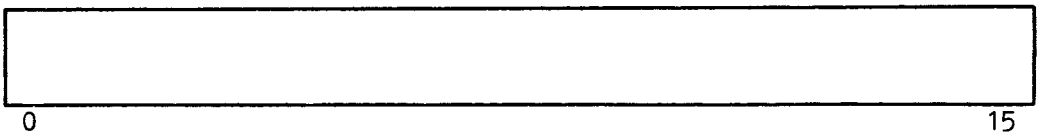
Command Number	IE	DV	0	0	0	0	Disk #
0 … 7	8	9	10			13	14 15

Flags

IE = Interrupt Enable Flag
DV = Data Verify Flag

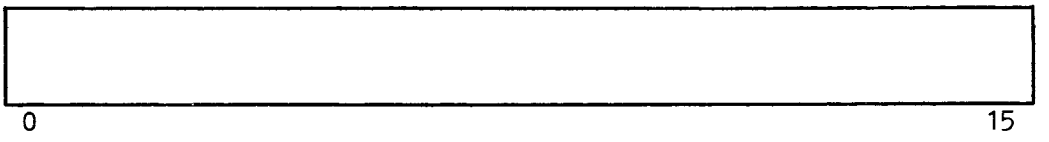
FIGURE 3-6. TEN-WORD COMMAND LIST (Sheet 1 of 2)

Word 3: Storage Address (Most Significant Word or Track Number)

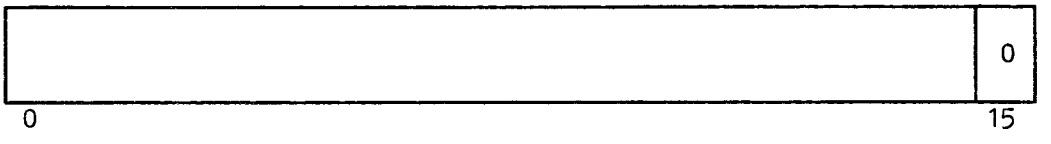


Bit 0: 1 = Mass storage mode, 0 = Physical storage mode

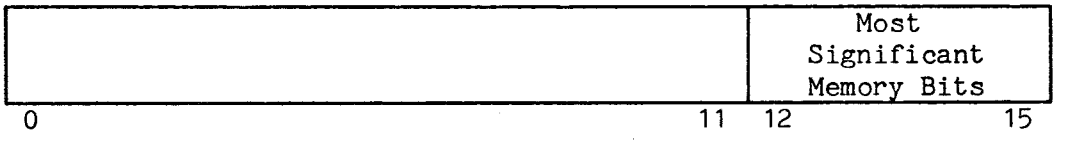
Word 4: Storage Address (Least Significant Word)



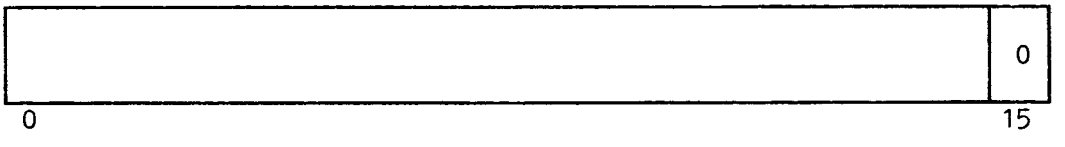
Word 5: Byte Count



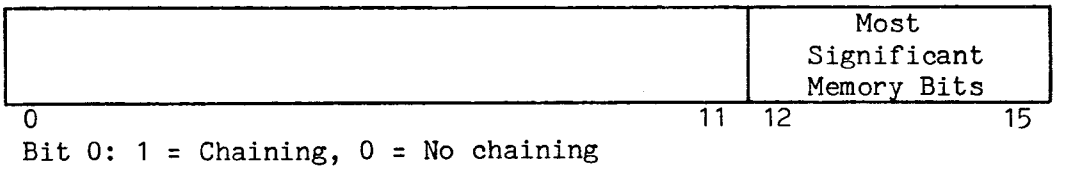
Word 6: Memory Address (Most Significant Word)



Word 7: Memory Address (Least Significant Word)



Word 8: Next Command Chain Address (Most Significant Word)



Bit 0: 1 = Chaining, 0 = No chaining

Word 9: Next Command Chain Address (Least Significant Word)

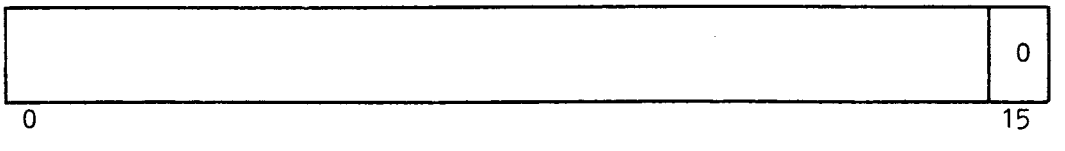
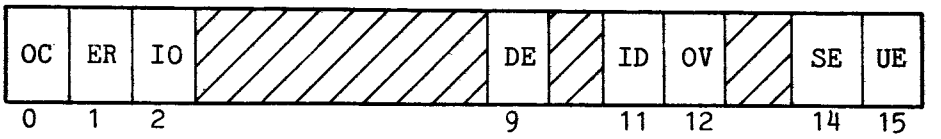


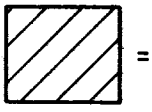
FIGURE 3-6. TEN-WORD COMMAND LIST (Sheet 2 of 2)

3.4.1 Word 0, First Status and Error Indicator Word



OC = Operation Complete
ER = Error Occurred
IO = Interrupt Occurred
DE = Data Error

ID = Disk ID Error
OV = Overrun Error
SE = Data Field Search Error
UE = Unit Error



= SET TO ZEROES, RESERVED

3.4.1.1 Word 0, Bit 0, Operation Complete (OC). This bit is set when the command (in word 2) has been completed successfully or has been terminated as the result of a an error (error causes are decoded by bits in words 0 and 1). Monitor this bit to determine command completion (rather than the BUSY bit which remains on for all commands).

3.4.1.2 Word 0, Bit 1, Error Occurred (ER). This bit is set when a command is terminated because of an error. Cause of the error is indicated by the error bits in words 0 (bits 9, 11, 12, and 14) and 1 (bits 5, 6, or 7). Since error indicators are in both words 0 and 1, monitor bit 15 of word 0 (UE); if a one, the enabled error indicator is in word 1; if a zero, the enabled indicator is in word 0.

3.4.1.3 Word 0, Bit 2, Interrupt Occurred (IO). When set, the disk controller had issued an interrupt to the host upon command completion. The interrupt enable bit on the CRU (CRU bit 13) must be set to a one and the Interrupt Enable (IE) flag in List Word 2 must be set to enable interrupts. After an interrupt occurs, interrupts must be cleared and re-enabled by setting the CRU Interrupt Enable bit to a one, then a zero, and then a one. Level of the interrupt is jumper selectable at jumper J3 as explained in Table 2-1.

NOTE

The following bits in word 0 (bits 9 and 11 to 15) and bits 5, 6, and 7 in word 1 explain errors. The Error bit (bit 1) will be set and the command will terminate if one of these error indicators are set. Bit 15 of word 0 in dicates whether the set error bit is in word 0 (bit 15 a 0) or word 1 (bit 15 a 1).

3.4.1.4 Word 0, Bit 9, Data Error (DE). This bit is set when an error occurs during the reading of the data field when executing a Read Data command (command 03) or when executing a Read Deleted Data command (command 0A) or during the reading of the ID field when executing Read ID command (command 0C). A data error occurs when the calculated cyclic redundancy check (CRC) word does not match the precalculated CRC value written on the disk for the respective field. Before a command is terminated because of this error, four attempts to correctly read the data will be made. This error sets the ER bit (word 0, bit 1) and terminates the command.

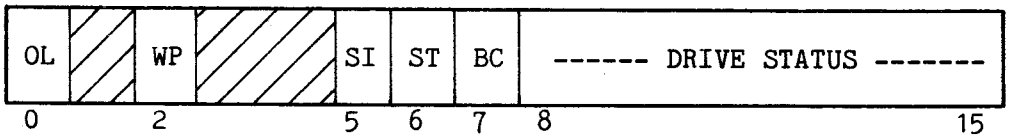
3.4.1.5 Word 0, Bit 11,Disk ID Error (ID). This bit is set when an unsuccessful search is made by the disk controller for the track ID in the header area of each sector. Five tracks will be searched for this header, and four tries will be made at each track. This error sets the ER bit (word 0, bit 1) and terminates the command.

3.4.1.6 Word 0, Bit 12, Overrun Error (OV). This bit will be set if the DMA interface cannot transfer data at the rate required by the disk controller. This error will occur during execution of the Read Data or Write Data commands (specified in Word 2). This error sets the ER bit (word 0, bit 1) and terminates the command.

3.4.1.7 Word 0, Bit 14, Search Error (SE). This bit is set when the disk controller fails to read a data field in 3 to 5 milliseconds after the track and sector ID field has been read (could be bad track format-check jumper settings on controller and disk drive, they should match the desired drive format). This error sets the ER bit (word 0, bit 1) and terminates the command.

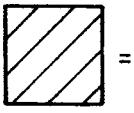
3.4.1.8 Word 0, Bit 15, Unit Error (UE). This bit is a logical OR of the error indicators in word 1 (i.e., if any of the three errors indicated in word 1 are active, this indicator is set). This error also sets the ER bit (word 0, bit 1), which allows the host to more quickly determine system error. If the ER bit is set, a check of the UE bit will indicate whether to scan word 1 for the error (UE bit set) or to scan word 0 for the error (UE bit reset). This error indication sets the ER bit (word 0, bit 1) and terminates the command.

3.4.2 Word 1, Second Status and Error Indicator Word



OL = OFF LINE
WP = WRITE PROTECT

SI = SEEK INCOMPLETE
ST = SELF TEST ERROR
BC = BAD COMMAND



= SET TO ZEROES, RESERVED

3.4.2.1 Word 1, Bit 0, Unit Off Line Status (OL). This bit indicates that the DRIVEREADY- signal is not active (drive not ready). This signal becomes active (drive is ready) by the diskette being placed in a drive and both of the following:

- power is applied to the drive, and
- two index holes have been sensed.

The DRIVEREADY- line is disabled whenever the power is cycled, the door is opened, diskette removed, or side 1 (vs. side 0) of a single-sided diskette is selected, and consequently the OL bit is reset. This error sets two error bits (ER bit in word 0, bit 1, and the UE error in word 0, bit 15) and terminates the command.

3.4.2.2 Word 1, Bit 2, Write Protect Status (WP). This status bit will be set when an attempt is made to write to a diskette that has been write protected. Data cannot be written to a write-protected diskette. Figure 3-7 shows the write-protect tab locations. A diskette is write protected when:

- the write protect tab is <u>removed</u> from a standard-sized diskette
- the write protect tab is <u>installed</u> on a mini-sized diskette.

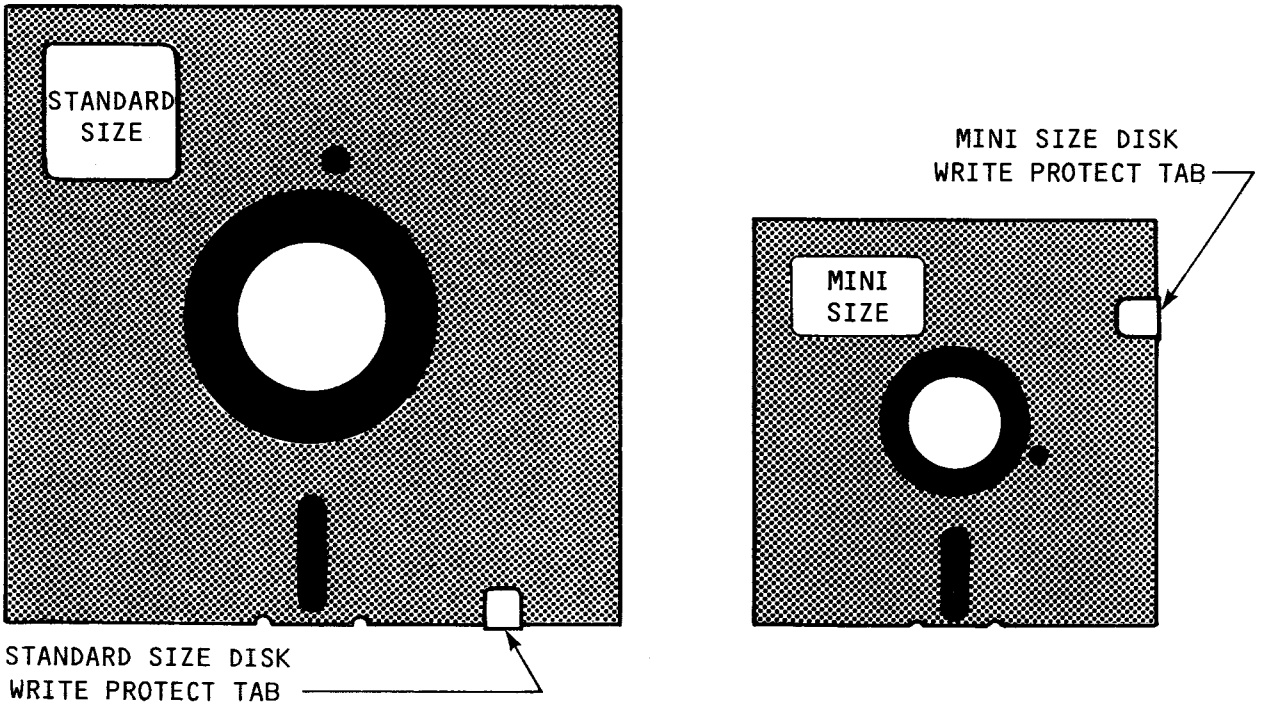
3.4.2.3 Word 1, Bit 5, Seek Incomplete Error (SI). This bit will be set after failure to sense a signal stating that the disk access arm reached the position over track 00 after completion of a Restore command. To see if this error can be overcome, issue another Restore command; if this seek is then successful, the SI bit will be reset. This error sets two error bits (ER bit in word 0, bit 1, and the UE error in word 0, bit 15) and terminates the command.

3.4.2.4 Word 1, Bit 6, Self Test Error (ST). This bit is set when an error occurs during the running of the controller self test. This diagnostic is executed by the Controller Test command (command code 01). This error sets two error bits (ER bit in word 0, bit 1, and the UE error in word 0, bit 15) and terminates the command.

3.4.2.5 Word 1, Bit 7, Bad Command Error (BC). This bit is set when an attempt is made to execute an invalid command (code not recognized) or execute a command with an invalid disk storage address. Examples are:

- mass storage address is not on a sector boundry
- the mass storage address is too large
- an illegal disk is defined in the parameter table such as single density TI

Disk storage address is contained in words 3 and 4 of the Command List and is computed as explained in section 3.4.4. This error sets two error bits (ER bit in word 0, bit 1, and the UE error in word 0, bit 15) and terminates the command.



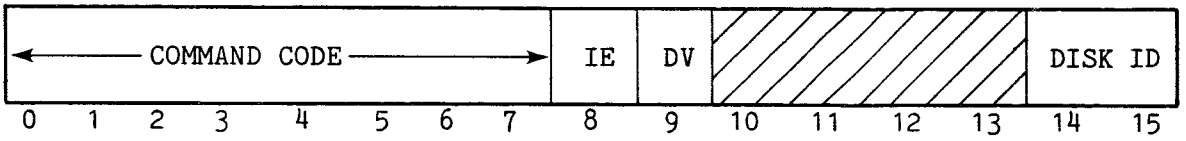
Note:
For standard drives, remove tab for write protect.
For mini drives, attach tab for write protect.

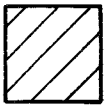
FIGURE 3-7. WRITE PROTECT TAB ON DISKETTE

3.4.2.6 Word 1, Bits 8 to 15, Drive Status. With bit 15 a spare (see below), bits 8 to 14 indicate specifications and status of the disk drive and diskette as seen by the disk controller. Some reflect the position of jumper-selectable options on the disk controller board. These bits indicate the following:

Bit 8: Spare Input (shows logic state of pin P4-24 from disk drive).
Bit 9: Diskette Format (Jumper J8): 0 = TI double
1 = IBM
Bit 10: Diskette Size (Jumpers J9, J10, & J11): 0 = Mini size
1 = Standard size
Bit 11: Diskette Write Protected: 0 = Not write protected
1 = Write protected
Bit 12: Head at Track 00: 0 = Not at track 00
1 = At track 00
Bit 13: Drive Ready: 0 = Drive not ready
1 = Drive ready
Bit 14: Diskette Changed: 0 = Same diskette (no change)
1 = Diskette changed since bit a zero
Bit 15: Number of Diskette Sides: 0 = Single sided
1 = Double sided

3.4.3 Word 2, Commands, Flags, and Disk ID



IE = INTERRUPT ENABLE FLAG DV = DATA VERIFY FLAG  = SET TO ZEROES, RESERVED

3.4.3.1 Word 2, Bits 0 to 7, Command Code. These bits contain the command to be executed by the disk controller. These command bytes are listed in Table 3-1 and explained in detail in Table 3-2.

3.4.3.2 Word 2, Bit 8, Interrupt Enable Flag. When set to one, the disk controller will issue an interrupt to the host when the command is either successfully or unsuccessfully completed. This is the preferred method of command completion since it allows the host to determine corrective action should the command be terminated without successful completion. Interrupts are covered more in detail in section 3.5.

3.4.3.3 Word 2, Bit 9, Data Verify Flag. The Data Verify flag pertains only to read and write commands. If set to one during a read command, data is read again from the diskette and compared to the data stored in host memory during the first read. If the bit is set during a write command, the controller performs a read-after-write and verifies the data written to diskette. A comparison error will set the Data Error flag (bit 9, word 0).

3.4.3.4 Word 2, Bits 14 & 15, Disk ID. These two bits contain the binary ID number of disk units 0 to 3, indicating which disk unit will be acted upon by the Command List. Connections to the disk drives of signals DSELECT1- to DSELECT4- from the disk controller select the specified drive. A jumper at the disk drive must correspond to the disk ID as follows: ID 00 corresponds to drive jumpered to DS1, ID 01 to DS2, ID 10 to DS3, and ID 11 to DS4.

TABLE 3-1. SUMMARY OF COMMANDS TO DISK CONTROLLER

Command Code (Hex)	Meaning
00	Store Status Command. Stores the diskette and disk drive status into a specified host memory location.
01	Controller Test Command. Executes disk controller self test.
02	Reset Command. Resets the controller.
03	Read Data Command. Reads data from diskette in sector blocks.
04	Write Data Command. Writes data to diskette in sector blocks.
05	Bootstrap Load Command. Causes the controller to execute a bootstrap load (executes a Command List at a predefined disk sector).
06	Read Controller Memory. Reads data from controller memory; writes it to host RAM.
07	Write Controller Memory. Writes data from host memory to controller RAM.
08	Execute Controller Memory. Causes controller to branch to a controller memory location as if it was the start of a two-word vector address of a BLWP instruction. The controller then executes the code at those vectors.
09	Format Track Command. Causes one track of diskette to be formatted according to specified or default format parameters.
0A	Read Deleted Data Command. Read data from a deleted sector.
0B	Write Deleted Data Command. Writes data to sector(s) and designates sector(s) as deleted sector(s).
0C	Read ID Command. Read ID field at specified track and sector
0D	Read Unformatted Command. Starting at a specified track and sector, read specified number of bytes without respect to data and control fields.
0E	Seek Command. Places read/write head at specified physical track of diskette.
0F	Restore Arm Command. Places read/write head at track 00.
10	Define Drive Command. Specifies characteristics of the diskette (e.g., mini or standard size, number of surfaces, disk density), and disk drive (e.g., step time, step settling time, head load and unload time), and format (e.g, number of tracks, sync type, sector interlace factor, bytes per sector, sectors per track).

TABLE 3-2. COMMANDS TO DISK CONTROLLER IN WORD 2

Command Code (Hex)	Meaning
00	Store Status Command. A 16-byte block of disc controller RAM is used to store a list of parameters describing the disk drive and diskette used. These parameters are default values or as specified by the Define Disk Drive command (10). The Store Status command reads these parameters into host memory at the host address specified in words 6 and 7 of the command list. It is not necessary to specify the controller memory address in words 3 and 4, nor specify the byte count in word 5. The parameter list in controller RAM can be changed by the Define Drive command.
01	Controller Test Command. This causes the controller self test to execute. This test is contained in controller firmware and consists of a checksum test on the firmware in addition to a test of the TMS 9901 timer and RAM. The RAM test consists of a walking one, walking zero, and non-destructive read/write. The timer test consists of timer operation, timer verification, and timer interrupt. If an error occurs during this self test, the Self Test error bit (ST, bit 6 in word 1) will be set.
02	Reset Command. The Reset command performs the same steps as for a reset request through the floppy disk controller CRU interface (bit E as explained in section 3.3.1.5) except that (1) a bootstrap load is not performed even though this operation is jumpered, and (2) a controller test is performed (same as Controller Test command above). Naturally, this command cannot bring the controller out of a "locked up" condition as commands cannot be sent to the controller when this condition exists. Recovery would be via the CRU RESET bit (3.3.1.5).
03	Read Data Command. (See cautions below and on next page.) The read data command reads the specified number of bytes (in Command List Word 5) from the disk storage address (list Words 3 and 4) to the host memory address (list words 6 and 7). Data will be read beginning at sector boundaries only. Naturally, the controller will not support reads from more than one disk in a single read operation. All reads will be on a word (16 bits) basis; thus, the byte count in word 5 of the Command List must be even.

CAUTION

Data is read and written in sectors only. This means that the byte count in word 5 of the Command List must be a multiple of sector length. During a read or write, the byte count in word 5 is decremented to determine the transfer count. When at the beginning of a sector read or write, the transfer count must be equal to a sector byte length or greater. If less than a sector byte length, the command is terminated and any value remaining to be transferred will not be transferred (i.e., a transfer count less than a sector byte length will be ignored). This may result in a transfer of less data than originally specified in word 5. (Note: this CAUTION also applies to the Write Data command and is repeated above the Write Data command on the next page.)

TABLE 3-2. COMMANDS TO DISK CONTROLLER IN WORD 2 (Continued)

Command Code (Hex)	Meaning
	After this command is read, the host can later check for unit errors returned in case the disk drive was off line (OL error, word 1, bit 0) or track 00 status was not found (Seek Incomplete error, word 1, bit 5). Bytes transferred will be returned in word 5. When this command is executed, a seek is executed to the specified track. A read of sector IDs is performed until the correct starting sector is read. If the sector ID is not found, four retries to read it will be made. Failure to read the sector ID after five tries will result in command termination and the ID error (word 0, bit 11) being set. After finding the correct sector ID, the data from that sector is read on a word-by-word basis. As each word is read, the transfer count from word 5 is decremented. Before a (next) sector is read, the (remaining) transfer count is checked; if the count is less than the byte count of a sector, the read operation ceases. If the entire sector is read and the remaining byte count is at least equal to the number of bytes in a sector, the next sector(or next cylinder in two-sided disks only) will be read. If the sector at the end of a track is read and the byte count is at least equal to the number of bytes in a sector, the controller will switch to the first sector on the next logical track (or next surface or next cylinder in two-sided disks) and resume the read operation. A CRC check is made after each sector is read. If a CRC error is encountered, the Data Error (DE, word 0, bit 9) will be set and the command terminated. **CAUTION** 1. Do not read or write across 64 K address boundaries (this applies to extended addressing only). For example, to write to host memory address $FE00_{16}$ to $101FE_{16}$, do two writes (or reads) -- one to $FE00_{16}$ to $FFFE_{16}$ and a second to 10000_{16} to $101FE_{16}$. 2. Data is read and written in sectors only. This means that the byte count in word 5 of the Command List must be a multiple of sector length. During a read or write, the byte count in word 5 is decremented to determine the transfer count. When at the beginning of a sector read or write, the transfer count must be equal to a sector byte length or greater. If less than a sector byte length, the command is terminated and any value remaining to be transferred will not be transferred (i.e., a transfer count less than a sector byte length will be ignored). This may result in a transfer of less data than originally specified in word 5.
04	Write Data Command. (See CAUTIONS above.) This command writes the specified number of bytes (Command List word 5) from the host memory address (list words 6 and 7) to the disk storage address (list words 3 and 4). Data will be written to the disk address beginning at a sector boundary, and data is transferred on a word (16 bits) basis.

TABLE 3-2. COMMANDS TO DISK CONTROLLER IN WORD 2 (Continued)

Commnand Code (Hex)	Meaning
	After this command is read, the host later can check for unit errors returned in case the disk drive was off line (OL, word 1, bit 0) or track 00 status was not found (Seek Incomplete, word 1, bit 5). The number of bytes transferred will be returned in word 5. When this command is executed, a seek is executed to the specified track (cylinder for two-sided disks). A read of sector IDs is performed until the correct starting sector is read. If the sector ID is not found, three retries to read it will be made. Failure to read the sector ID after five tries will result in command termination and the ID error (word 0, bit 11) being set. After finding the correct sector, the number of bytes in word 5 (transfer count) will be checked. If the transfer count is less than the number of bytes in a sector, the command terminates. As each word is written to the sector, the transfer count is decremented. After writing to each sector, the transfer count is checked; if less than the number of bytes in a sector, the command terminates. After writing to a sector, the CRC value for the sector is computed and entered. If after writing to a sector and the transfer count is <u>not</u> less than a sector byte count, the next logical sector on a track (or next cylinder or next surface in two-sided disks) is written to. If the transfer count is <u>not</u> less than a sector byte and the last sector on a track was written to, the first sector on the next track (or next cylinder or next surface in a two-sided disk) is written to. This continues until the sectors are written to with CRC values compiled and entered for each sector.
05	<u>Bootstrap Load Command</u>. This command causes the controller to seek to sector 0, track 00, surface 0 of unit 0 (designated DS1 by drive jumpers), ignore the first 56 bytes and interpret the next ten words as the Command List for that unit. The eleventh word is the checksum (two's complement of the ten-word Command List) for the Command List. The controller then executes the specified command. This command does not observe a chain address, and a chain to a next Command List will not occur. There are two command lists involved with this command, this bootstrap command list originating in memory and the command list on the diskette that will be executed. Upon command completion, values will be returned to words 0, 1, and 5 of the command list in memory for either command list -- for the command list in memory if it was prevented from executing, or for the command list on the diskette after it was executed. Values returned (to words 0, 1, and 5 respectively) include the primary status word (word 0) and data for the secondary status word (word 1) which will contain command status data for the command list, and the byte transfer length in word 5. (A power-up bootstrap load differs in that these three values are returned in words 1, 2, and 5 beginning at the address in words 8 and 9 of the command list on the diskette.) The checksum word causes the 11-word block to be summed to a 0 value. This command is also executed during a powerup reset where the

TABLE 3-2. COMMANDS TO DISK CONTROLLER IN WORD 2 (Continued)

Command Code (Hex	Meaning
	bootstrap load is jumpered. It is very useful in a system which does not have a controller initialization routine in ROM. The address in sector 0 of track 00 must already have been set up by a disk initialization routine. This is the same as the optional bootstrap load performed at powerup as explained in section 3.6. After a powerup bootstrap load, the two status words and transfer length are placed in host memory as if the address in the chain field was the address of the command list. These three words are not placed contiguously in the host memory. The two status words will be placed in words 0 and 1 of this ten-word area; the transfer length will be placed in word 5.
06	Read Controller Memory Command. This command causes the number of words specified in list word 5 to be read from the controller memory (ROM or RAM), starting at the address in list words 3 and 4, and to write these to the host memory address specified in list words 6 and 7.
07	Write Controller Memory Command. This command causes the number of words specified in list word 5 to be written from host memory to the (RAM only) controller memory space starting at the address specified in list words 3 and 4 (see CAUTION below). Starting address of host memory is specified in list words 6 and 7. All data loaded into the controller RAM must be absolute or position independent since the controller has no relocating capability such as that provided by a relocating loader. **CAUTION** The host can write only to specified addresses in controller RAM; writing outside of this area could seriously upset system operation. These address bounds for writing to the controller are from $FC00_{16}$ to $FFFE_{16}$.
08	Execute Controller Memory Command. This command causes the controller CPU to branch to the memory address specified in list words 3 and 4 as if this address was the two-word vector destination of a BLWP instruction and the new workspace pointer and program counter values were stored at this location. This command can be used in conjunction with the Write Controller Memory command (07) where a code sequence is passed to controller RAM, then executed by the Execute Controller Memory command. The code sequence requires a termination with an RTWP command, the same as a normal BLWP-initiated subroutine. Note the restrictions on controller RAM in the CAUTION above.
09	Format Track Command. This command causes one track of a diskette to be formatted according to the format parameters generated by the Define Drive command (command 10_{16}). If parameters have not been redefined by the Define Drive command, the default format is either IBM standard or TI double depending upon jumper J8. After formatting the track, a CRC test is made of each sector on the track. The track

TABLE 3-2. COMMANDS TO DISK CONTROLLER IN WORD 2 (Continued)

Command Code (Hex)	Meaning
	address is placed in words 3 and 4 of the Command List. The data pattern to place in the sector data field is specified in a 16-bit word in host memory; this memory location is placed in words 6 and 7 of the Command List. The controller will first check signal lines to determine if the disk unit is in a ready status (not off line) or if the diskette is not write protected; negative status will be indicated in Command List word 1 (OL or WP bit). Any negative condition will terminate the command. When the arm is correctly positioned, the controller will then proceed to format the diskette track according to the format specified in the Define Drive command (or default format). If flaws are found during a CRC check of a sector, a Data Error will be indicated (DE, word 0, bit 9). It will be the host's responsibilty to label the track as bad (e.g., generate table lookup of track status). IBM double density diskettes will automatically be formatted with track 00 in single density and the remaining tracks in double density. Diskette formats are explained in detail in Appendix C.
0A	Read Deleted Data Command. This command is similar to the Read Data command (03) except that this command will read and transfer the data in a sector's data field which has been designated as a deleted sector by the Write Deleted Data command (command $0B_{16}$). Sectors can be designated as deleted sectors if found to be bad. Attempts to read a deleted sector by the Read Data command (03_{16}) will result in both an ID Error and Data Error (ID and DE, word 0, bits 11 and 9). More than one sector can be read; however, the byte count in word 5 must be a multiple of the amount of bytes in a sector.
0B	Write Deleted Data Command. This command writes data to sectors as in the Write Data command (04) except that this command also writes a specified code in field AM2 of the control fields of a sector to designate the sector as a deleted sector. The specified code will vary depending upon diskette format, discussed in Appendix C. The write function is initiated the same as the Write Data command (04_{16}). This command is normally used to indicate bad tracks.
0C	Read ID Command. This command causes the controller to seek to the specified track and sector and read the ID Field, which is four or six bytes that follow address marker 1 (AM1). The ID field specifies the track number, head number, record number, and physical record length as shown for the various diskette formats in Appendix C. The ID field contents are written to the host address specified in words 6 and 7.

TABLE 3-2. COMMANDS TO DISK CONTROLLER IN WORD 2 (Continued)

Command Code (Hex)	Meaning
0D	Read Unformatted Command. This command reads control data as well as user data beginning at a specified sector. It causes a drive to seek to the specified track, wait for the correct address marker (AM1) of the specified sector, and attempt to synchronize with the next address mark (AM2). Following synchronization, data words will be transferred to memory starting at the disk address specified (words 3 and 4 of the Command List) until the specified number of bytes are transferred. However, data following the sector specified may not be read correctly because of possible loss of synchronization. No cyclical redundancy check (CRC) will be made, and reading will continue without respect to data field boundaries (reading to include data bytes, control fields, etc.); however, data following the sector specified may not be read correctly because of possible loss of synchronization.
0E	Seek Command. This command causes the read head to seek the specified physical track. Prior to initiating the seek, the controller determines disk unit status by checking the DRIVEREADY- line which tells if the diskette is installed, door closed, and at least two index holes sensed. If not ready, the Unit Error bit (UE, bit 15, word 0) and Off Line bit (OL, bit 0, word 1) will be set. Seeks will be to one cylinder at a time.
0F	Restore Arm Command. This command causes the read head to return to physical track 00. To determine if the head failed to reach track 00, test the Seek Incomplete error (word 1, bit 5).
10	Define Drive Command. This command is used to modify items in the Drive Parameter List, located in controller RAM, which describes the characteristics of the disk drive and diskette. The Drive Parameter List is shown in Figure 3-8. Figure 3-9 shows the Drive Parameter List default values brought in from controller ROM to controller RAM during reset or powerup (the default format is that set at jumper J8 with IBM single density if J8 jumpered, or TI double density if J8 not jumpered). Figures in Appendix A show values that could be used depending on which diskette format is used. NOTE The parameters in the Define Drive command take precedence over jumper positions set on the board (Table 2-1) pertaining to the same drive parameter. The address in host memory of the newly generated Drive Parameter List is placed in words 6 and 7 of the Command List. Not needed are specific values for the destination controller address of the Drive Parameter List in words 3 and 4 and the list length (16 bytes) in word 5 as these are provided by the controller.

TABLE 3-2. COMMANDS TO DISK CONTROLLER IN WORD 2 (Concluded)

Command Code (Hex)	Meaning
	A definition of the words in the Define Drive Parameter List is as follows: Word 0: Bits 0 to 7: Diskette Size (must be jumpered the same at J10 and J11): 0 = mini size, 1 = standard size Bits 8 to 15: Surfaces (1 or 2) Word 1: Number of Tracks (or cylinders for 2-sided disks) Word 2: Step Time in 10 microsecond units (time to step the disk read head one track distance) Word 3: Step Settling Time in 10 microsecond units (wait time for disk read head to settle at a track) Word 4: Head Load Time in 10 microsecond units (time to lower head to diskette) Word 5: Head Unload Timeout in milliseconds (time to wait after a command is complete before unloading head) Word 6: Bits 0 to 3: Disk Density: 0 = single density 1 = double density Bits 4 to 7: Sync Type: 0 = IBM 1 = TI Bits 8 to 15: Sector Interlace Factor: This is the distance, measured in physical sectors, between each contiguous logical sector. Because there is a time factor occurring while the data is being manipulated in a read-sector or write-sector operation, the physical location of contiguous logical sectors can be offset for efficient use of time. The distance between physical locations of contiguous logical sectors allows for normal disk travel during the time needed to manipulate the data written to or read from that sector (e.g., bring it to a buffer or bring it from a buffer). The factor number minus 1 is the number of physical records separating contiguous logical records. Table 3-3 is a list of interlace factors (1 to 25) showing the placement of logical sectors according to physical sector (in the column on the left). Word 7: Bits 0 to 5: Sectors per Track Bits 6 to 15: Bytes per Sector

Word 0	(1= Std) DISK SIZE (0= Mini)	SURFACES (1 or 2)
	0 — 7	8 — 15
Word 1	NUMBER OF TRACKS (CYLINDERS)	
	0 — 15	
Word 2	HEAD STEP TIME (10 us units)	
	0 — 15	
Word 3	STEP SETTLING TIME (10 us units)	
	0 — 15	
Word 4	HEAD LOAD TIME (10 us units)	
	0 — 15	
Word 5	HEAD UNLOAD TIMEOUT (milliseconds)	
	0 — 15	
Word 6	DENSITY 0=Single 1=Double	SYNC TYPE 0=IBM 1=TI	SECTOR INTERLACE FACTOR
	0 — 3	4 — 7	8 — 15
Word 7	SECTORS PER TRACK	BYTES PER SECTOR
	0 — 5	6 — 15

FIGURE 3-8. DRIVE PARAMETER LIST

			Hex Value
Word 0	DISK SIZE (Std=1) (0–7)	SURFACES (1) (8–15)	0101
Word 1	NUMBER OF TRACKS (77) (0–15)		004D
Word 2	HEAD STEP TIME (1000 10-us units) (0–15)		03E8
Word 3	STEP SETTLING TIME (1500 10-us units) (0–15)		05DC
Word 4	HEAD LOAD TIME (3500 10-us units) (0–15)		0DAC
Word 5	*HEAD UNLOAD TIMEOUT (1000 ms) (0–15)		03E8
Word 6	DENSITY FM=0 (0–3)	SYNC TYPE IBM=0 (4–7)	INTERLACE FACTOR=0 (8–15)	0000
Word 7	SEC/TRACK=26 (0–5)	BYTES PER SECTOR=128 (6–15)	6880

Notes: 1. These are the IBM single-density default values with jumper J8 installed.

2. For Shugart model 800 only.

FIGURE 3-9. DEFAULT VALUES FOR DEFINE DRIVE FORMAT BLOCK, SHEET 1 OF 2 (IBM SINGLE DENSITY FOR SHUGART 800)

			Hex Value
Word 0	DISK SIZE (Std=1) (0–7)	SURFACES (2) (8–15)	0102
Word 1	NUMBER OF TRACKS (77) (0–15)		004D
Word 2	HEAD STEP TIME (300 10-us units) (0–15)		012C
Word 3	STEP SETTLING TIME (1500 10-us units) (0–15)		05DC
Word 4	HEAD LOAD TIME (3500 10-us units) (0–15)		0DAC
Word 5	HEAD UNLOAD TIMEOUT (1000 ms) (0–15)		03E8
Word 6	DENSITY MFM=1 (0–3)	SYNC TYPE TI=1 (4–7)	INTERLACE FACTOR=0 (8–15)	1100
Word 7	SEC/TRACK=26 (0–5)	BYTES PER SECTOR=288 (6–15)	6920

Notes: 1. These are the default values with jumper J8 <u>not</u> installed.

2. For Qume DataTrak 8 only.

FIGURE 3-9. DEFAULT VALUES FOR DEFINE DRIVE FORMAT BLOCK, SHEET 2 OF 2 (TI DOUBLE DENSITY FOR QUME DATATRAK 8)

TABLE 3-3. STANDARD SIZE SECTOR PLACEMENT ACCORDING TO SECTOR INTERLACE FACTOR

Physical Sector Number	Logical Sector According to Interlace Factor*																								
	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
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	1	13	9	13	21	13	15	13	3	13	19	13	2	13	7	13	23	13	11	13	5	13	17	13	25
2	2	1	18	7	16	9	4	10	6	8	12	11	4	2	14	5	20	3	22	4	10	6	8	12	24
3	3	14	1	20	11	22	19	23	9	21	5	24	6	15	21	18	17	16	7	17	15	19	25	25	23
4	4	2	10	1	6	5	8	7	12	3	24	9	8	4	2	10	14	6	18	8	20	12	16	11	22
5	5	15	19	14	1	18	23	20	15	16	17	22	10	17	9	23	11	19	3	21	25	25	7	24	21
6	6	3	2	8	22	1	12	4	18	11	10	7	12	6	16	2	8	9	14	12	4	5	24	10	20
7	7	16	11	21	17	14	1	17	21	24	3	20	14	19	23	15	5	22	25	25	9	18	15	23	19
8	8	4	20	2	12	10	16	1	24	6	22	5	16	8	4	7	2	12	10	3	14	11	6	9	18
9	9	17	3	15	7	23	5	14	1	19	15	18	18	21	11	20	25	25	21	16	19	24	23	22	17
10	10	5	12	9	2	6	20	11	4	1	8	3	20	10	18	12	22	2	6	7	24	4	14	8	16
11	11	18	21	22	23	19	9	24	7	14	1	16	22	23	25	25	19	15	17	20	3	17	5	21	15
12	12	6	4	3	18	2	24	8	10	9	20	1	24	12	6	4	16	5	2	11	8	10	22	7	14
13	13	19	13	16	13	15	13	21	13	22	13	14	1	25	13	17	13	18	13	24	13	23	13	20	13
14	14	7	22	10	8	11	2	5	16	4	6	12	3	1	20	9	10	8	24	2	18	3	4	6	12
15	15	20	5	23	3	24	17	18	19	17	25	25	5	14	1	22	7	21	9	15	23	16	21	19	11
16	16	8	14	4	24	7	6	2	22	12	18	10	7	3	8	1	4	11	20	6	2	9	12	5	10
17	17	21	23	17	19	20	21	15	25	25	11	23	9	16	15	14	1	24	5	19	7	22	3	18	9
18	18	9	6	11	14	3	10	12	2	7	4	8	11	5	22	6	24	1	16	10	12	2	20	4	8
19	19	22	15	24	9	16	25	25	5	20	23	21	13	18	3	19	21	14	1	23	17	15	11	17	7
20	20	10	24	5	4	12	14	9	8	2	16	6	15	7	10	11	18	4	12	1	22	8	2	3	6
21	21	23	7	18	25	25	3	22	11	15	9	19	17	20	17	24	15	17	23	14	1	21	19	16	5
22	22	11	16	12	20	8	18	6	14	10	2	4	19	9	24	3	12	7	8	5	6	1	10	2	4
23	23	24	25	25	15	21	7	19	17	23	21	17	21	22	5	16	9	20	19	18	11	14	1	15	3
24	24	12	8	6	10	4	22	3	20	5	14	2	23	11	12	8	6	10	4	9	16	7	18	1	2
25	25	25	17	19	5	17	11	16	23	18	7	15	25	24	19	21	3	23	15	22	21	20	9	14	1

*NOTES:

1) The above table is for standard sized TI format. To change to standard IBM format, add the value one (1) to each logical sector number in the body of the table. For example, the logical sectors under interlace factor 1 (numbered 0 to 25) would be renumbered with the value 1 added to each (1 to 26).

2) This table does not pertain to mini sized diskettes.

3) Interlace Factor 0 and 1 are the same.

4) Columns under each Interlace Factor number show the physical consecutive (concatenated) postions of the logical sectors starting with sector 0 (top line). For example, looking downward at a diskette and following the sectors counterclockwise, the logical records for a diskette with an Interlace Factor of 2 would be a logical sector 0 followed by logical sector 13, then 1, 14, 2, 15, 3, etc.

3.4.4 Words 3 and 4, Storage Address on Diskette

Command List Words 3 and 4 may be used in two modes:

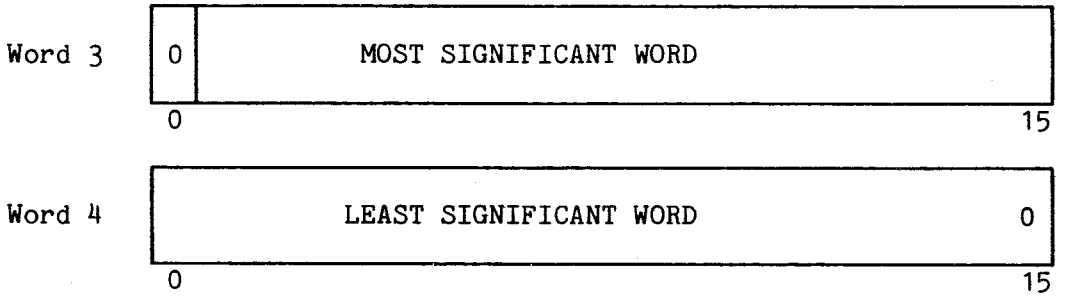
- mass storage mode (section 3.4.4.1)
- physical storage mode (section 3.4.4.2)

The modes are specified by a zero (mass storage) or a one (physical storage) in bit zero of word 3.

CAUTION

When words 3 and 4 contain a memory address (i.e., for the Read Controller Memory (06), Write Controller Memory (07), and Execute Controller Memory (08) commands, these words must also be the equivalent of a sector address. If not, an error will be given when the address is checked for being a valid sector address. It is recommended that address $FC00_{16}$ be used as this will not cause an error (also equivalent to a sector address).

3.4.4.1 Mass Storage Mode



Note: bit zero of word 3 is a 0 to specify mass storage.

In the mass storage mode, list words 3 and 4 comprise storage address on the diskette. By dividing this 32-bit number, the disk controller can determine physical track (cylinder), surface (head), and sector address. Diskette storage address (SA) is determined by the following equation:

Diskette Storage Address: $\{(T \times N_H \times N_S) + (H \times N_S) + S\} \times N_B$

Where:

T = logical track number (0 to 76)
H = surface number (0 or 1)
S = sector number (0 to 25)
N_H= number of surfaces per diskette (1 or 2)
N_S= number of sectors per track (16 or 26)
N_B= number of bytes per sector (128, 256, or 288)

The following are parameters for different diskette formats:

	IBM SD	IBM DD	TI DD	Mini SD	Mini DD
Sectors per Track	26	26	26	16	16
Bytes per Sector	128	256	288	128	256

Example 1. Address for logical track 10 (T), surface 0 (H), sector 3 (S) on an IBM single-density single-sided diskette:

Diskette Storage Address: $\{(10 \times 1 \times 26) + (0 \times 26) + 3\} \times 128$

$(260 + 0 + 3) \times 128$
$(263) \times 128 = 33{,}664 = 8380_{16}$

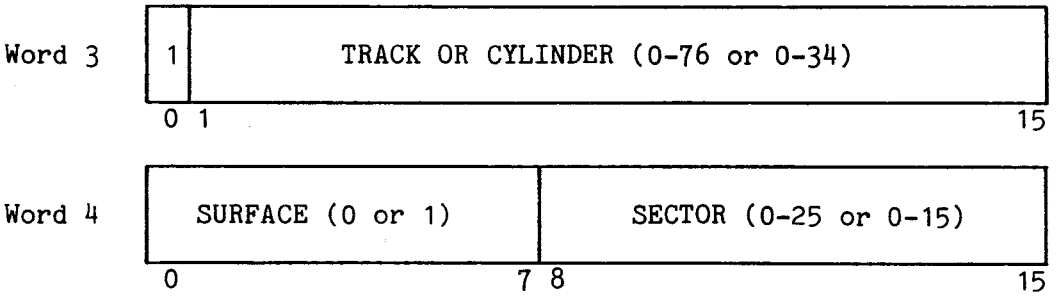
Example 2. Address for logical track 12 (T), surface 1 (H), sector 20 (S) on a TI double-sided double-density diskette:

Diskette Storage Address: $\{(12 \times 2 \times 26) + (1 \times 26) + 20\} \times 288$

$(624 + 26 + 20) \times 288$
$(670) \times 288 = 192{,}960 = 2F1C0_{16}$

The diskette storage address <u>must be a multiple of sector length</u>. With 31 bits the user has the capability to address up to 2^{31} or 2,147,483,648 bytes. The address range will be 0 to $2{,}147{,}483{,}646_{10}$ or $7FFF\ FFFE_{16}$ bytes.

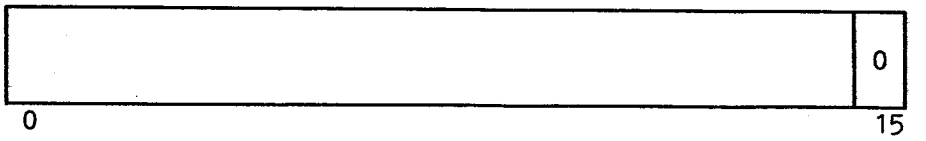
3.4.4.2 Physical Storage Mode



Note: bit zero of word 3 is a 1 to specify physical storage.

The sector field identifies the logical sector on the track. Note that TI double density format begins with sector 00 while the other four diskette formats supported by the TM 990/303A have sectors beginning with 01.

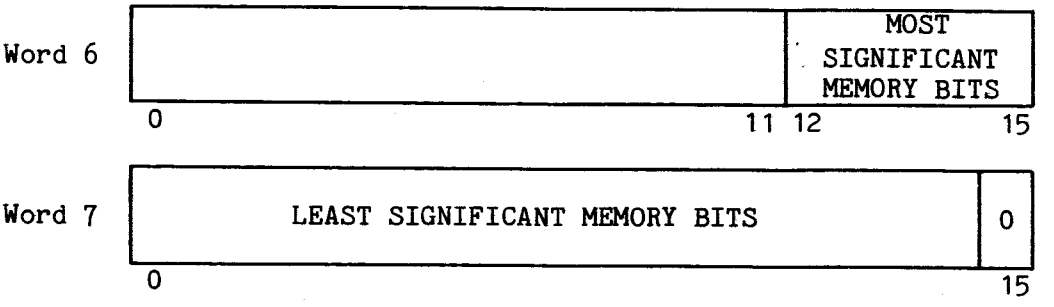
3.4.5 Word 5, Byte Count



Note: Bit 15 must be a zero (even byte count only)

Word 5 contains the amount of bytes to be transferred. This value must be even (bit 15 a zero), and must be a multiple of a sector length.

3.4.6 Words 6 and 7, Memory Address of Data to Transfer

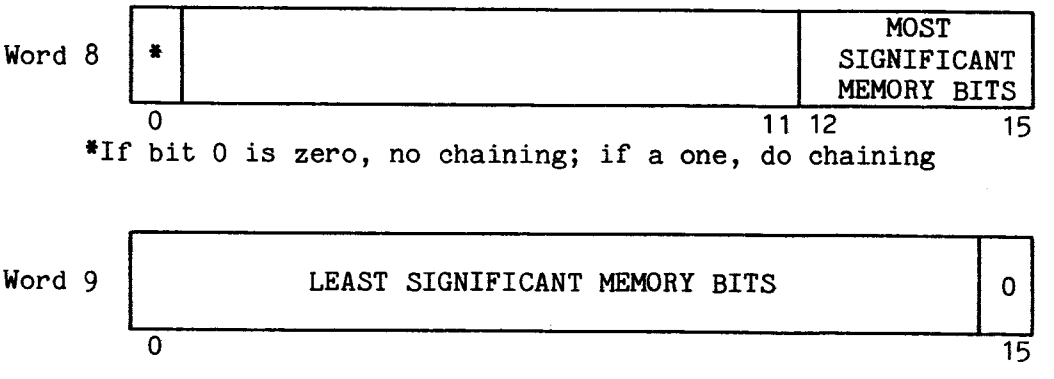


Note: Bit 15 of word 7 must be a zero (word boundary only)

Words 6 and 7 contain the beginning address in host memory for a data transfer. This is a 20-bit address with the extended address bits in bits 12 to 15 of word 6, and the least significant 16 bits in word 7 with bit 15 a zero (word boundary).

3.4.7 Words 8 and 9, Chain Address of Next Command List

In the bootstrap load Command List, these words contain the address where status wil be stored.



*If bit 0 is zero, no chaining; if a one, do chaining

Note: Bit 15 of word 9 must be a zero (word boundary only)

If bit 0 of word 8 contains a one, a chaining function will be executed and the disk controller will seek out this address as the beginning of the next Command List to be executed. In this manner a series of Command Lists can be executed. Words 8 and 9 contain the host memory address of the next Command List to be executed. Bit 0 of word 8 must contain a one in order to enable the chaining function. This is a 20-bit address with the extended address bits in bits 12 to 15 of word 8, and the least significant 16 bits in word 9 with bit 15 a zero (word boundary).

3.5 COMMUNICATION THROUGH INTERRUPTS

A third means of communication between the host microcomputer and the disk controller is through one interrupt to the host from the controller and one interrupt to the controller from the host. These interrupts are described as follows:

- From disk controller to host: An interrupt to a jumper-selectable interrupt level on the host that tells the host that a command has been completed. To determine the condition of completion (successful, error occurred, etc.), the host should monitor the status bits returned back to the Command List of the respective command. The disk controller is shipped jumpered for a level 2 interrupt to the host.

- From host to disk controller: An interrupt to level one of the disk controller that tells the controller that the host is to send data to the controller via the CRU. This interrupt is caused by setting the CUE bit to a one as described in section 3.3. Because the controller will not respond to an interrupt if busy, the host should first check the BUSY bit at the CRU (bit C_{16} using hardware base address 0108_{16}) before issuing the interrupt. This CRU interface is shown in Figures 3-2 and 3-3.

Both interrupts are maskable at the respective TMS 9901 parallel interfaces as well as through the interrupt mask at the microprocessor (if the mask level is set to a lower number than the interrupt level, the interrupt is masked out).

3.5.1 Command Completion Interrupt from Disk Controller to Host

Through an interrupt, the controller can tell the host microcomputer that a command has been completed. In order for the host to utilize interrupts, the host must supply vectors (workspace pointer and program counter values) for the interrupt in lower memory. The TM 990/100M TIBUG monitor comes from the factory with vectors for interrupt traps 3 and 4. The TM 990/101MA TIBUG monitor uses a scheme whereby all interrupt vectors are populated. The following discussion is for a TM 990/101MA microcomputer.

The interrupt level is jumper selectable on the TM 990/303A at J3 (jumper positions are explained in Figure 2-1 and Table 2-1). For example purposes, the following must be accomplished in order to enable a level 2 interrupt to be recognized at the host TM 990/101MA microcomputer (coding to accomplish this is shown in Figure 3-10):

1. Set up the level 2 interrupt link area at the TM 990/101 by:
   - Loading memory address $FF6E_{16}$ with 0420_{16} (BLWP instruction).
   - Loading memory address $FF70_{16}$ with the address of the 2-word vector area of the interrupt service routine (ISR).
   - Loading memory address $FF72_{16}$ with 0380_{16} (RTWP instruction).
2. At the two-word vector area of the ISR, load the ISR workspace pointer and ISR entry (PC value) addresses. Conclude the ISR with an RTWP.
3. Set the jumper on the disk controller at J3 to level 2 to use interrupt 2 in this example (this is the position as shipped).
4. Enable the interrupt level at the host microcomputer board's TMS 9901 so that it recognizes the desired interrupt level (level 2 in this example). To do this:
   - Load the TMS 9901 software base address in R12 (0100_{16} for TM TM 990/101 microcomputer).
   - Through the CRU set bit 0 to a zero to place the TMS 9901 in the interrupt mode (SBZ 0).

```
      IDT   'INTDEM'
      TITL  'INTERRUPT DEMO'
* SET UP INTERRUPT LINK AREA IN HOST RAM
      LI    R1,>FF6E      ADDR OF 3-WORD BLOCK IN INT LINK AREA
      LI    R2,>0420      BLWP MACHINE CODE
      MOV   R2,*R1+       MOVE TO INTERRUPT LINK AREA
      LI    R2,>FC00      ADDR OF VECTORS TO INT SERV RTN
      MOV   R2,*R1+       MOVE TO INTERRUPT LINK AREA
      LI    R2,>0380      RTWP MACHINE CODE
      MOV   R2,*R1        MOVE TO INTERRUPT LINK AREA
* SET UP TWO-WORD VECTOR TO INTERRUPT SERVICE ROUTINE (ISR)
      LI    R1,>FC00      ADDRESS OF VECTORS IN HOST RAM
      LI    R2,>FC80      ISR WORKSPACE POINTER
      MOV   R2,*R1+       MOVE TO VECTOR AREA (WORD 1)
      LI    R2,>FCA0      ISR ENTRY (PC VALUE)
      MOV   R2,*R1        MOVE TO VECTOR AREA (WORD 2)
* ENABLE TMS 9901 AT HOST TO RECOGNIZE INTERRUPT 2
      LI    R12,>100      9901 SOFTWARE BASE ADDRESS
      SBZ   0             9901 TO INTERRUPT MODE
      SBO   2             ENABLE INTERRUPT 2
* ENABLE TMS 9900 AT HOST TO RECOGNIZE INTERRUPT 2
      LIMI  2
* ENABLE INTERRUPTS AT CONTROLLER
      LI    R12,>210      CRU BASE ADDRESS OF CONTROLLER
      SBZ   13            SET TO ZERO
      SBO   13            TOGGLE BACK TO ONE
* IN BUILDING COMMAND LIST, BE SURE TO SET INTERRUPT
* ENABLE BIT TO A ONE (WORD 2, BIT 8)
      END
```

FIGURE 3-10. EXAMPLE OF CODING TO LOAD INTERRUPT LINK AREAS AND ENABLE INTERRUPTS ON A TM 990/101MA

- through the CRU set bit 2 to a one to enable interrupt 2 in this example (SBO 2).

5. Set the interrupt mask at the microprocessor to a level 2 or lower priority (LIMI 2 or higher number in operand for this example).
6. Through the CRU (explained in section 3.2), change (toggle) the Interrupt Enable bit (bit D_{16}) from a logical zero to a logical one.
7. When building the Command List, set the Interrupt Enable bit (bit 8 of word 2) to a one so that completion is signaled with an interrupt.

When the interrupt is issued, the Interrupt Occurred bit in the Command List (bit 2 of word 0) is set to one. To re-enable interrupts, set the Interrupt Enable bit at the CRU to zero, then a one, and repeat steps 1 to 6.

NOTE

It is not practical to use interrupts with chained commands unless an interrupt is requested for each command (not just for the last command). This is because if a error occurs during a command in the middle of the chain and no interrupt is requested, execution of the entire chain halts without informing the host through an interrupt. The host would have to poll the Command Completion bit of each Command List to find the Command List last completed.

The interrupt service routine in the host can be given several responsibilities:

1) Determine address of the just-completed Command List through a pointer pointing to the address of the last-executed Command List. Keep this pointer updated.

2) Determine if an error occurred during the last command by checking word 0, bit 1 (ER). If a one, check word 0, bit 15 (UE) to see if the error indicator is in word 1; if not, check the error indicators in word 0 (bit 15 a one means indicator is in word 1, a zero indicates indicator is in word 0).

3) Re-enable interrupts at the CRU and at TMS 9901 and microcomputer of host (see example at beginning of this section).

3.5.2 Command-Ready Interrupt from Host to Disk Controller

When the user wishes to call the controller to execute a Command List, he calls it through the CRU as explained in section 3.3. This call through the CRU actually causes an interrupt at the controller. This is the only interrupt to the disk controller from the host.

This interrupt is initiated by setting the CUE bit to a one. When this occurs, an interrupt is sent to INT1- of the controller TMS 9901 programmable interface. See section 3.3 for a full explanation of this interface.

3.6 POWERUP BOOTSTRAP LOAD OPTION

If jumpered, a bootstrap load feature allows the initialization of the system by executing a specified command list placed on the disk by the user. If jumpered at J1 and J2 (see Table 2-1), the bootstrap load occurs during one of the following:

- upon powerup of the disk controller,
- upon execution of the Bootstrap Load command (command 05),
- an active PRES.B- signal on the backplane.

NOTE

Powerup bootstrap load can be used only with the disk format specified at jumper J8; IBM single density, single sided, 8 inch (J8 jumpered) or TI double density, single sided 8 inch (J8 unjumpered).

This option is activated by the PRES.B- signal (not the IORST.B- signal). When executed, the host microprocessor is placed in a hold state until the bootstrap load is completed (see CAUTION on next page). To implement this option, the following jumpers must be installed:

- E1 to E2
- E4 to E5

The bootstrap load causes the controller to search for and execute a Command List in the diskette (which has been formatted accordingly by the user) located at the first sector of track 00 on disk unit 0 (DS1). The first 56

bytes are ignored; the next 10 words (16 bits) are interpreted as the command list and the eleventh word is used as a checksum word. This Command List cannot be chained to another command. Load the chain field (words 8 and 9) with the address in host memory where the controller will return a three-word block consisting of: the primary status word (same as word 0 of the Command List), the secondary status word (same as word 1), and the transfer length (same as word 5). These can be read for command completion status. It is the responsibility of the host to first format the diskette with the desired bootstrap Command List starting with the 57th byte of track 00, surface 0.

While the Command List is being retrieved, the host processor is held in an idle state by controller hardware holding HOLD.B- low. When the host processor is released from this hold state, it does a reset, obtaining WP and PC vectors from lower memory.

CAUTION

The disk controller could lock up the system during a bootstrap load under one of these circumstances: (1) A CRC error in reading the sector containing the Command List, (2) a checksum error in reading the Command List (perhaps failure to find the Command List), or (3) error during execution of the Command List read. Any of these will place the controller in the idle mode without releasing the host microprocessor which remains in the hold mode.

The procedure for preparing the diskette for the bootstrap load is as follows:

1. Write the command list in the first sector of track 00, side 0, of disk zero, starting with the 57th byte.

2. If the bootstrap load command list (see 1, above) is to read data from the disk and write this to host RAM, this data also must have been written onto the disk prior to executing the bootstrap load.

NOTE

This initialization of the system diskette must be done before the diskette is used by the host system bootstrap load routine.

SECTION 4

HARDWARE DESCRIPTION

4.1 GENERAL

An overview of a typical system using the TM 990/303A disk controller will be presented first, followed by a detailed description of the disk controller including the local processor, disk drive interface, host system interface, and read/write controller.

4.2 SYSTEM DESCRIPTION

A typical microcomputer disk system is shown in Figure 4-1. The major elements in the system are the TM 990/303A disk controller, TM 990/101 microcomputer, TM 990/203 memory, TM 990/510 card cage, terminal, and one to four disk drives.

The TM 990/303A disk controller communicates with the microcomputer system using the Communications Register Unit (CRU) for initialization and direct-memory access (DMA), for command, status, and data transfer. The disk controller can be configured to automatically load a program into memory upon system power-up.

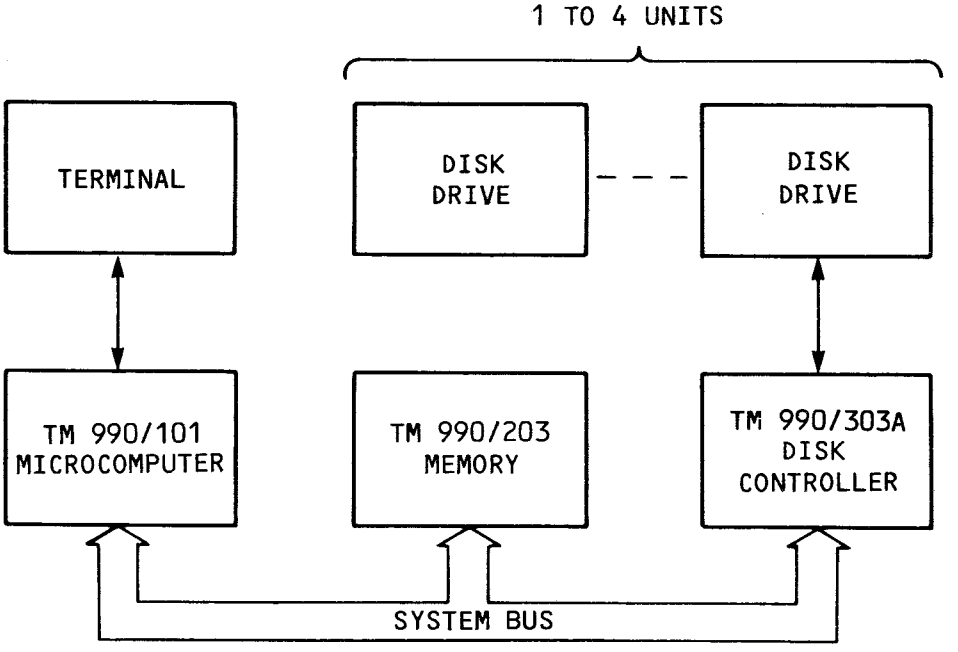
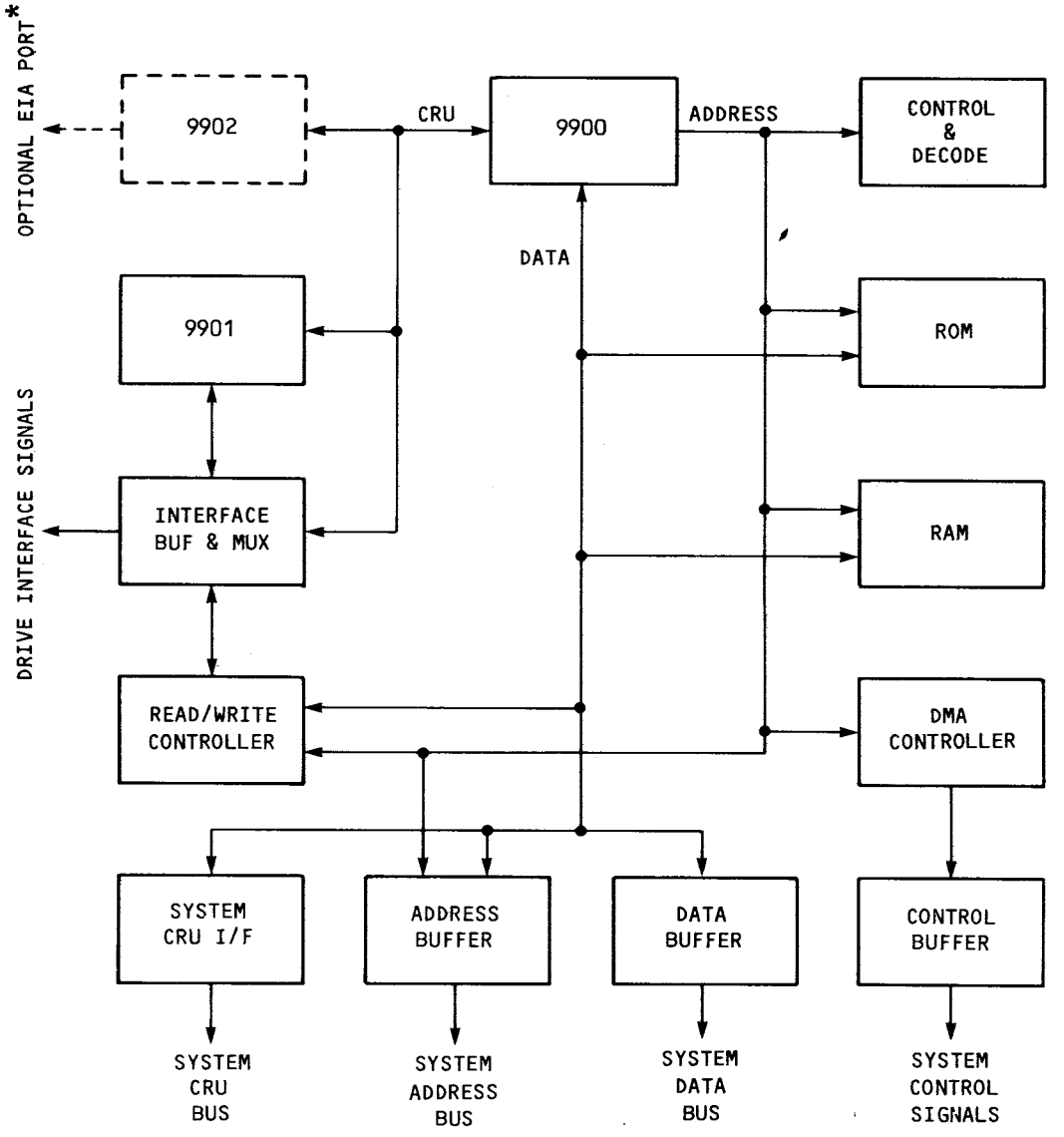


FIGURE 4-1. TYPICAL SYSTEM BLOCK DIAGRAM

4.3 CONTROLLER DESCRIPTION

The TM 990/303A disk controller consists of a local processor system (section 4.4), disk drive interface (section 4.5), host system interface (section 4.6), and read/write controller (section 4.7) as shown in Figure 4-2. The local processor system contains a TMS 9900 microprocessor, TIM 9904 clock generator, read-only memory (ROM), read-write memory (RAM), decode PROM and wait-state control logic. The disk drive interface contains a TMS 9901 programmable interface adaptor, a multiplexer for additional inputs, and disk drive interface driver and receiver circuits. The host system interface consists of a CRU interface and a DMA interface. The DMA interface contains an input data register, an output data register, a memory address register, and memory access control logic. The read/ write controller contains two ROM controllers for data bit and word processing, a data shift register, a cyclic-redundancy-check (CRC) circuit, a phase-locked loop (PLL) for data synchronization and control logic.



*Optional EIA port is used for board test and checkout. This port is not populated when shipped.

FIGURE 4-2. DISK CONTROLLER BLOCK DIAGRAM

4.4 LOCAL PROCESSOR SYSTEM

The local processor system contains a TMS 9900 microprocessor operating at 3 MHz. The clocks for the processor are provided by a TIM 9904 clock generator using a 48 MHz crystal oscillator which is divided down to give the four-phase clocks for the processor. The clock generator also provides synchronization of the processor RESET signal as well as crystal-controlled clock signals for the read/write controller.

The local processor memory consists of 2048 16-bit words of ROM (with provision for future expansion to 4096 words) and 1024 16-bit words of RAM. The local memory bus is also used for local, memory-mapped I/O. Local memory addresses are decoded by a 32 X 8 PROM and by a 3 to 8-line decoder which is gated with write enable (MWE-) to generate load signals for local RAM and data registers.

The DMA address, DMA data, command data (from host system CRU), and read/write controller data register are memory-mapped I/O devices. The local processor memory map is given in Table 4-1. Certain memory address bits are used during memory-mapped I/O to the DMA and read/write controllers in addition to the data bus. These features will be described in the appropriate sections of this manual. The DMA and read/write controllers will delay the local processor using wait states via the READY signal if they are accessed prior to data being available.

TABLE 4-1. PROCESSOR MEMORY ADDRESS MAP

Address	Read Function	Write Function
0000-1FFE	ROM	
2000-3FFE	Reserved	Reserved
4000-5FFE		DMA Address
6000-7FFE	*DMA Data	*DMA Data
8000-9FFE	*Read/Write Controller	
A000-BFFE		*Read/Write Controller
C000-DFFE	Command Data	
E000-FFFE	RAM	RAM

*Processor wait states are used for synchronization.

4.5 DISK DRIVE INTERFACE

The disk drive interface consists of a TMS 9901 programmable interface adapter, an 8-input multiplexer, a drive-select decoder, output buffers, and input receiver circuits. The TMS 9901 contains parallel I/O, interrupt, and timer sections interfaced to the local processor through the CRU. The parallel I/O port CRU bit assignments are given in Table 4-2. Fifteen ports are used as outputs and one port is used as an input. The ACCEPT, BUSY, SYSINT-, and COMMAND signals are used by the host system interface. The DDENSITY, RFMTWPCOMP, LED- and RWRST- signals are used by the read/write controller. The other I/O ports are used as outputs to the disk drive interface circuit.

The disk drive select signals, DSELECT1- through DSELECT4-, are generated by decoding the DSEL0 and DSEL1 signals from the TMS 9901. The drive select signals and all other drive control signals except HEADLOAD- and WRITEDATA- are controlled by the DRIVENABLE- output from the TMS 9901. The DRIVENABLE-

signal is pulled-up to the inactive state whenever the TMS 9901 is reset (All TMS 9901 outputs become inputs). This prevents any disk operation except headload from occurring in the event of system power-on, power-off, or reset.

TABLE 4-2. TMS 9901 PARALLEL I/O PORT BIT ASSIGNMENT

CRU Base Address (R12) is $1F60_{16}$			
BIT ADDRESS	IN/OUT	SIGNATURE	FUNCTION
0	Out	DSEL0	Disk drive unit select LSB.
1	Out	DSEL1	Disk drive unit select MSB.
2	Out	WGATE	Disk drive write gate. When active (high), WGATE enables data to be written on the disk.
3	Out	DIR	Disk drive step direction.
4	Out	SIDE	Disk drive side select.
5	Out	DDENSITY	When active (high), DDENSITY causes the data separator to interpret read data in double density mode. After changing this bit, allow at least 12 microseconds before trying a read or write operation. When inactive, DS3 is illuminated.
6	Out	RFMTWPCOMP	When active (high), RFMTWPCOMP selects the TI sync format (55_{16}) in read-mode and enables precompensation in write mode. When inactive in read mode,IBM sync format (00_{16}) is selected. When inactive, DS2 is illuminated.
7	Out	STEP	Disk drive step.
8	Out	HLOAD	Disk drive head load.
9	Out	ACCEPT	When active (high),ACCEPT indicates to the host system that a command or data byte has been read by the disk controller.
10	Out	BUSY	When active (high), BUSY indicates to the host system that the disk controller is busy and cannot accept a new command.
11	Out	RWRST-	When active (low), RWRST- clears all flip-flops, registers, and counters in the read/write controller except for the data separator and clears the read/write controller address bit functions (All inactive except RESET- and PRESETCRC-).
12	Out	DRIVENABLE-	When active (low), DRIVENABLE- enables all disk drive control signals except HEADLOAD and WRITE DATA (outputs from the controller). DRIVENABLE- is inactivated when the controller is reset.

TABLE 4-2. TMS 9901 PARALLEL I/O PORT BIT ASSIGNMENT (CONTINUED)

BIT ADDRESS	IN/OUT	SIGNATURE	FUNCTION
13	Out	SYSINT-	System interrupt. The high-to-low transition of SYSINT- sets the INT flip-flop. This will cause an interrupt to be issued to the host system if interrupts have been enabled through the CRU. When active (low), SYSINT- clears the automatic bootload flip-flop.
14	Out	LED-	When active (low), LED- turns on disk controller status indicator DS1.
15	In	CCOMMAND	When active (high), CCOMMAND indicates that the host system is sending a command byte to the disk controller. This signal can also be used as interrupt 7 by the 9901.

The TMS 9901 interrupt assignment is given in Table 4-3. These signals can be used as either interrupts or inputs as required by the firmware. The TMS 9901 timer section is used by the firmware to provide timing for various disk operations.

An 8-bit multiplexer is also interfaced to the local processor through the CRU. This multiplexer is used as the auxiliary input port P4. The auxiliary input port CRU bit assignments are given in Table 4-4. In addition to disk drive inputs, two jumpers (J8 and J9) are connected to the auxiliary input port. These jumpers are used in conjunction with the two-sided signal (TWOSD-) to define the disk drive type to be used during an automatic bootload operation. Jumper J9 is not monitored by controller firmware; however, J8 is used to determine the default value of the disk format (IBM or TI) during a bootload. The jumper configurations are given in Table 4-5.

TABLE 4-3. TMS 9901 INTERRUPT ASSIGNMENT

CRU Base Address (R12) is $1F40_{16}$.			
BIT ADDRESS	INTERRUPT LEVEL	SIGNATURE	FUNCTION
0			Control bit. Set to zero to write mask or read interrupts.
1	1	CCUE-	When active (low), CCUE- indicates that the host system is sending data or command to the disk controller.
2	2	COMINT-	Communication interrupt. When active (low) COMINT- indicates that the optional 9902 communication interface is issuing an interrupt.
3	3	ALHOLDQ-	Automatic bootload. When active (low), ALHOLDQ- indicates that the automatic bootload flip-flop is set as the result of system power-up or the power-on reset signal (PRES.B-).
4	4	OVERRUNQ-	Data overrun error. When active (low), OVERRUNQ- indicates that a data timing error has occurred during a disk read or write operation.
5	5	CRCERRORQ-	Cyclical redundancy check error. When low (active), CRCERRORQ- indicates that a data error has occurred during a disk read operation.
6	6	INDX-	Disk drive index. When active (low), INDX- indicates that the index hole in the diskette is being sensed by the disk drive.
7	7	CCOMMAND	See TMS 9901 parallel I/O port bit assignment (bit 15).

TABLE 4-4. AUXILIARY PARALLEL INPUT PORT BIT ASSIGNMENT

CRU Base Address (R12) = $1F80_{16}$		
BIT ADDRESS	SIGNATURE	FUNCTION ADDRESS
0	TWOSD-	Disk drive two sided.
1	DCHG	Disk drive disk change.
2	DRDY	Disk drive ready.
3	TZERO	Disk drive track zero.
4	WPROT	Disk drive write protect.
5	SIZE-	Disk drive size. When active (high), SIZE- indicates that the disk drive is mini-sized using 5-inch diskettes; when low, standard size is indicated. This is via jumpering J9 which is not interpreted in the present configuration.
6	FMT-	Format type. When active (high), TI format is indicated. When low, IBM format is indicated. SEL2 indicates that host system memory data is organized as 8-bit bytes, otherwise host system memory is organized as 16-bit words.
7	SPAREIN	Spare input. This bit indicates the state of pin 24 of the disk drive connector.

TABLE 4-5. BOOTLOAD FORMAT SELECTION

FORMAT SELECTED	SIGNAL			JUMPER CONNECTIONS	
	SIZE-	FMT-	TWOSD-	J9 E21 to E22	J8 E23 to E24
IBM Single	0	0	0	Install	Install
IBM Double	0	0	1	Install	Install
TI Double	0	1	X	Install	Remove
MINI Single	1	0	X	Remove	Install
MINI Double	1	1	X	Remove	Install

NOTE: Jumper J9 is not interpreted under the present configuration.

4.6 HOST SYSTEM INTERFACE

The TM 990/303A disk controller contains a host system CRU interface for initialization and a host system DMA interface for command and data transfer.

4.6.1 Host System CRU Interface

A block diagram of the host system CRU interface is shown in Figure 4-3. The host system software CRU base address is set to 0210_{16} by a 256 X 4 PROM which is installed in the disk controller prior to shipment (section 3.3 and Figure 3-3 explain the CRU software base address and the resulting 32 CRU bits used). If some other CRU base address is required by the user, the PROM can be replaced with an appropriate PROM (Refer to Appendix E for PROM coding requirements). The disk controller is assigned 32 consecutive CRU bit addresses starting at the base address. 16 bits are assigned to the general-purpose CRU interface scheme as shown in Table 4-6. The other 16 bits are reserved.

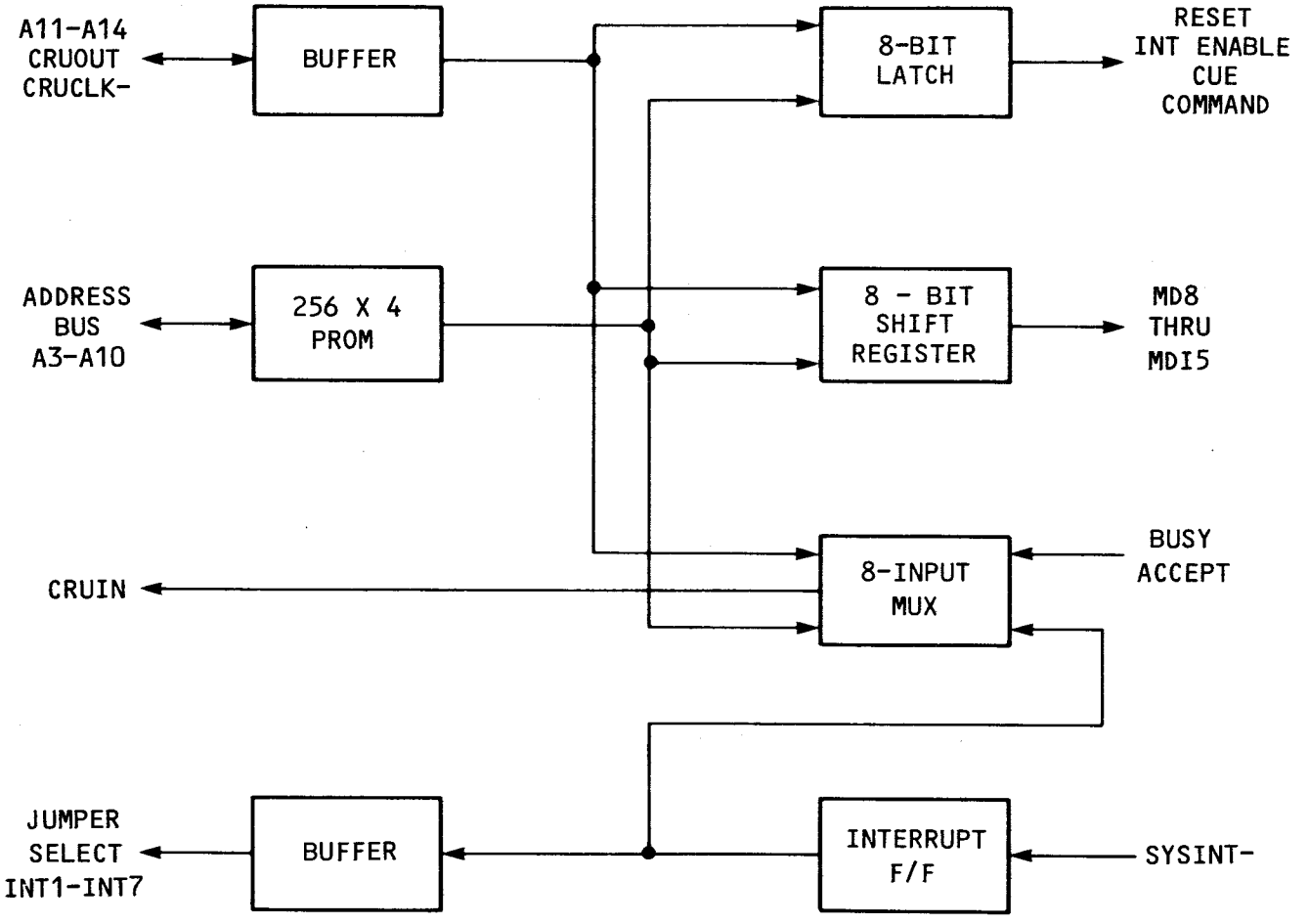


FIGURE 4-3. SYSTEM CRU INTERFACE BLOCK DIAGRAM

TABLE 4-6. GENERAL-PURPOSE CRU INTERFACE

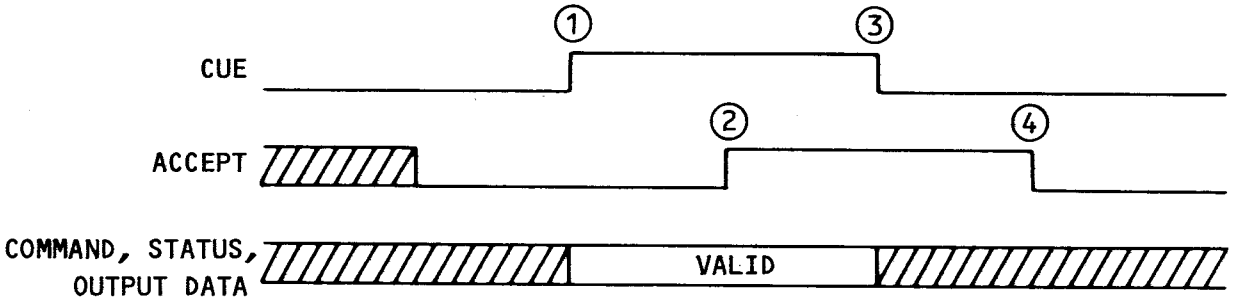
Host System CRU Base Address is 0210_{16}.		
HOST SYSTEM RELATIVE CRU BIT ADDRESS	OUTPUT FUNCTION (TO DISK CONTROLLER)	INPUT FUNCTION (FROM DISK CONTROLLER)
0_{16}	LSB	--
1_{16}		--
2_{16}		--
3_{16}	Output Data	--
4_{16}		--
5_{16}		--
6_{16}		--
7_{16}	MSB	--
8_{16}	COMMAND	--
9_{16}	--	--
A_{16}	CUE	--
B_{16}	--	ACCEPT
C_{16}	--	BUSY
D_{16}	INTERRUPT ENABLE	--
E_{16}	RESET	--
F_{16}	--	INTERRUPT ISSUED

The general-purpose CRU interface provides a means to transfer data between two systems. The disk controller implements a subset of the general-purpose CRU interface as shown in Table 4-6. Eight-bit data fields are transferred from the host system to the disk controller using the CUE and ACCEPT signals in a handshaking scheme as shown in Figure 4-4a. The handshaking procedes sequentially as shown. The CUE signal is activated by the host system when ACCEPT is inactive and after the data field and COMMAND signal are valid. The ACCEPT signal is activated by the disk controller after the data field and COMMAND signal have been read. The CUE and ACCEPT signals are then inactivated in sequence as shown.

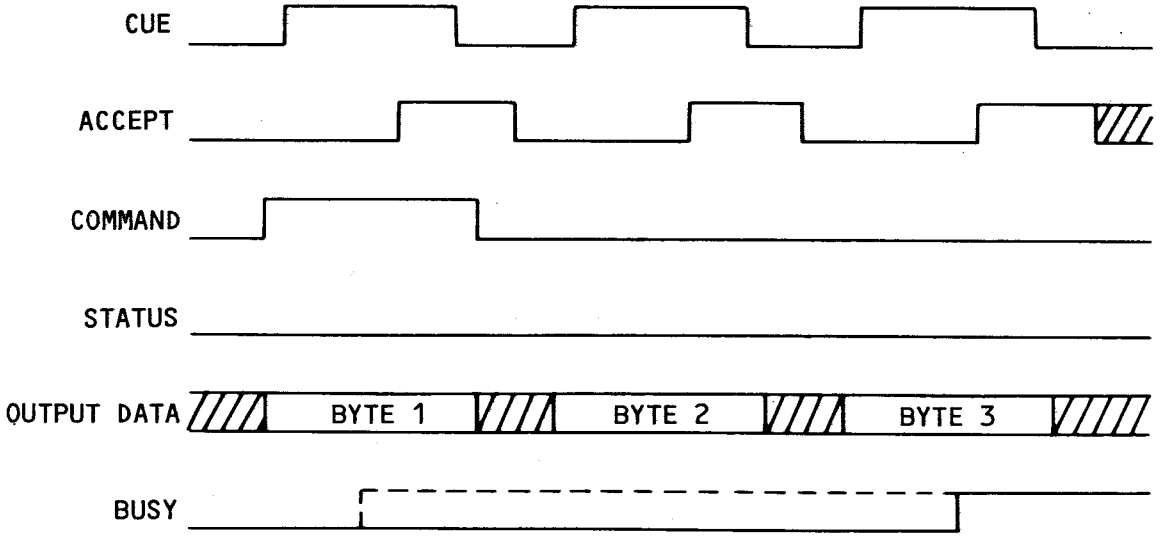
The 20-bit command list address is sent to the disk controller in three 8-bit data transfers as shown in Figure 4-4b. The COMMAND signal is activated only for the first data byte. The BUSY signal is activated by the disk controller

prior to accepting the last data byte and remains set until the controller can accept a new command.

The general-purpose CRU interface scheme allows the host system to reset the disk controller by activating the RESET signal. This causes the local processor to reset and execute its initialization firmware when the RESET signal is inactivated. The host system can control disk controller interrupts through the INTERRUPT ENABLE signal. Activating this signal allows the disk controller to interrupt the host system. Inactivating this signal resets any disk controller interrupt.



(a) Handshaking Scheme



(b) Multi-byte Command Sequence

FIGURE 4-4. GENERAL-PURPOSE CRU INTERFACE

A interrupt is reset and re-enabled by inactivating, then activating the INTERRUPT ENABLE signal. If an interrupt is being issued by the disk controller, the INTERRUPT ISSUED signal is activated. This signal can be tested by the host system to determine if the disk controller is interrupting in systems which allow other devices to share the disk controller interrrupt level. Once issued, the disk controller interrupt is latched until reset by the host system.

The general-purpose CRU interface data field is transferred from the host system CRU into an 8-bit, parallel-output shift register. This shift register is a memory-mapped I/O device on the local processor memory bus (bits 8 through 15). The shift register must be loaded by an 8-bit Load CRU (LDCR) instruction prior to setting the CUE bit.

4.6.2 Host System DMA Interface

The host system DMA interface contains two 16-bit data registers (one for input, one for output), a 20-bit address register, memory-access control logic, control signal buffers, and the automatic bootload latch. The data registers allow data to be transferred between the host system memory bus and the local processor memory bus. These registers are memory-mapped I/O devices on the local memory bus. The host system memory address is stored in a 20-bit register. This register is also a memory-mapped I/O device on the local memory bus. The address register must be loaded with a new address prior to each host system DMA cycle. The sixteen least-significant address bits are loaded from the local memory data bus; the four most-significant address bits are loaded from the local memory address bus. Thus all twenty address bits can be loaded with one firmware instruction. In addition, the data transfer direction is also defined through the local memory address bus. The DMA controller address bit utilization is given in Table 4-7.

The extended address and transfer direction are loaded into the DMA controller from the address bus at the same time the lower 16 address bits are loaded from the data bus using a base address of 4000_{16}.

TABLE 4-7. DMA CONTROLLER ADDRESS BIT UTILIZATION

ADDRESS BIT	SIGNATURE	FUNCTION
7	WRITEQ-	When active (low), this bit allows the DMA controller to control the system data bus during direct memory access and write data into memory. When inactive (high), a direct memory access will read data from system memory.
11	XA0	These bits are the four most significant bits of the 20-bit system memory address.
12	XA1	
13	XA2	
14	XA3	

The disk controller performs a single-cycle direct memory access each time the memory address register is loaded by the firmware. Loading the host system memory address sets the DMABUSYQ flip-flop. The DMABUSYQ flip-flop is reset at the end of each memory access by the high-to-low transistion of the MEMCYCQ

signal. If either of the data registers is accessed by the local processor during a DMA cycle, the local processor is held in a wait state until the DMA cycle is complete. The DMA controller memory access timing is shown in Figure 4-5. The DMA logic equations are given in Table 4-8. The subscripts used in the table refer to the respective flip-flop inputs.

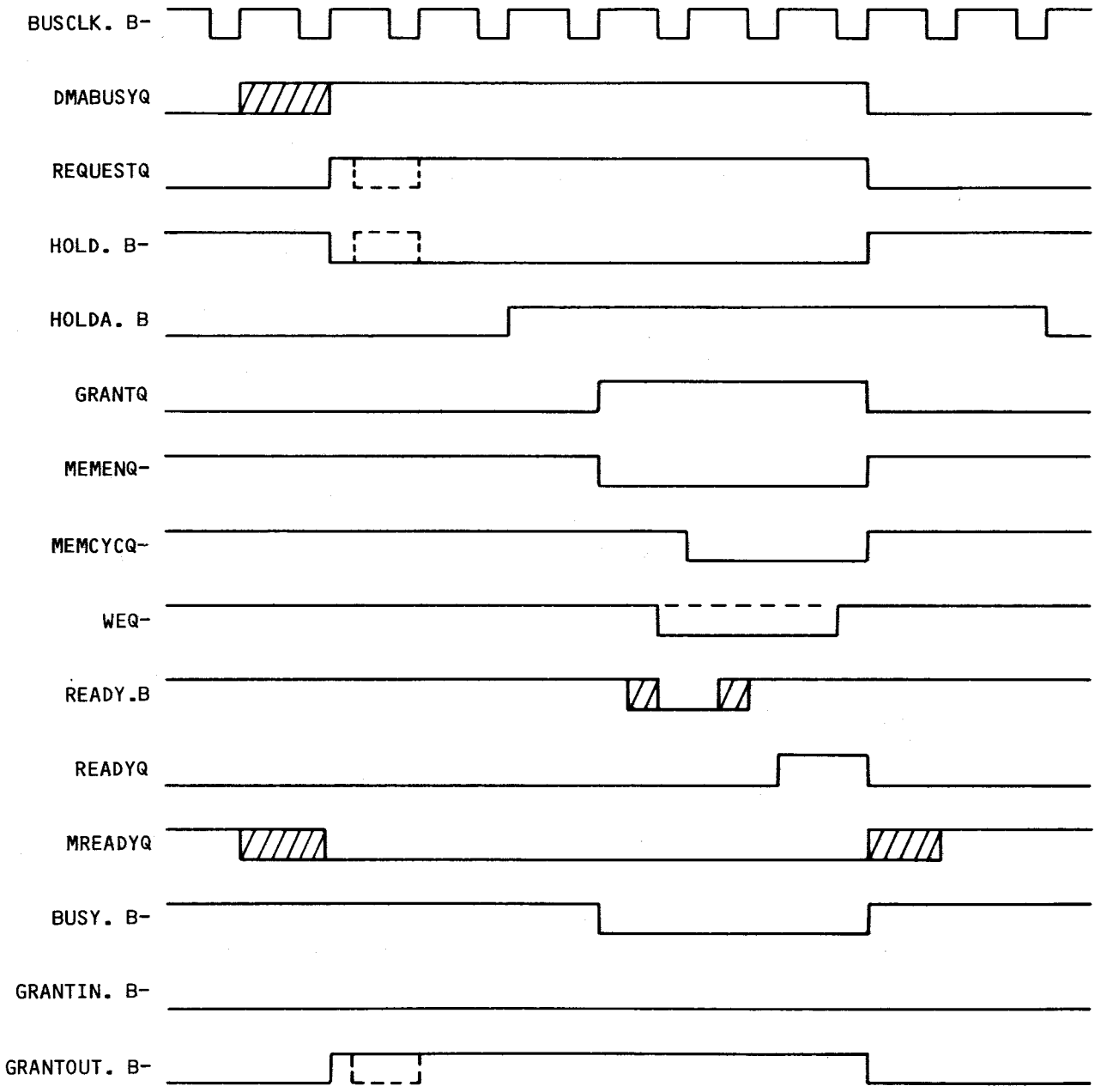


FIGURE 4-5. DMA MEMORY ACCESS TIMING (1 WAIT STATE)

TABLE 4-8. DMA CONTROLLER LOGIC EQUATIONS

$REQUESTQ_J$ = DMABUSYQ * (HOLD.B + HOLDA.B-)

$REQUESTQ_K$ = READYQ

$GRANTQ_{PRE-}$ = ALHOLDQ-

$GRANTQ_J$ = REQUESTQ * GRANTIN.B * HOLDA.B * BUSY.B-

$GRANTQ_K$ = READYQ * ALHOLDQ-

$MEMCYCQ_J$ = REQUESTQ * GRANTQ

$MEMCYCQ_K$ = READYQ

$READYQ_J$ = REQUESTQ * GRANTQ * READY.B

$READYQ_K$ = READYQ

WEQ_J = REQUESTQ * GRANTQ

WEQ_K = READYQ

WEQ_{CLR-} = WRITEQ

MEMENQ = REQUESTQ * GRANTQ

GRANTOUT.B- = (GRANTIN.B * REQUESTQ-)-

HOLD.B- = (REQUESTQ + GRANTQ)-

BUSY.B- = GRANTQ-

The DMA interface also contains the automatic bootload latch ALHOLDQ. This latch is set during system power-up by either the PRES.B- signal or by an on-board RC timing circuit. The automatic bootload can be disabled by jumpers. When the ALHOLDQ latch is set, the disk controller activates the HOLD.B- signal thus disabling the host processor. The ALHOLDQ latch remains active until reset by the firmwares activation of the SYSINT- signal. The HOLD.B- signal remains active until the end of the next DMA cycle. The automatic bootload timing is shown in Figure 4-6. The processor memory timing with wait states is shown in Figure 4-7.

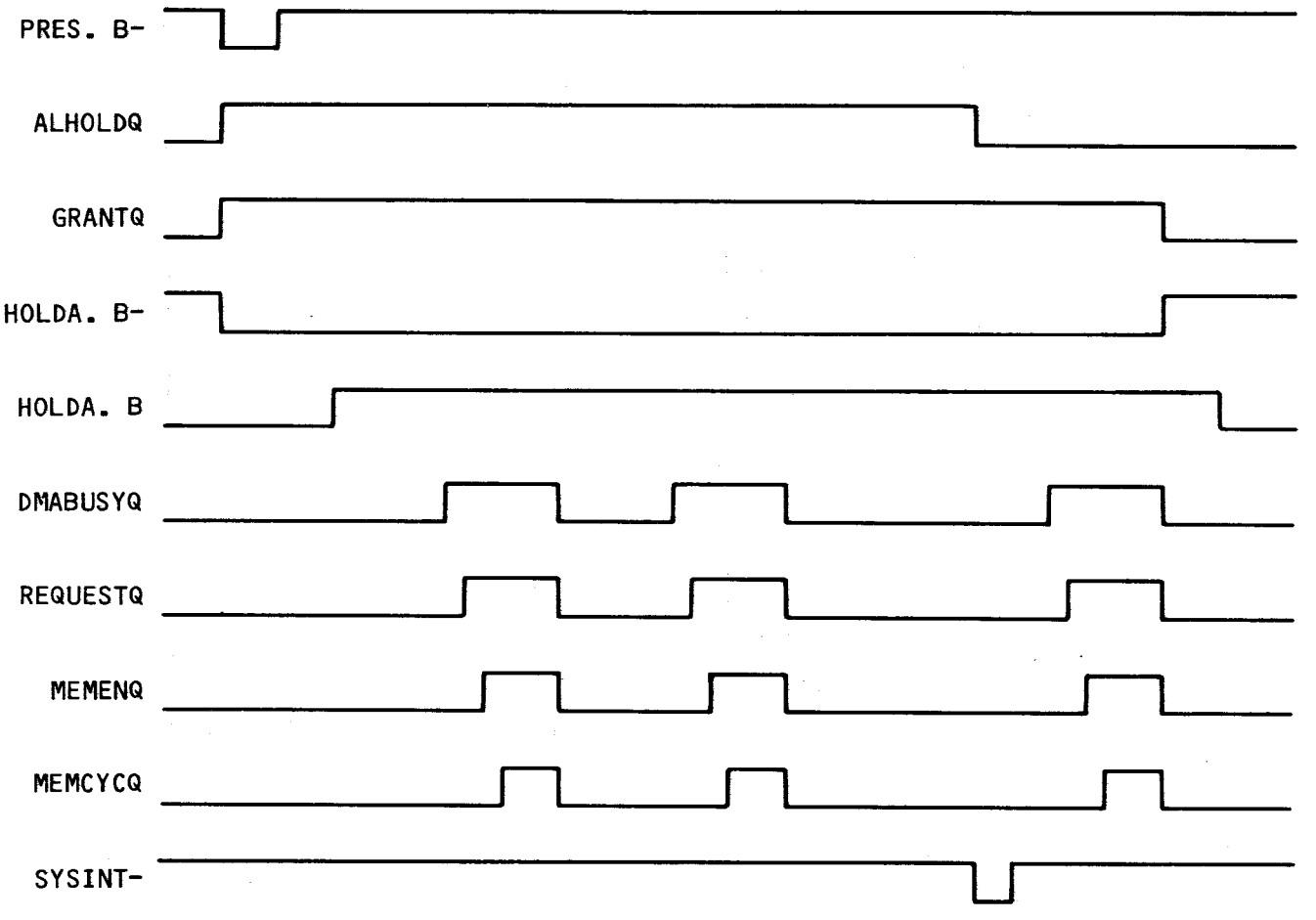


FIGURE 4-6. DMA TIMING - AUTOMATIC BOOTLOAD

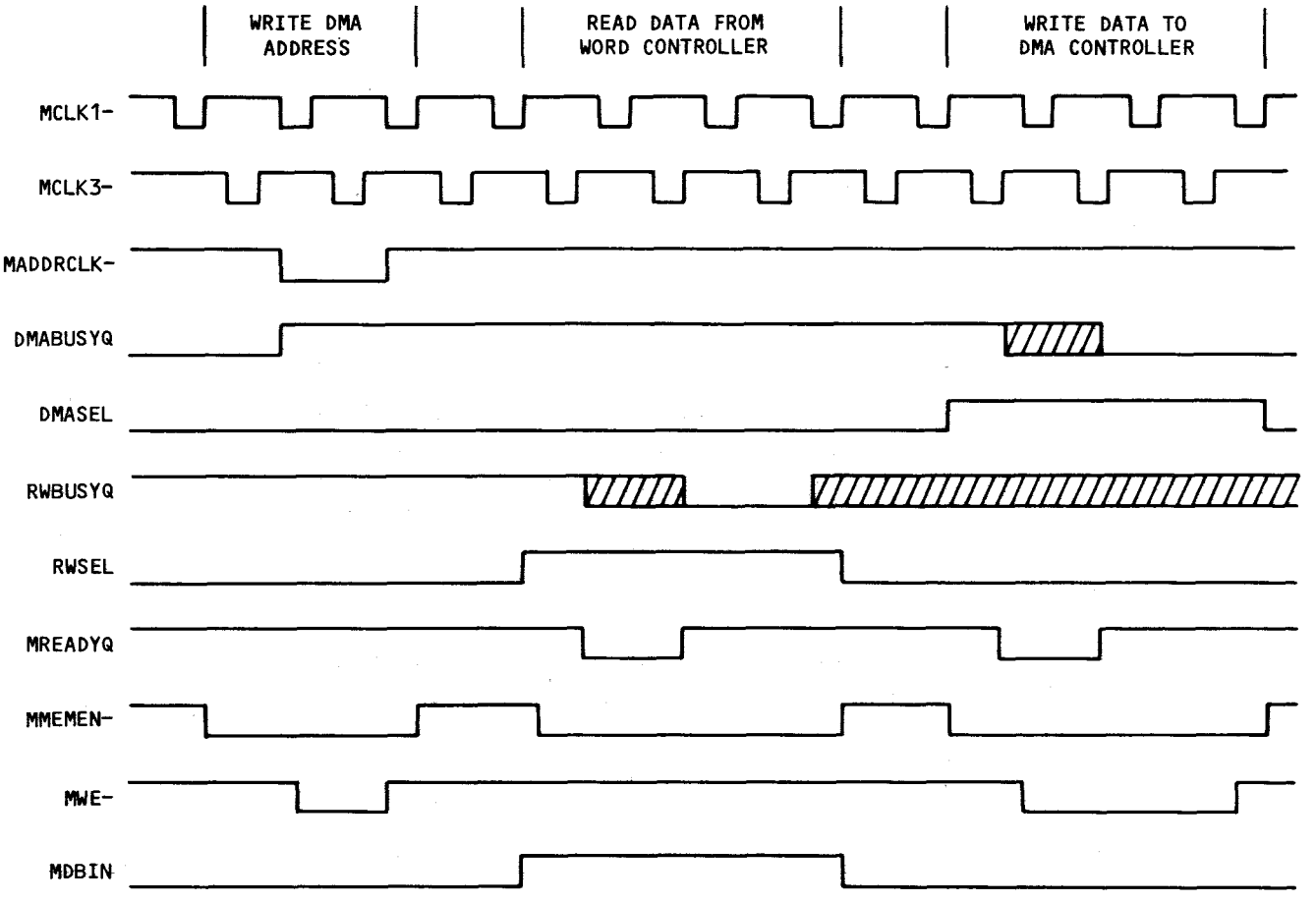


FIGURE 4-7. PROCESSOR MEMORY TIMING WITH WAIT STATES

4.7 READ/WRITE CONTROLLER

The read/write controller provides the interface between the local processor memory bus and the disk drive serial data stream. A block diagram of the read/write controller showing the major components is given in Figure 4-8. The read/write controller contains a phase-locked loop (PLL) for data synchronization, a PROM-based bit controller, a synchronization and precompensation decode PROM, a flip-flop for data and clock separation, a PROM-based word controller, a 16-bit data shift register, a cyclic-redundancy-check (CRC) function, and a control register.

The read/write controller is a memory-mapped I/O device on the local processor memory bus. The local processor memory address is used to control the operation of the read/write controller. The address bit utilization is given in Table 4-9. These bits (MA7 through MA14) are loaded into the control register during a memory write operation to the read/write controller. The control register configures the bit controller, word controller, and the data path for read or write operations.

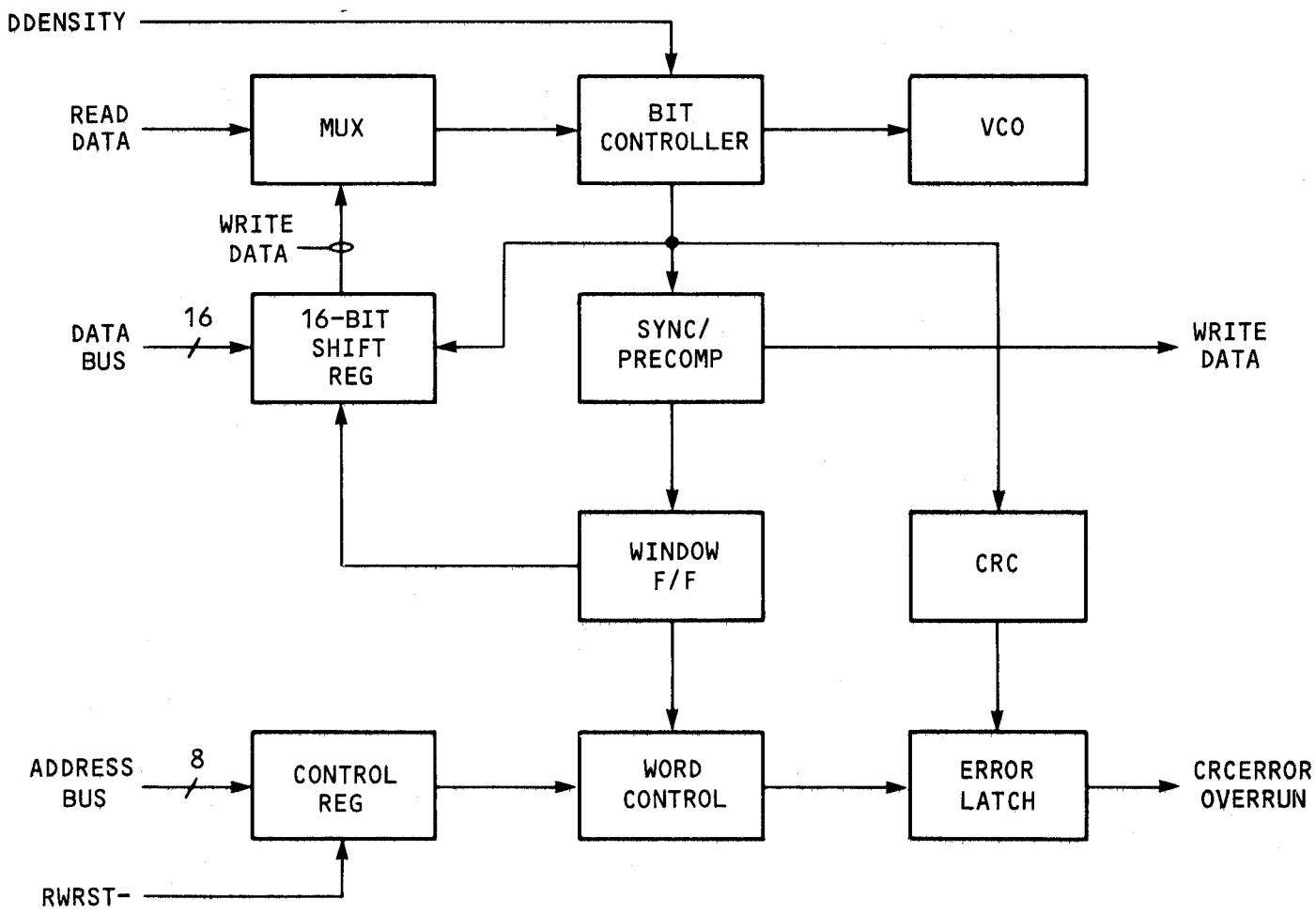


FIGURE 4-8. READ/WRITE CONTROLLER BLOCK DIAGRAM

TABLE 4-9. READ/WRITE CONTROLLER BIT UTILIZATION

Base Address When Writing To Read/Write Controller Is $A000_{16}$		
ADDRESS BIT	SIGNATURE	FUNCTION
7	RESET-	When active (low), RESET- clears OVERRUNQ and and CRCERRORQ flip-flops.
8	EXPROM	When active (high), EXPROM selects the expansion section of the data separator, sync/precomp, or read/write PROMs (jumper enabled).
9	READENABLE	When active (high), READENABLE allows the read/write controller to search for address marks and assemble data words.
10	BEREAD	When active (high), BEREAD enables the phase-lock loop to lock onto read data. When inactive the phase-lock loop locks onto the crystal oscillator.
11	READ	When active (high), READ sets the data path multiplexer and data separator to the read mode.
12	WRITEMARK	When active (high), WRITEMARK enables the read/write controller to write an 8-bit address mark using the most-significant data byte as the clock pattern (0 = delete clock, 1 = enable clock).
13	ACCUMCRC	When active (high), ACCUMCRC enables the CRC generator to accumulate the CRC check word and selects the data shift register to write data. When inactive, the CRC check word is shifted to write data.
14	PRESETCRC-	When active (low), PRESETCRC- initializes the CRC check word to all ones.

4.7.1 Read/Write Data Path

The read/write data path is shown in Figure 4-9. The data path is configured for read or write operations by the control register. Read operations use all blocks in the data path except the write data multiplexer and the precompensation shift register. Write operations use all blocks in the data path.

Write data is selected through the write data multiplexer as shown in Table 4-10. Write data is selected from either 8-bit section of the 16-bit data shift register or from the CRC generator. While an address mark is being written, the deleted clock pattern is contained in the least-significant half of the data shift register. The bit controller selects data or clock information through the write data multiplexer as required for data encoding.

The data path is configured for read or write operation by the READ multiplexer (shown in Table 4-11) and by the BCREAD multiplexer (shown in Table 4-12). The BCREAD multiplexer also controls the VCO phase detector inputs which causes the VCO to synchronize to either read data or the local processor clock.

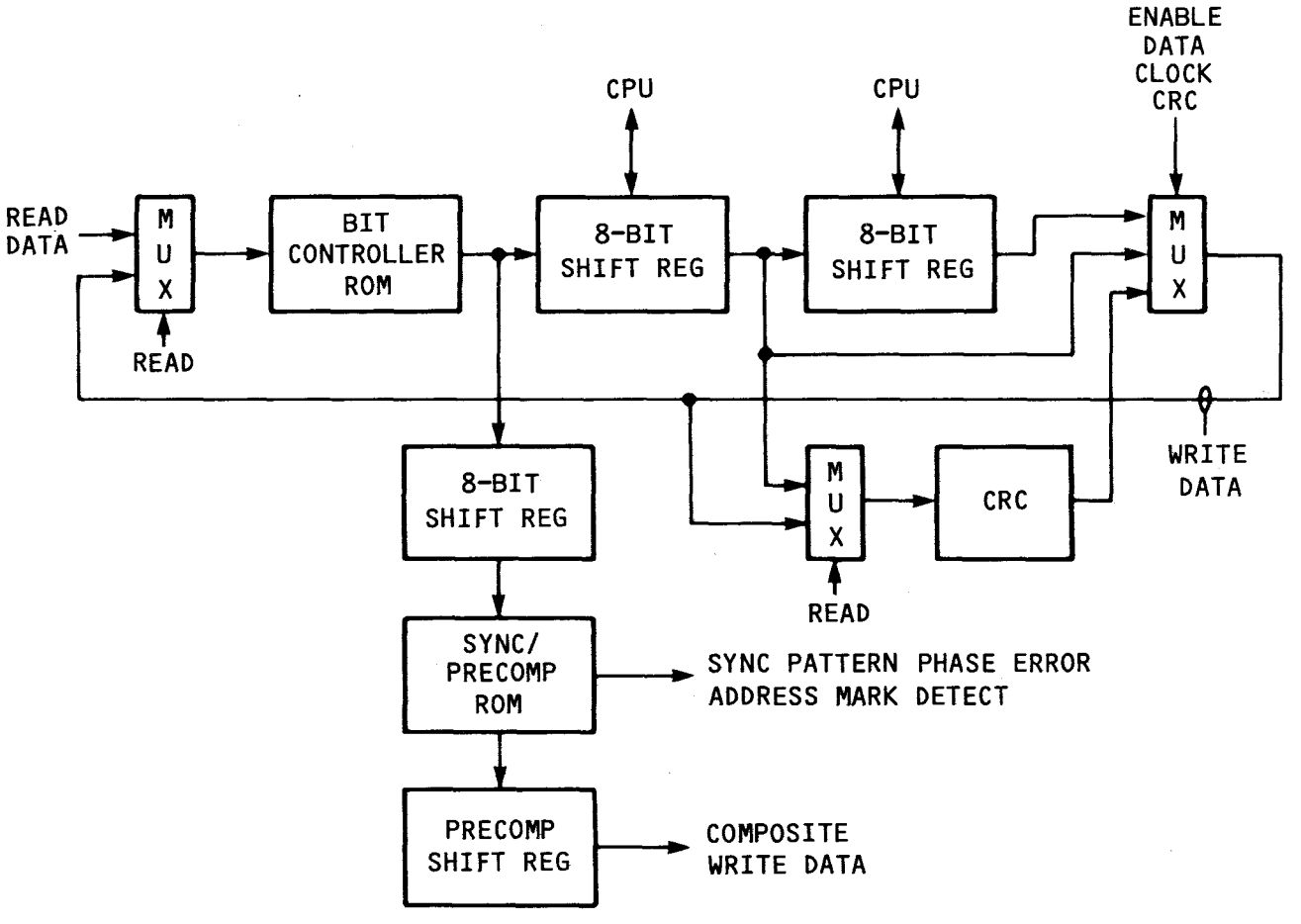


FIGURE 4-9. READ/WRITE DATA PATH

TABLE 4-10. WRITE DATA MULTIPLEXER
(74LS151)

SELECT INPUT			SIGNAL ROUTED TO OUTPUT	
UPQ(SDWINDOW) SEL C	ACCUMCRC SEL B	WRITEMARK SEL A	OUTPUT Y	OUTPUT W
0	0	0	CRCOUT	CRCOUT-
0	0	1	CRCOUT	CRCOUT-
0	1	0	SRDATAMSB	SRDATAMSB-
0	1	1	SRDATAMSB	SRDATAMSB-
1	0	0	1	0
1	0	1	SRDATALSB	SRDATALSB-
1	1	0	1	0
1	1	1	SRDATALSB	SRDATALSB-

NOTE: SELECT- is held low.

TABLE 4-11. READ MULTIPLEXER

OUTPUT SIGNAL	SIGNAL SELECTED WHEN READ = 0	SIGNAL SELECTED WHEN READ = 1
SETCRC-	PRESETCRC-	RW3Q(READSYNCQ-)
WINDOWJ	UPQ(SDWINDOW)	PHASEGOOD
DATAQ-	WRITE DATA MUX FALSE OUTPUT	RDATAQQ
CRCIN	WRITE DATA MUX TRUE OUTPUT	SRDATALSB

TABLE 4-12. BCREAD MULTIPLEXER

OUTPUT SIGNAL	SIGNAL SELECTED WHEN BCREAD = 0	SIGNAL SELECTED WHEN BCREAD = 1
UPCLK	XTAL6MHZ	UPQ(SDWINDOW)
DNCLK	VCO6MHZ	DNQ(PCLOADQ)
CLOCKSHIFT	Logic ONE	CLOCKWINDOW
BCTRIN	MA12	RW1Q(RBCTRINQ)

4.7.2 Bit Controller

The bit controller, shown in Figure 4-10, provides read data bit synchronization and write data encoding. The bit controller provides inputs to the PLL phase detector during reading and provides a synchronized data and clock stream BC5Q(SDATAQ) to the data path. Data and clock information is not separated by the bit controller. A half-bit-cell clock BC4Q(HBCLK) is provided to the window flip-flop which performs the data separation function. During writing, the bit controller encodes the write data into a data and clock stream BC5Q(SDATAQ), sets the window flip-flop to the correct state using BC4Q(HBCLKQ) and UPQ(SDWINDOW), and provides a load signal DNQ(PCLOADQ) for the precompensation shift register.

The phase-locked loop (PLL), shown in Figure 4-11, consists of a phase detector, low-pass filter, voltage-controlled oscillator (VCO), a clock rate multiplier, the bit controller, and part of the BCREAD multiplexer. During write operations, the PLL synchronizes the VCO to a 6-MHz clock from the local processor. During read operations, the bit controller synchronizes the VCO to the incoming data stream producing a nominal 6-MHz clock which tracks the read data.

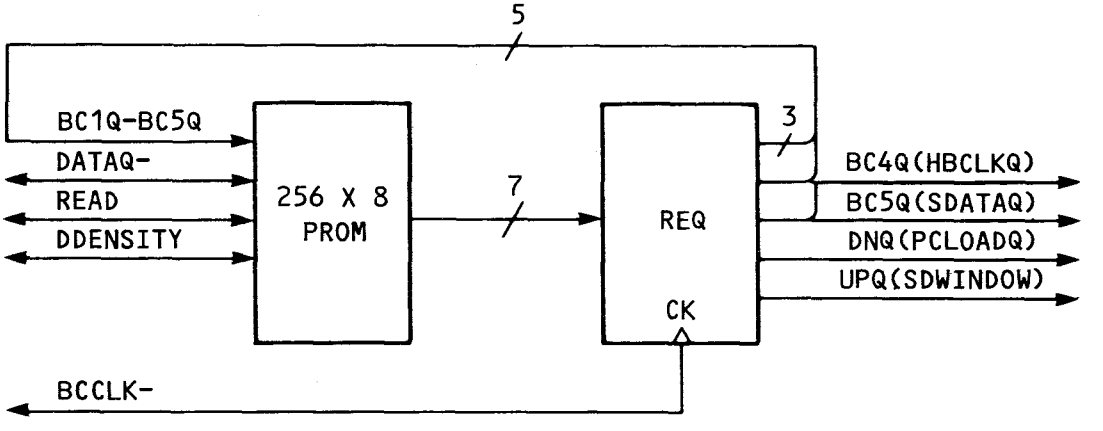


FIGURE 4-10. BIT CONTROLLER BLOCK DIAGRAM

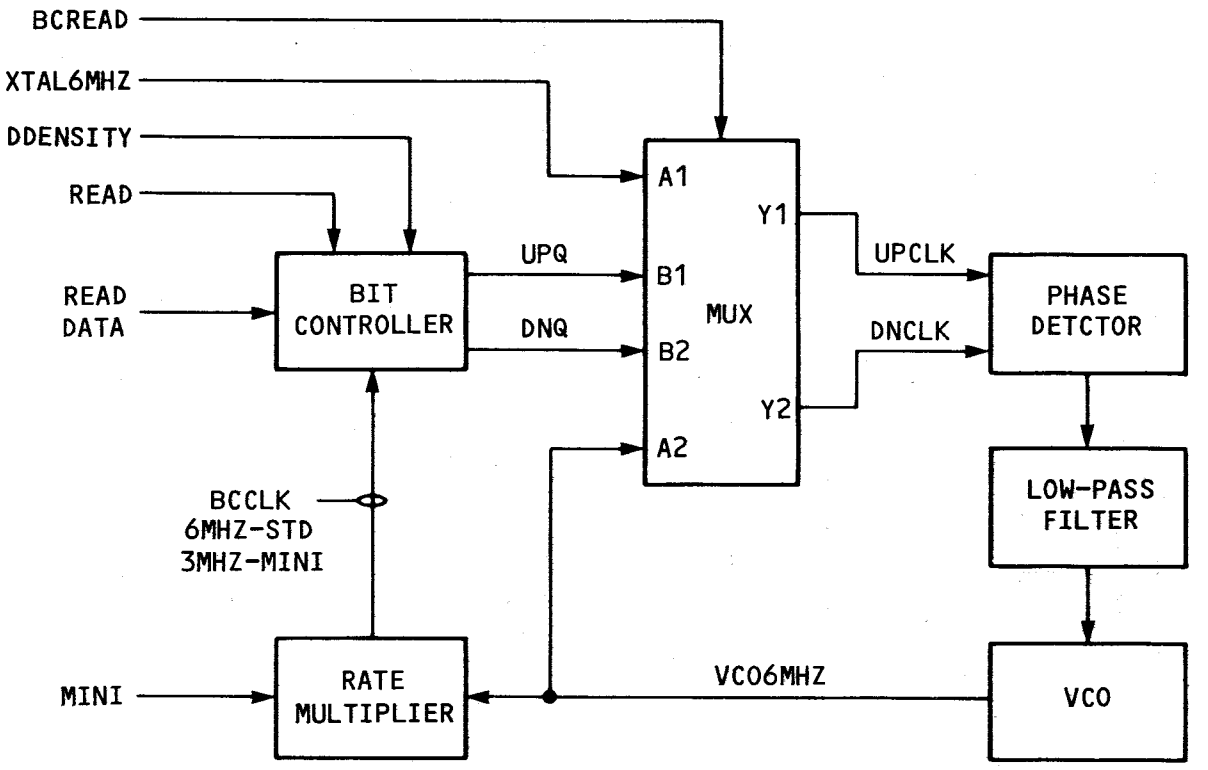


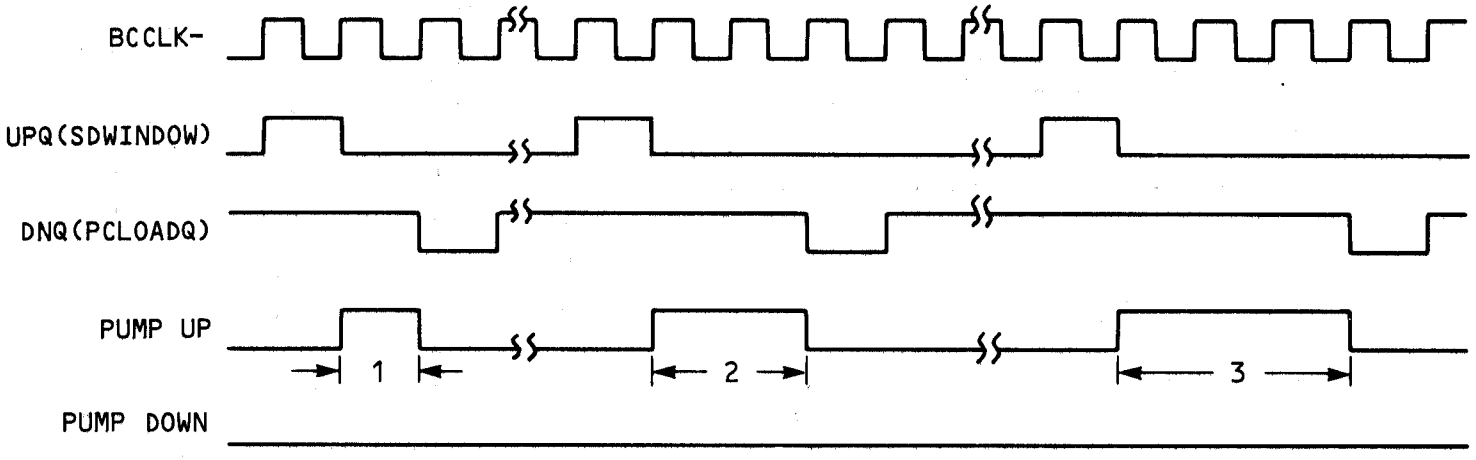
FIGURE 4-11. PHASE-LOCKED LOOP BLOCK DIAGRAM

The VCO output is used directly (6 MHz) for standard-size diskettes or divided by two (3 MHz) for mini-sized diskettes by the rate multiplier. The read/write controller operates synchronous to the VCO output clock.

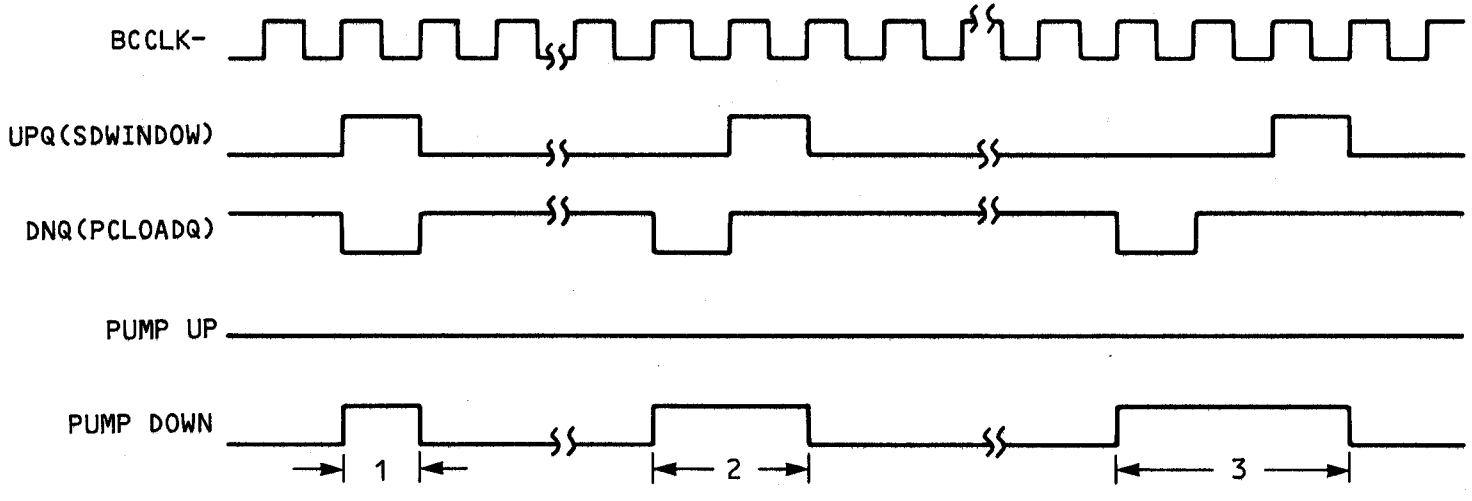
Typical phase detector timing for double-density read operations is shown in Figure 4-12. The phase detector inputs, UPQ(SDWINDOW) and DNQ(PCLOADQ) are used to cause phase corrections of 1, 2, or 3 bit controller clock periods. Note that the phase detector has negative-edge triggered inputs.

The bit controller state diagrams for writing single density (FM) and double density (MFM) are shown in Figures 4-13 and 4-15. The state transitions occur vertically from top to bottom of the diagram except as shown. The bit controller timing diagrams for writing FM and MFM are shown in Figures 4-14 and 4-16.

The bit controller state diagrams for reading FM and MFM are shown in Figures 4-17 and 4-18. In the absense of a data pulse from the disk drive, the state transitions occur vertically from top to bottom of the figure. If a data pulse occurs, the horizontal state transition path (if shown) is taken and the indicated phase correction is made (E^1, E^2, etc.) using UPQ(SDWINDOW) and DNQ(PCLOADQ)



(a) Data Occurs Early In Window



(b) Data Occurs Late In Window

FIGURE 4-12. DOUBLE DENSITY PHASE DETECTOR TIMING

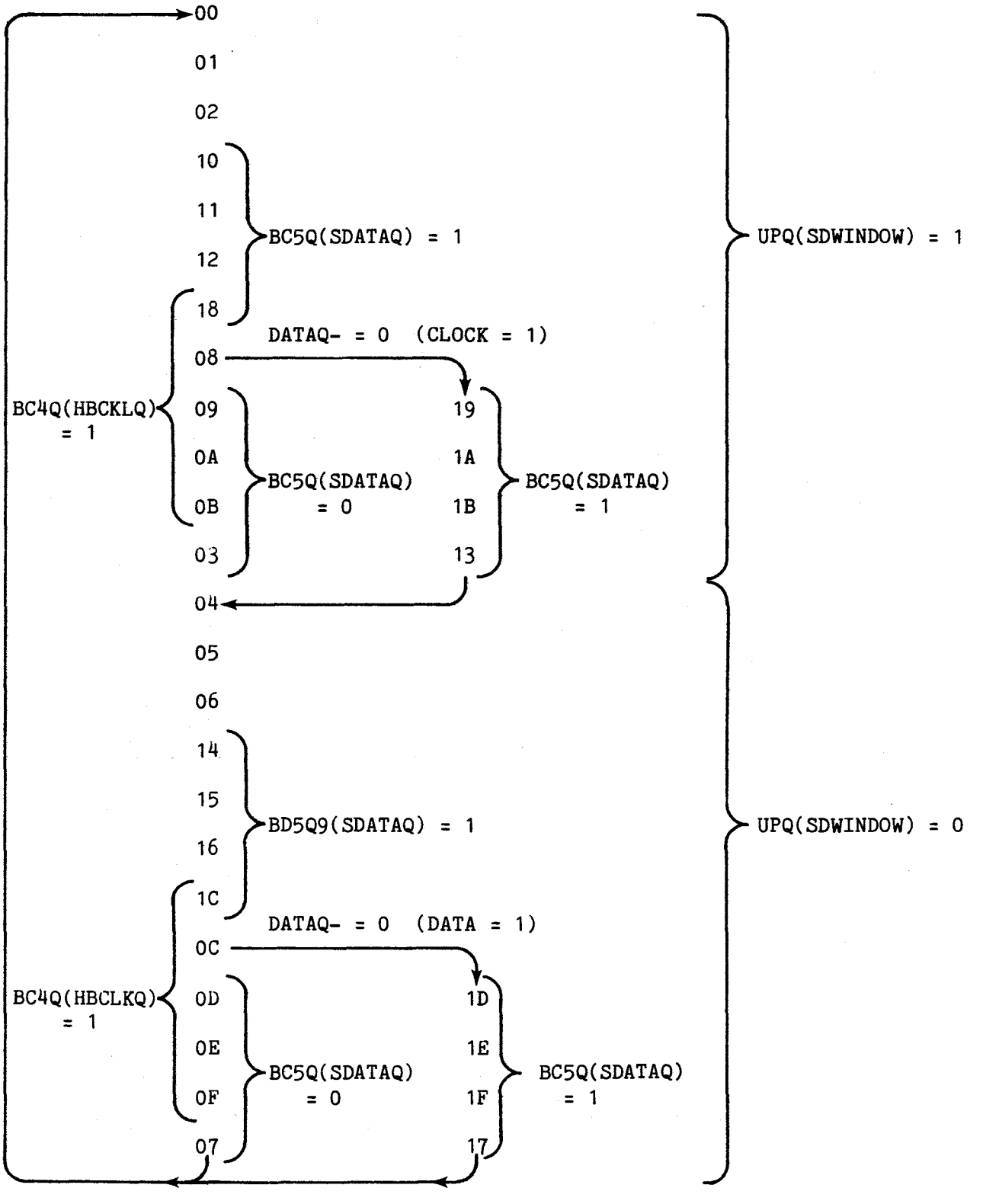


FIGURE 4-13. BIT CONTROLLER WRITE FM STATE DIAGRAM

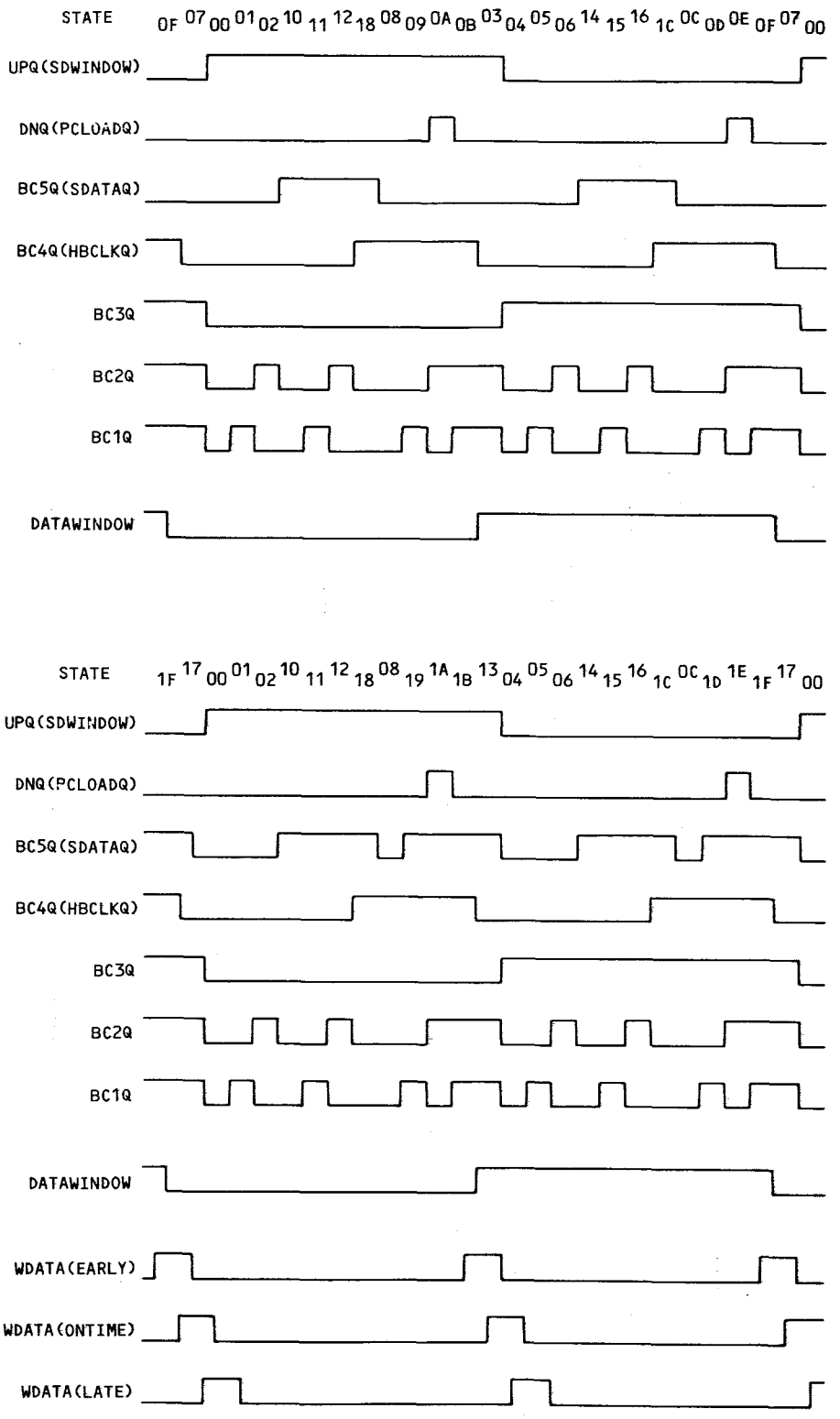


FIGURE 4-14. WRITE FM TIMING

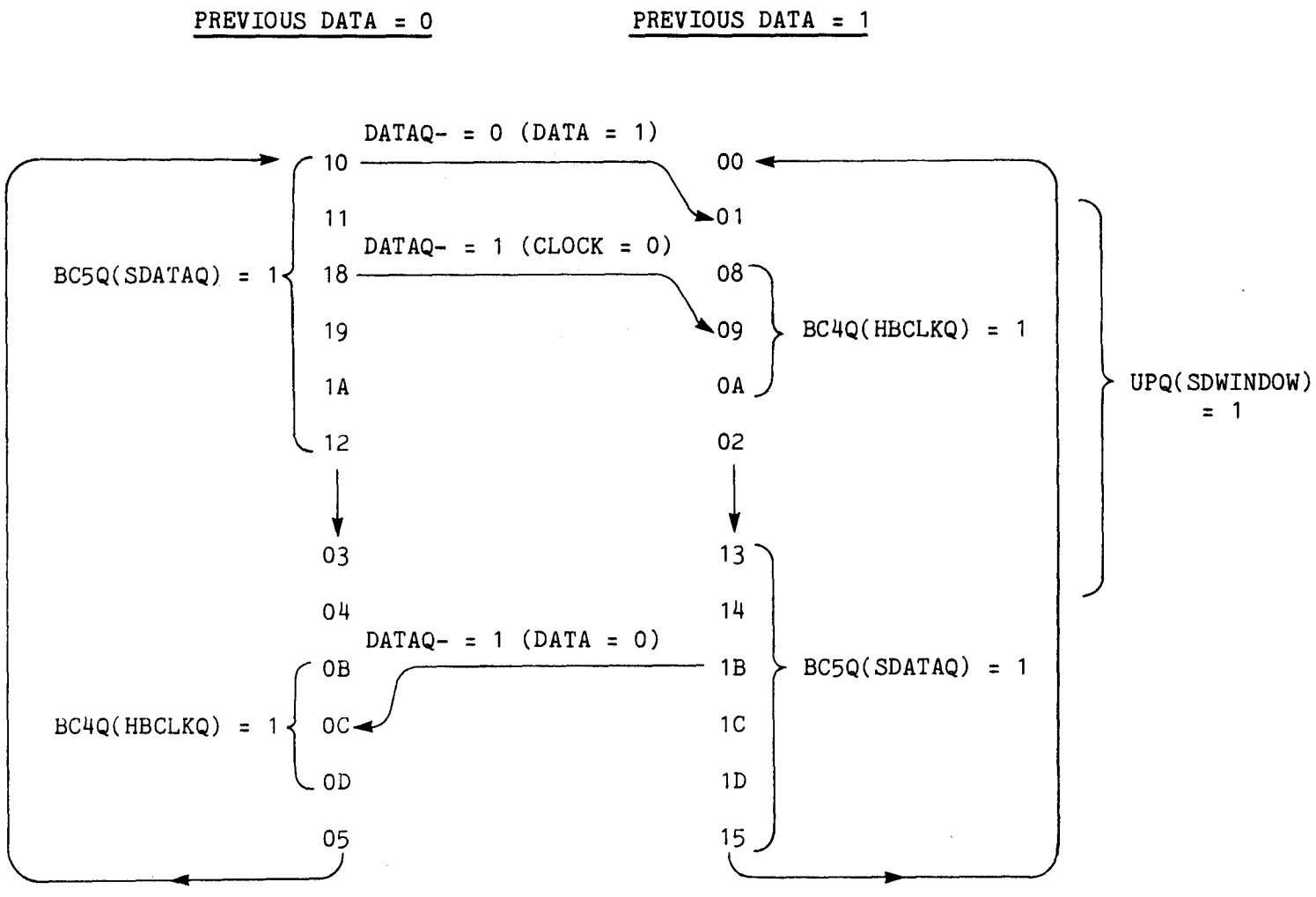


FIGURE 4-15. BIT CONTROLLER WRITE MFM STATE DIAGRAM

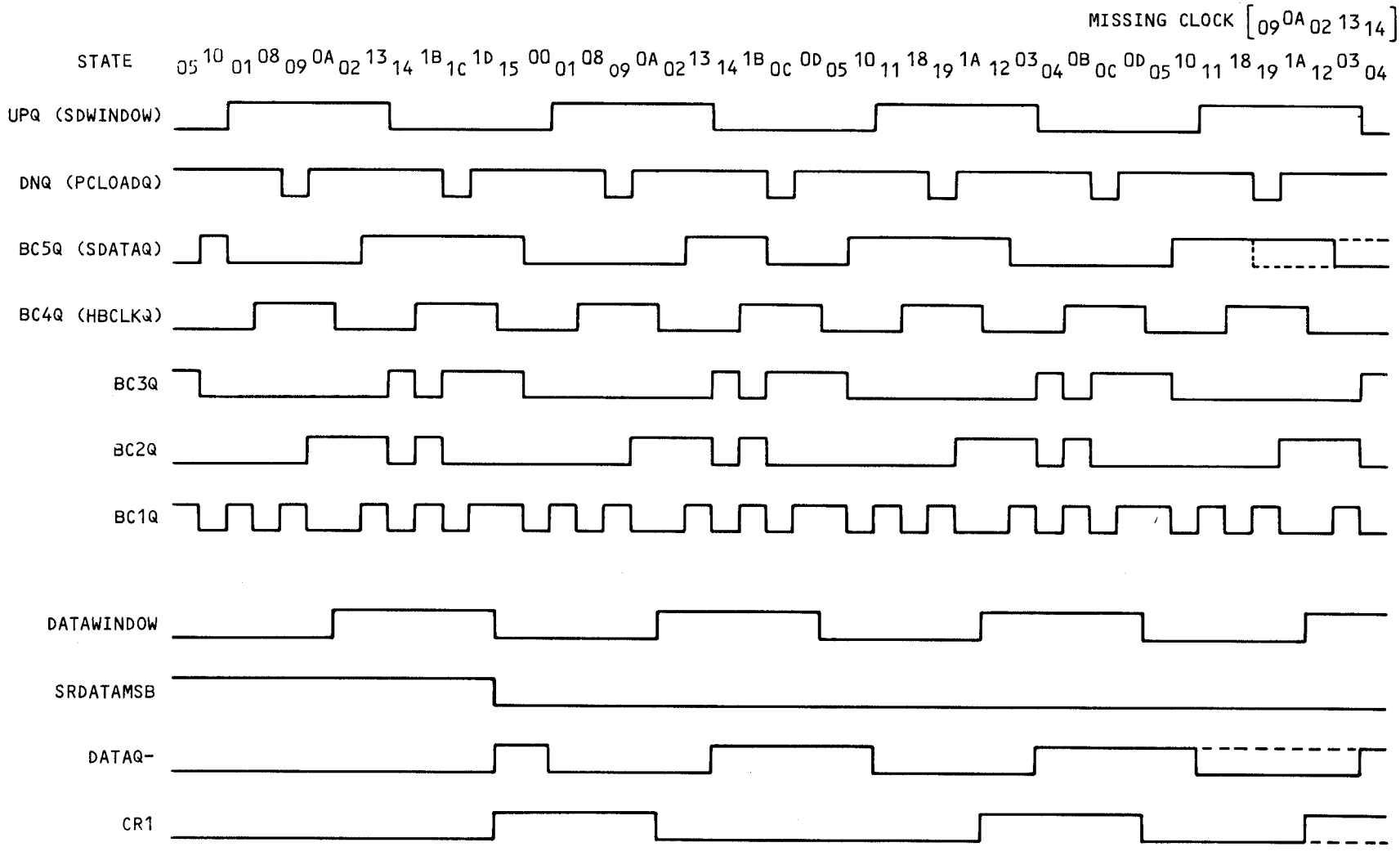
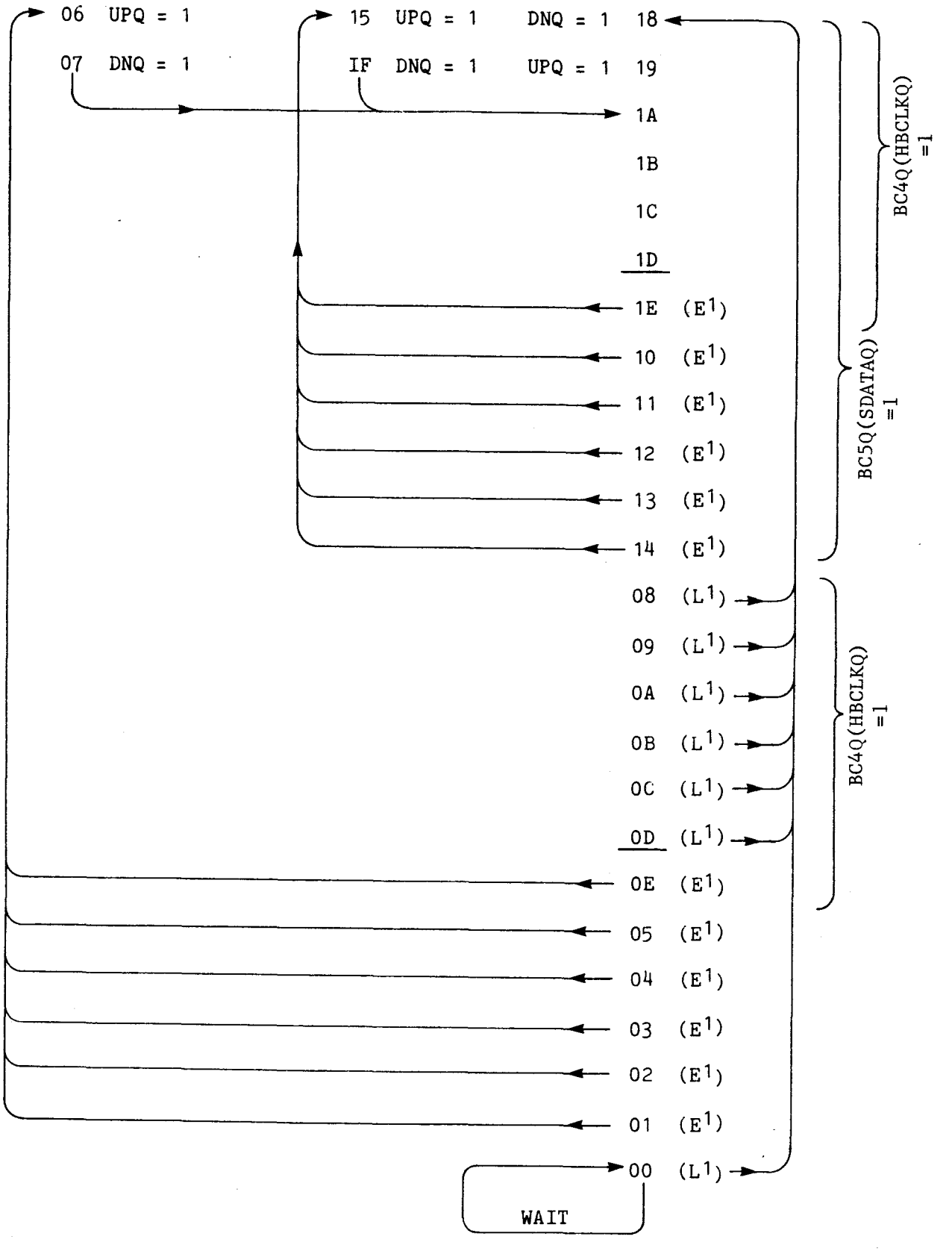


FIGURE 4-16. WRITE MFM



Unused States: 0F, 16, 17

FIGURE 4-17. BIT CONTROLLER READ FM

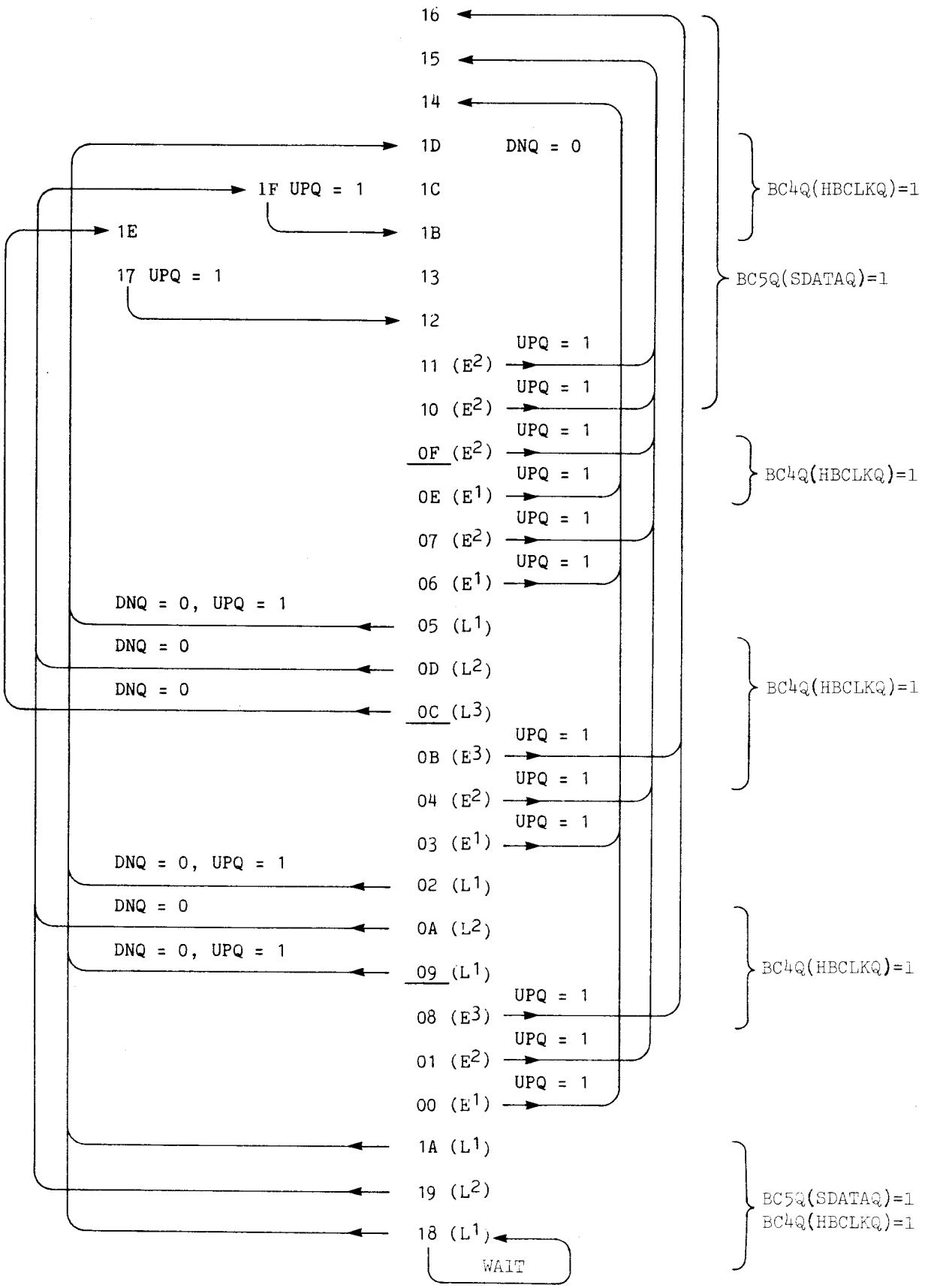


FIGURE 4-18. READ MFM

4.7.3 Synchronization and Address Mark Detection

Synchronization and address mark patterns are recorded on the disk preceding ID and data fields. Synchronization patterns allow the phase-locked loop to lock into sync with the incoming data and provide a data and clock reference for bit synchronization. Address marks provide a reference for byte synchronization. Address marks are unique bit patterns which violate the encoding rules by deleting clock bits. The synchronization and address mark patterns for various recording formats are given in Table 4-13.

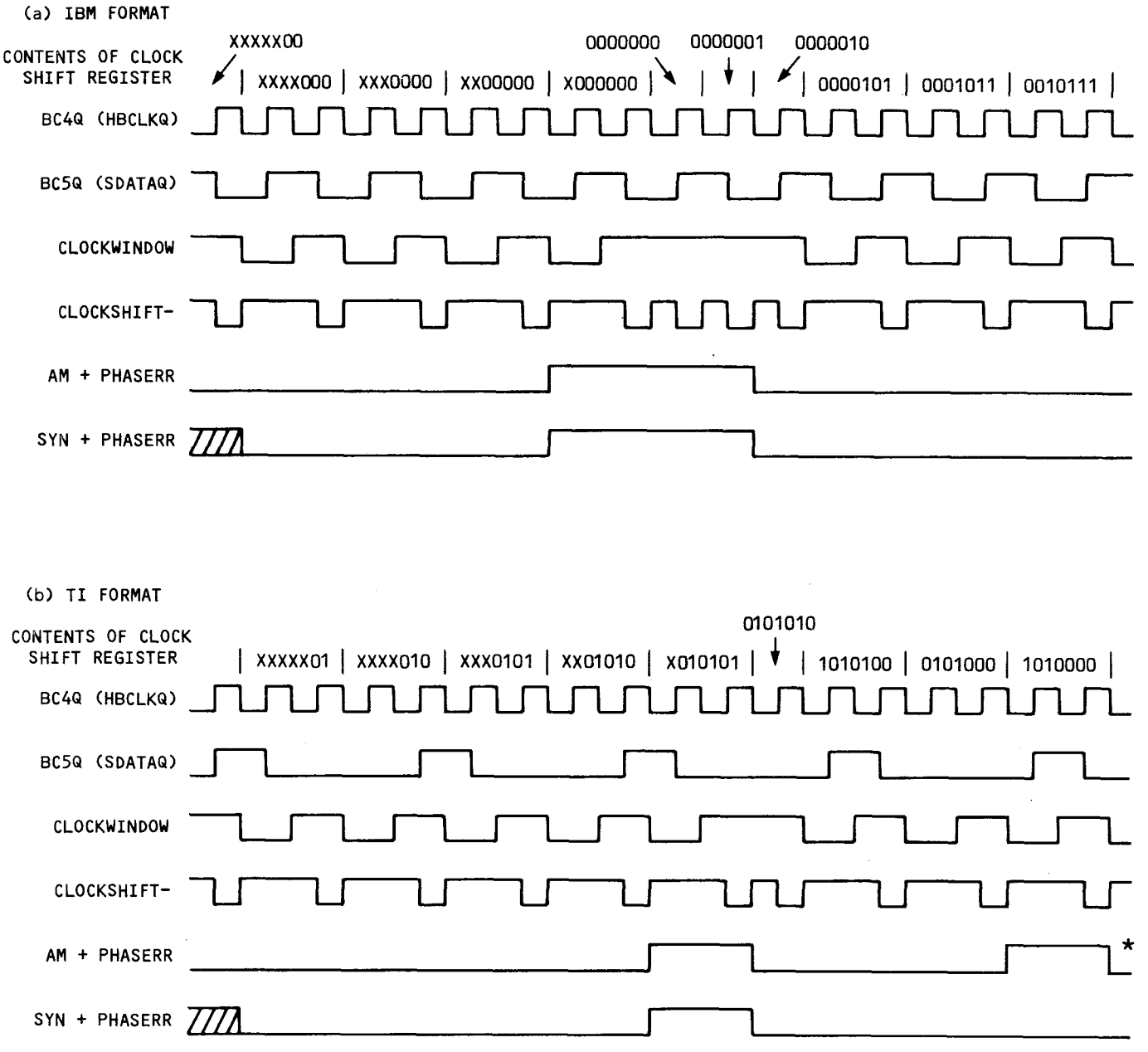
The synchronization and address mark patterns are detected by the sync/precompensation PROM. The incoming clock bit pattern is shifted into an 8-bit shift register and decoded by the PROM. If a phase error condition exists during a synchronization pattern, the window flip-flop is held in the clock window state until the phase is correct. This process is shown in Figure 4-19. The sync detect, address mark detect, and phase error conditions are encoded into the AM+PHASERR and SYN+PHASERR signals for use by the word controller.

TABLE 4-13. SYNCHRONIZATION AND ADDRESS MARK PATTERNS

PATTERN	IBM-FM FORMAT	IBM-MFM FORMAT	TI-MFM FORMAT
Sync Data	00000000	00000000	01010101
Sync Clock	11111111	11111111	00000000
Track AM Data	11111100	11000010	
Track AM Clock	11010111	00010100	
ID AM Data	11111110	10100001	00001010
ID AM Clock	11000111	00001010	01010000
Data AM Data	11111011	10100001	00001011
Data AM Clock	11000111	00001010	01010000
Delete AM Data	11111000	10100001	
Delete AM Clock	11000111	00001010	
Synchronization Phase Error Clock Pattern	XXX00000	XXX00000	XX010101

NOTE

IBM-MFM address marks are repeated 3 times and followed by the appropriate IBM-FM address mark data pattern and normal clock pattern.



*NOTE:
False address mark detection will be ignored by read/write controller.

FIGURE 4-19. PHASE ERROR RECOVERY TIMING DIAGRAM

4.7.4 Precompensation

Data precompensation is a method of compensating for bit shift caused by the magnetic media. The write data is shifted in the direction opposite of the expected shift due to the media prior to recording the data. The precompensation process is performed by two 8-bit shift registers and a PROM (a 74LS166 at U63) which is also used for synchronization and address mark detection.

Data and clock bits are shifted into an 8-bit shift register. The shift register parallel outputs are encoded by the sync/precompensation PROM. The precompensation patterns are given in Table 4-14. The PROM outputs are loaded into another 8-bit shift register and shifted by a 6 MHz clock to produce a 167-nanosecond precompensation. The output shift register timing is shown in Figure 4-20.

TABLE 4-14. ROM-GENERATED PRECOMPENSATION PATTERNS

ROM INPUT SIGNAL								ROM OUTPUT SIGNAL					COMMENT
RFMTWPCOMP	CR1	CR2	CR3	CR4	CR5	CR6	CR7	PCE	PCE0	PC0	PCL0	PCL	
1	X	X	0	0	1	0	0	0	1	1	1	0	On-Time
1	X	X	0	0	1	0	1	0	0	1	1	1	Late
1	X	X	0	0	1	1	0	0	0	1	1	1	Late*
1	X	X	0	0	1	1	1	0	0	1	1	1	Late*
1	X	X	0	1	1	0	0	1	1	1	0	0	Early*
1	X	X	0	1	1	0	1	1	1	1	0	0	Early*
1	X	X	0	1	1	1	0	0	1	1	1	0	On-Time*
1	X	X	0	1	1	1	1	0	1	1	1	0	On-Time*
1	X	X	1	0	1	0	0	1	1	1	0	0	Early
1	X	X	1	0	1	0	1	0	1	1	1	0	On-Time
1	X	X	1	0	1	1	0	0	0	1	1	1	Late*
1	X	X	1	0	1	1	1	0	0	1	1	1	Late*
1	X	X	1	1	1	0	0	1	1	1	0	0	Early*
1	X	X	1	1	1	0	1	1	1	1	0	0	Early*
1	X	X	1	1	1	1	0	0	1	1	1	0	On-Time*
1	X	X	1	1	1	1	1	0	1	1	1	0	On-Time*
X	X	X	X	X	0	X	X	0	0	0	0	0	No Data
0	X	X	X	X	1	X	X	0	1	1	1	0	Disabled

NOTES

1. X is a don't care condition.

2. Patterns marked with an asterisk (*) do not occur for double-density (MFM) encoding.

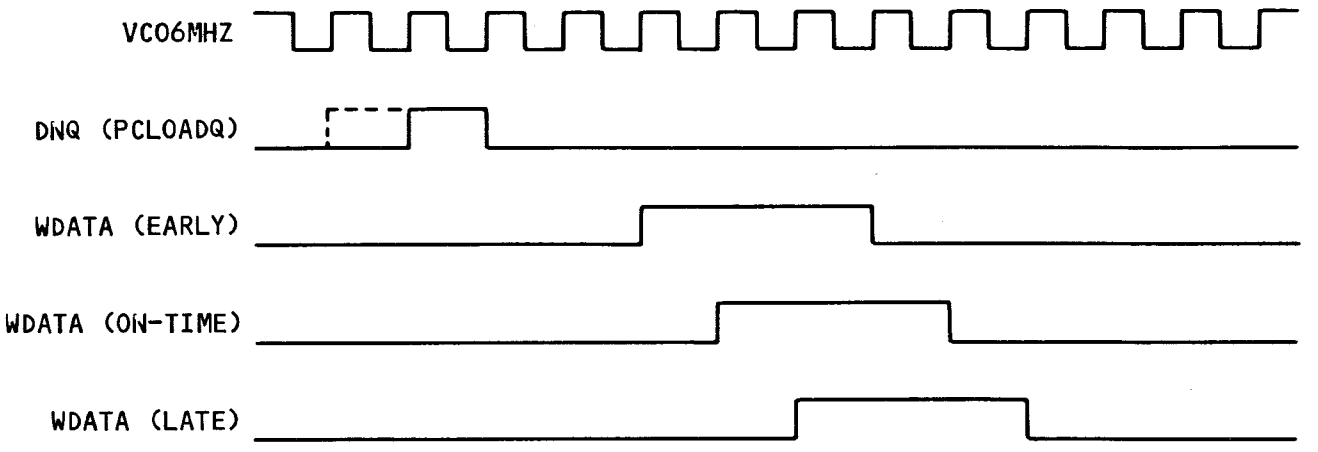


FIGURE 4-20. PRECOMPENSATION SHIFT REGISTER TIMING

4.7.5 Word Controller

The word controller, shown in Figure 4-21, provides data synchronization on a 16-bit word basis. During reading, the word controller operates as a PROM-based controller to check for a synchronization pattern followed by an address mark before establishing word synchronization. The word controller read mode flowchart is given in Figure 4-22. If a phase error is encountered prior to word synchronization, the word controller corrects the window flip-flop using the PHASEGOOD signal. During writing, the word controller operates as a logic array. The word controller logic equations are given in Table 4-15.

The word controller provides the interface between the read/write controller and the local processor. When the read/write controller is accessed as a memory-mapped I/O device, the word controller causes the local processor to enter a wait state until the data transfer occurs by using the RWBUSYQ signal. If the processor is not accessing the read/write controller at the time when a data transfer must occur, the OVERRUN latch is set.

The word controller utilizes a 4-bit counter to count data bits. During data transfer, the counter is cleared and sixteen data bits are counted. During address mark transfer, the counter is set to eight and eight data bits are counted. When the counter carry occurs, data is transferred between the data shift register and the local processor. The word controller timing is shown in Figures 4-23, 4-24, and 4-25.

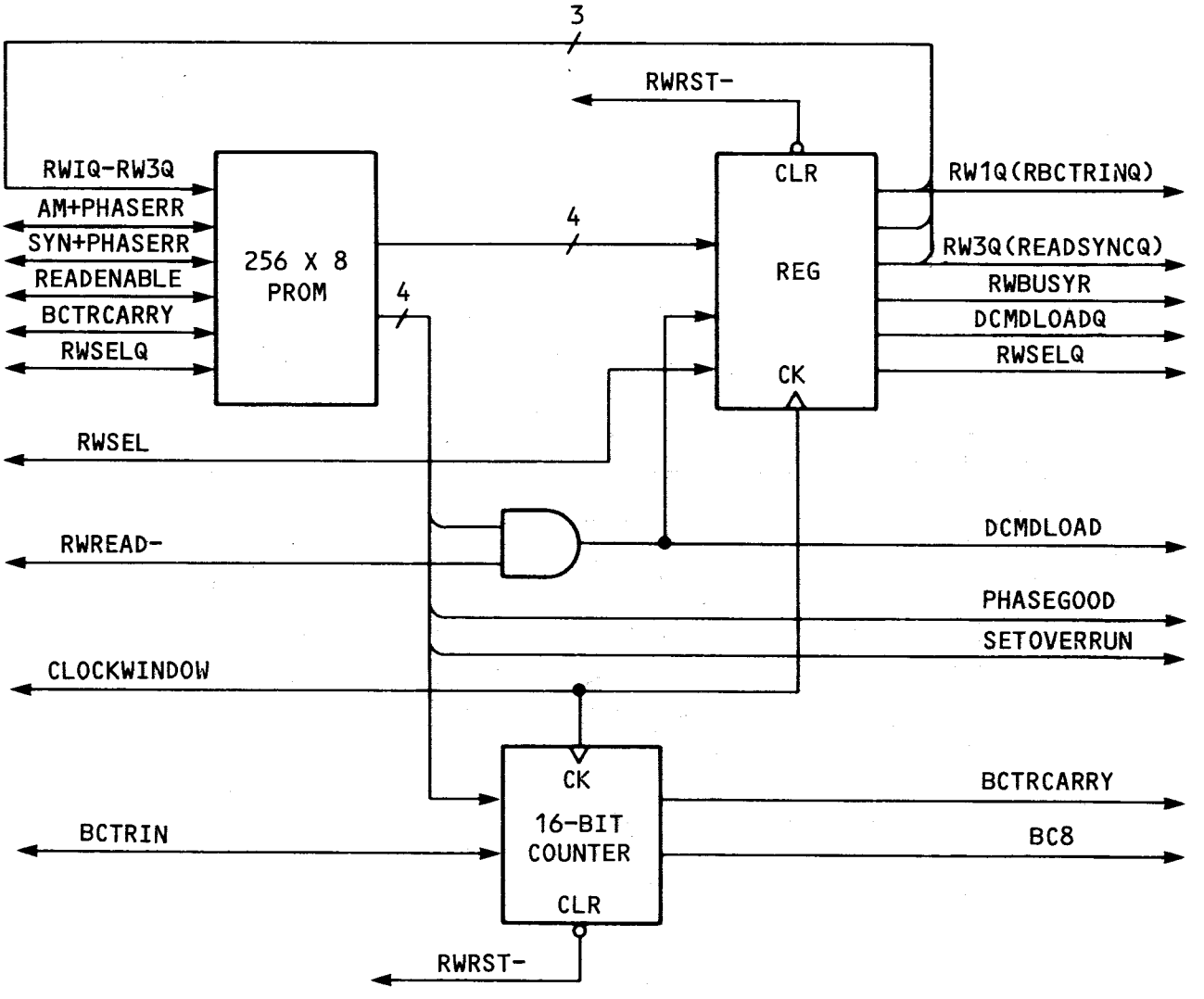
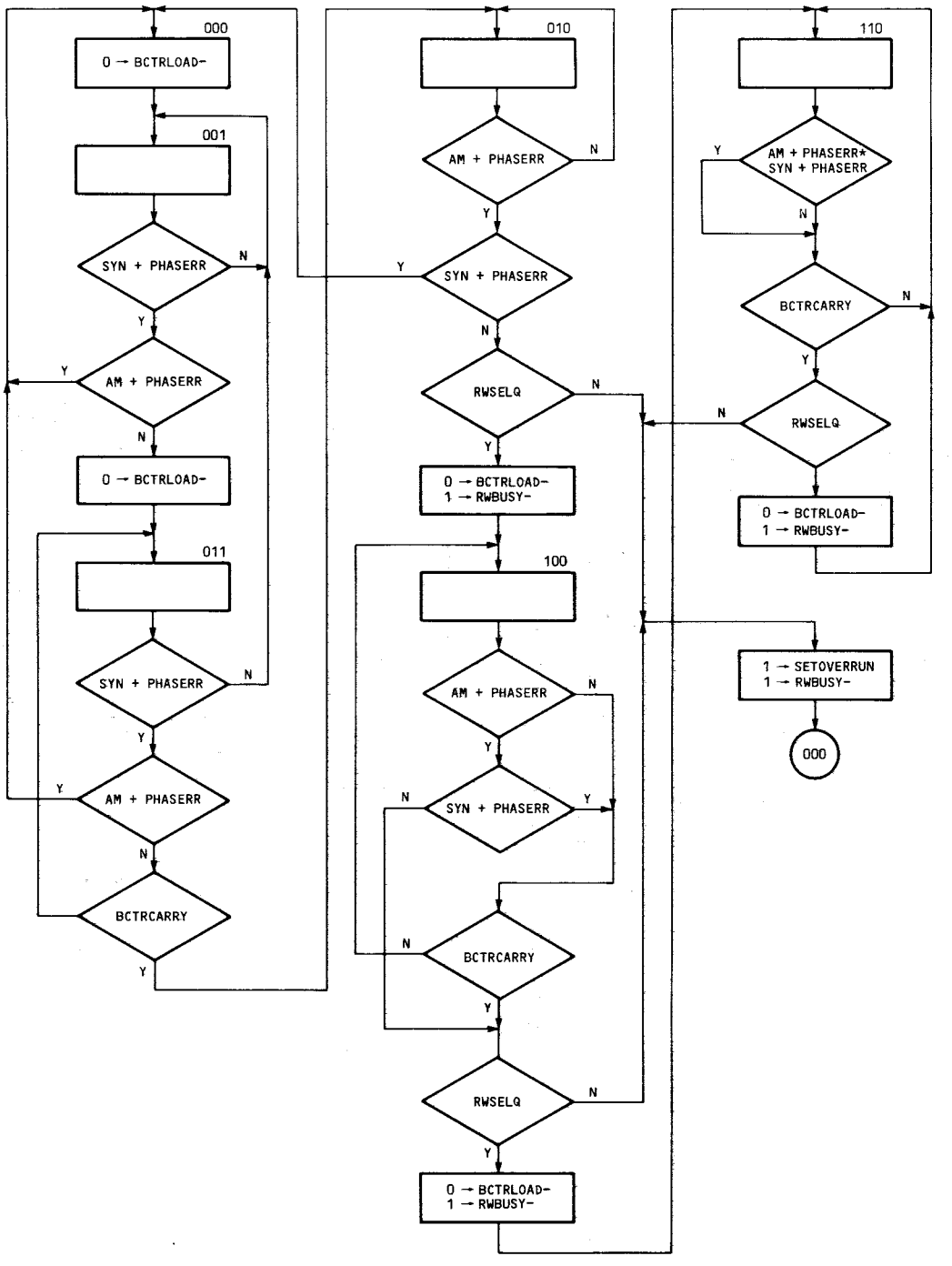


FIGURE 4-21. WORD CONTROLLER BLOCK DIAGRAM



NOTES

1. Binary States (000 to 110) are shown atop some rectangular blocks some blocks are left blank (used only to show state).

2. The final procedure block in the lower right is the last block to states 010, 100, and 110.

FIGURE 4-22. WORD CONTROLLER READ MODE FLOWCHART

TABLE 4-15. WORD CONTROLLER LOGIC EQUATIONS

```
PHASEGOOD =  READENABLE * AM+PHASERR * SYN+PHASERR * ( 000 + 001 + 010 + 011 )

RWBUSY- = READENABLE * RWSELQ * BCTRLOAD + READENABLE- * RWSELQ * BCTRCARRY

SETOVERRUN = READENABLE * RWSELQ- * BCTRLOAD * + READENABLE- * RWSELQ-
             * BCTRCARRY

BCTRLOAD = READENABLE * ( 001 * SYN * AM- +  011  * BCTRCARRY * (SYN * AM)-
           + 010  * AM * SYN- +  100  * (BCTRCARRY + AM * SYN-) + 110 *
           BCTRCARRY) + READENABLE- * RWSELQ * BCTRCARRY

DCMDLOAD = RWBUSY- * RWREAD-
```

NOTE

Binary numbers (e.g. 011) represent the word controller state defined by the signals RW3Q, RW2Q, and RW1Q.

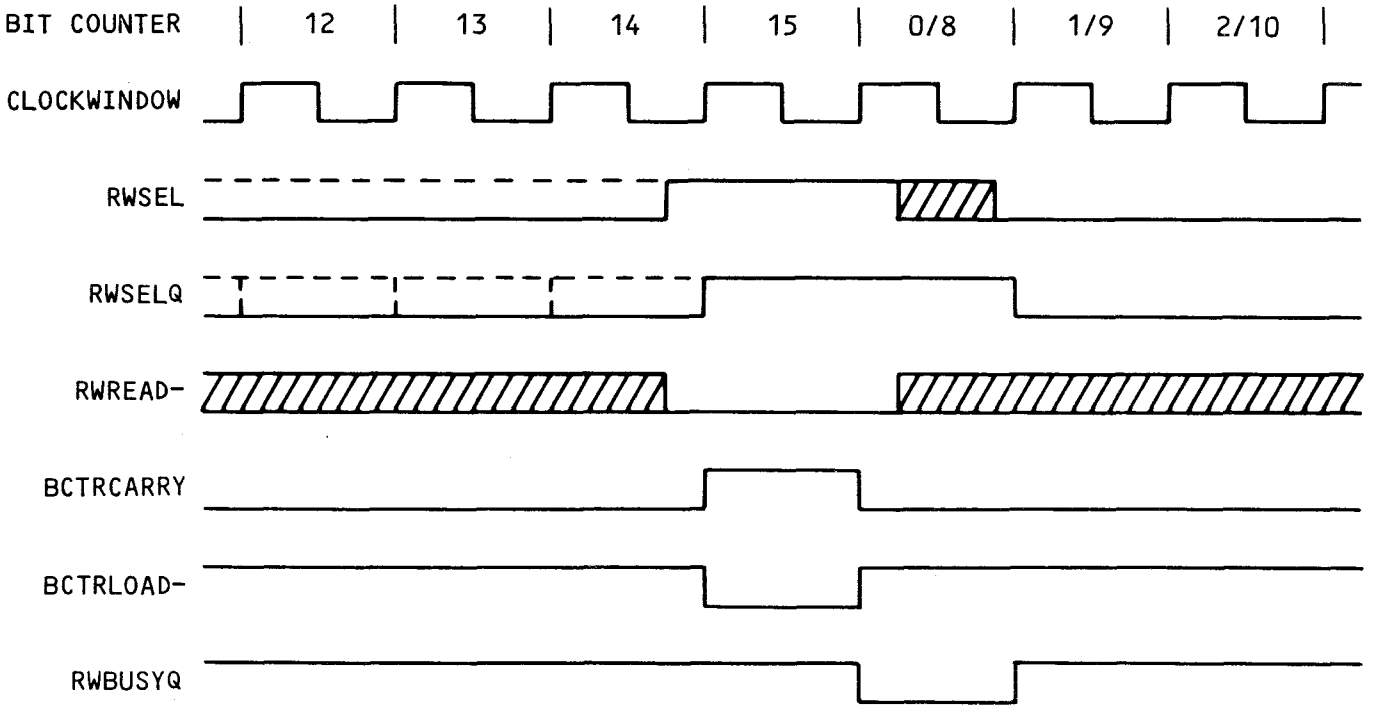


FIGURE 4-23. WORD CONTROLLER READ TIMING

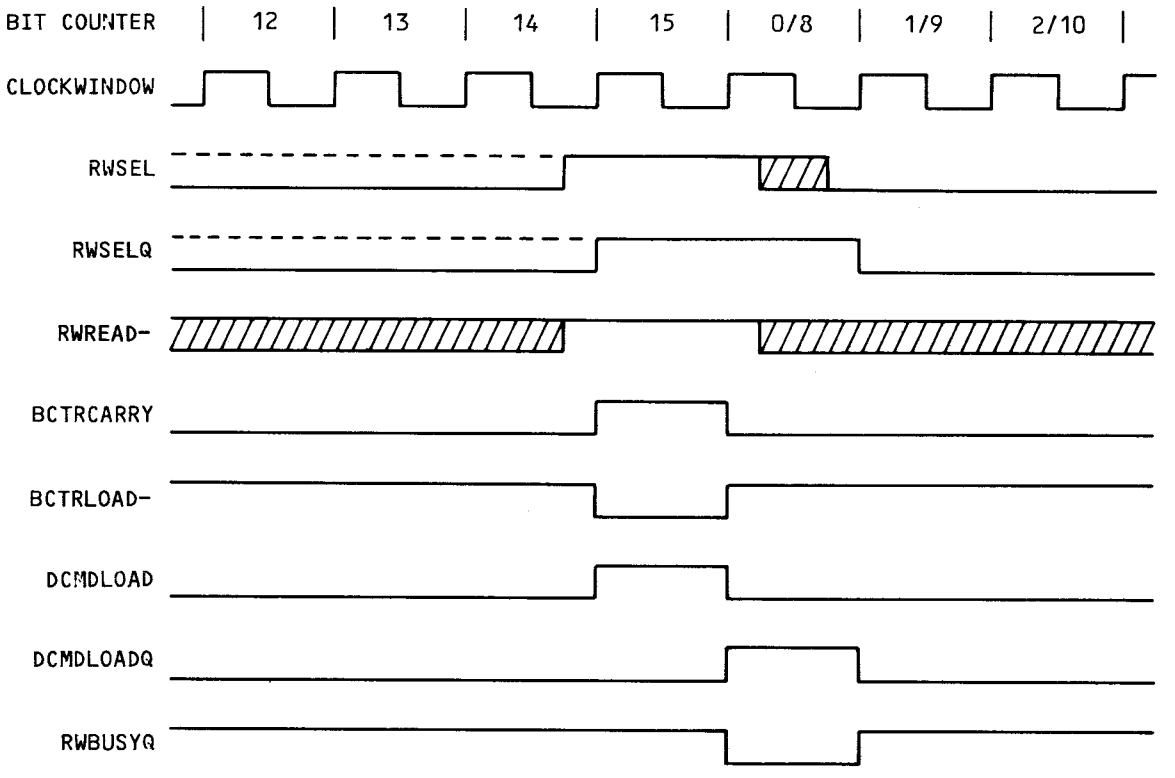


FIGURE 4-24. WORD CONTROLLER WRITE TIMING

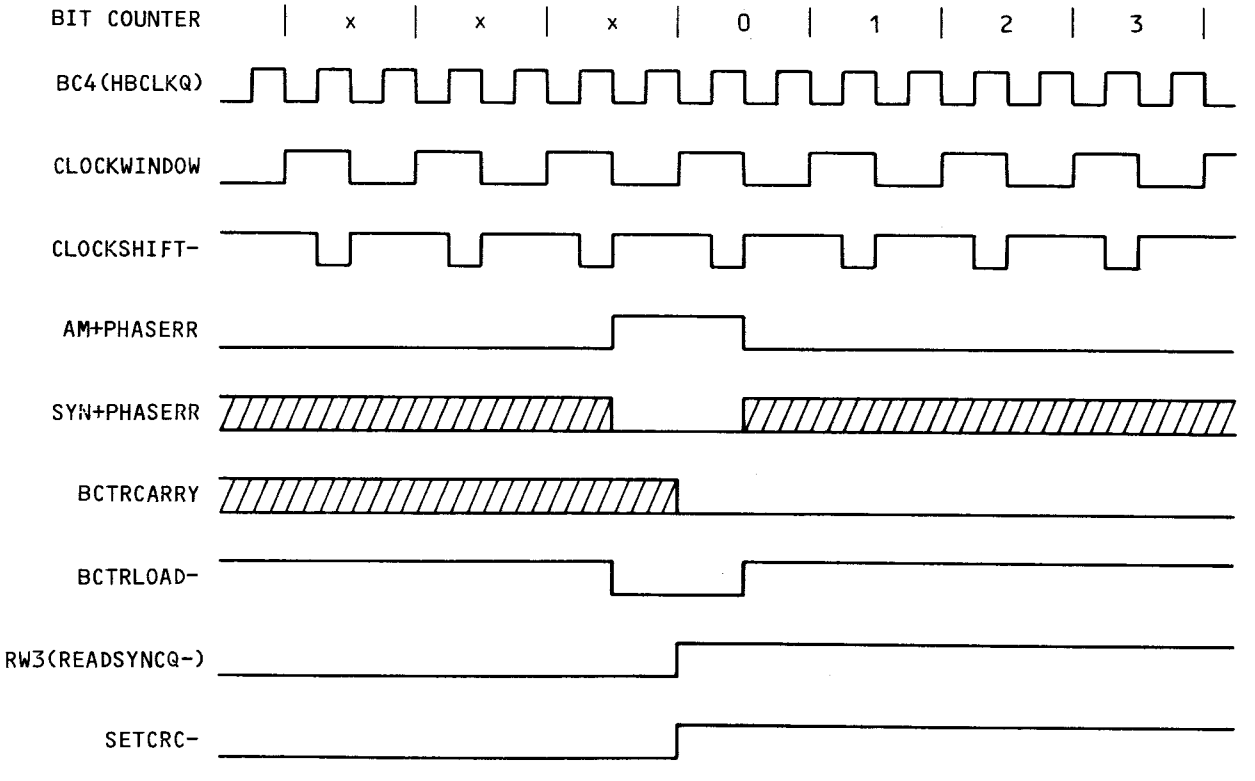


FIGURE 4-25. WORD CONTROLLER READ TIMING-ADDRESS MARK DETECT

4.7.6 Control of Read and Write Operations

Read and write operations are controlled by the CRU output bits DDENSITY and RFMTWPCOMP and by the control register. Writing an ID or data field is accomplished by changing the control register contents while loading the data into the read/write controller. The ID field write timing is shown in Figures 4-26 and 4-28. Reading an ID or data field is accomplished by using the control register to start the word controller. The word controller detects the address mark, synchronizes the data, and starts the CRC checking process. The ID field read timing is given in Figures 4-27 and 4-29.

The CRC check output is latched in the CRCERROR flip-flop and held valid for one 16-bit word time. The CRCERROR signal must be sampled during this time at the end of a read operation. The CRC error latch timing is shown in Figure 4-30.

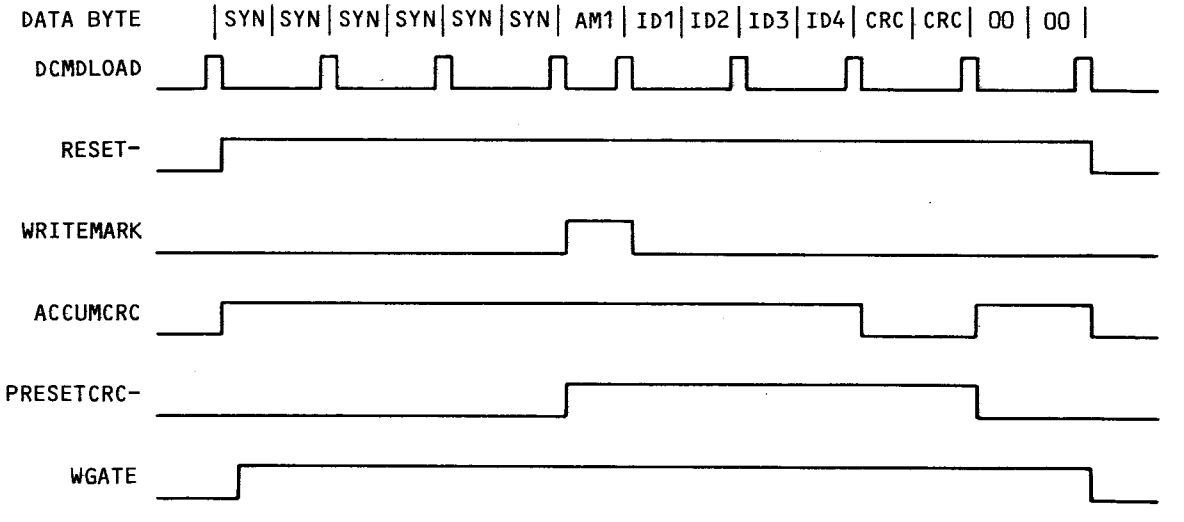


FIGURE 4-26. ID FIELD WRITE TIMING - SINGLE DENSITY

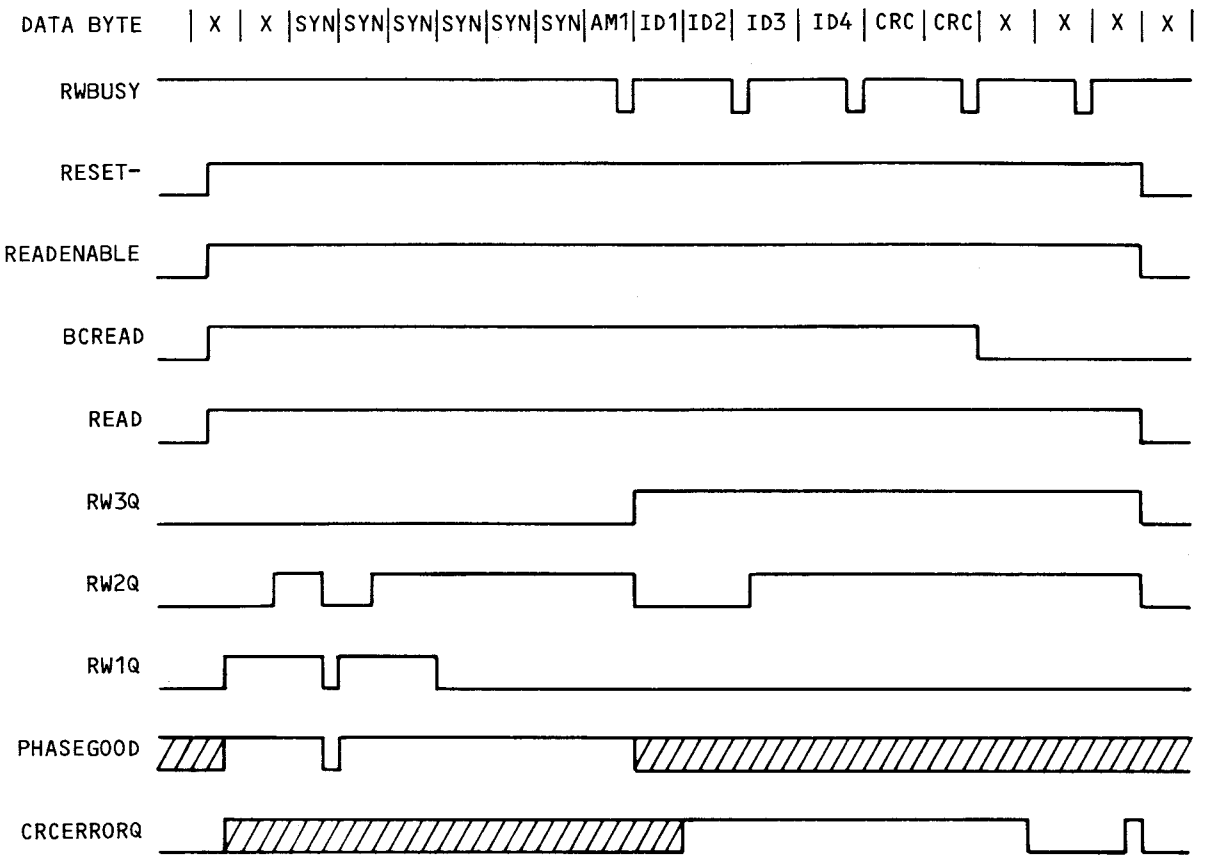


FIGURE 4-27. ID FIELD READ TIMING - SINGLE DENSITY

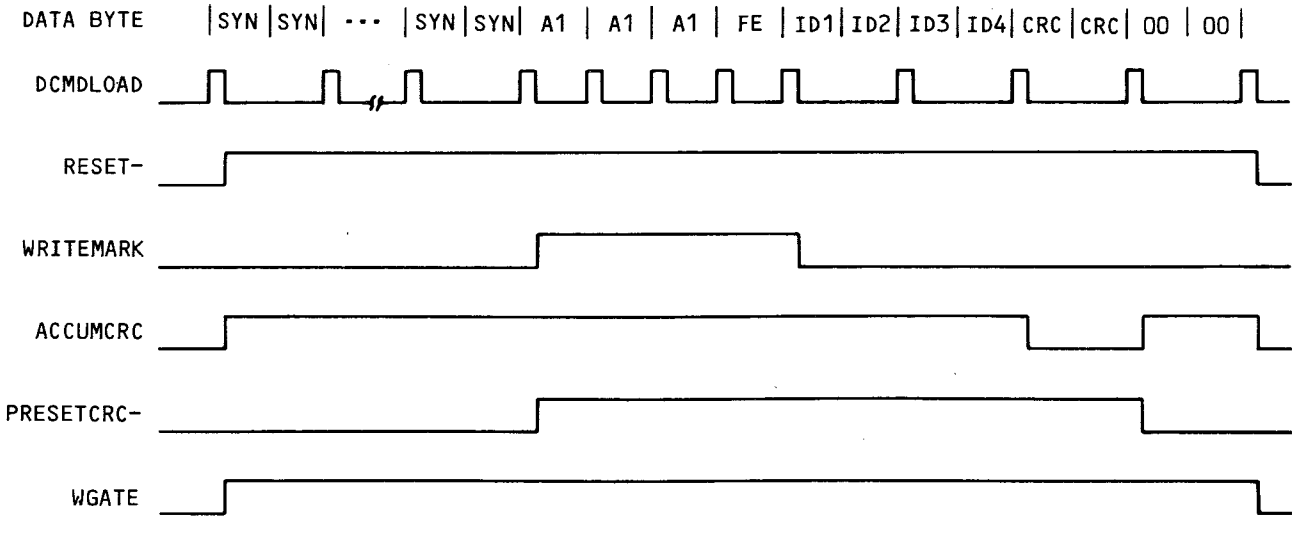


FIGURE 4-28. ID FIELD WRITE TIMING - IBM DOUBLE DENSITY

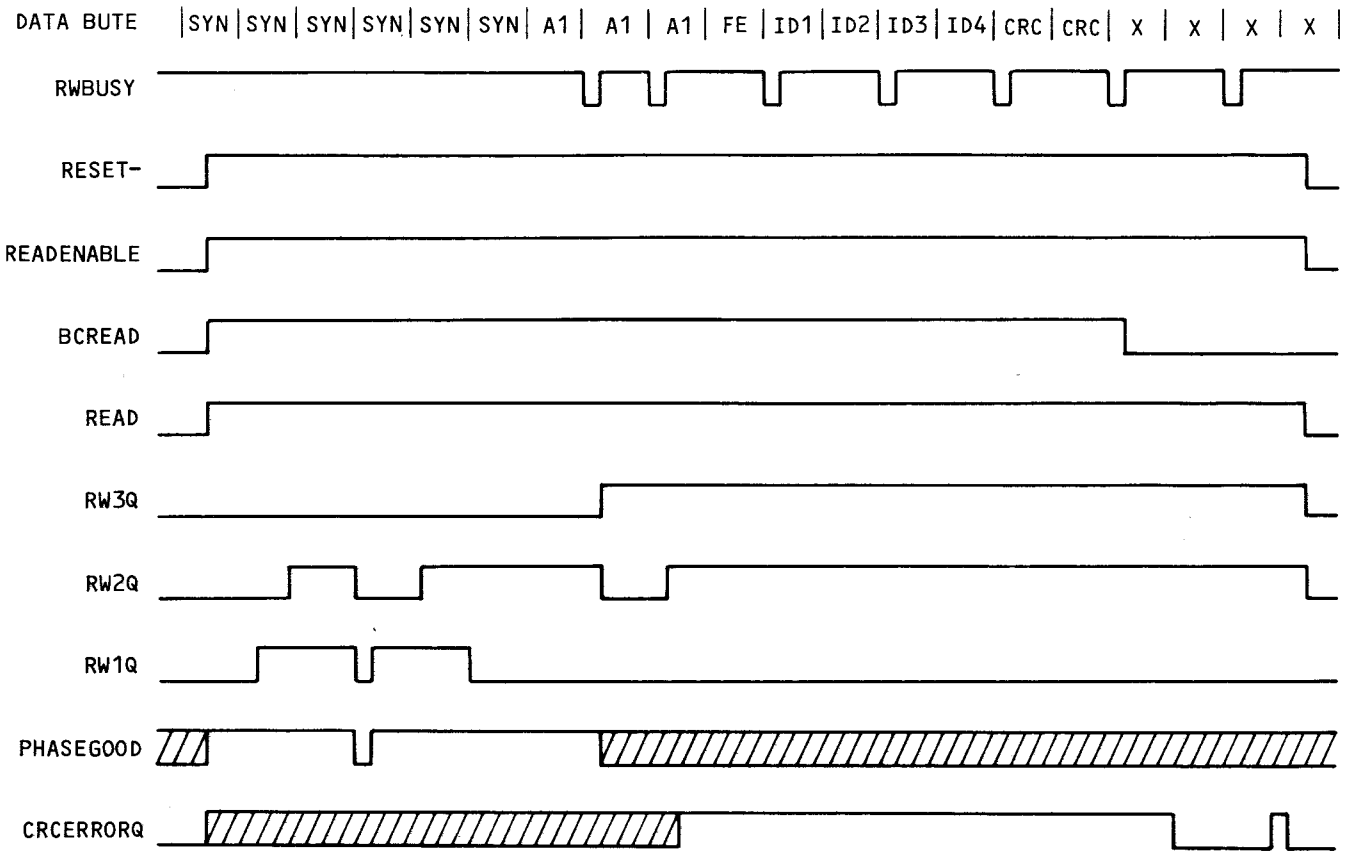


FIGURE 4-29. ID FIELD READ TIMING - IBM DOUBLE DENSITY

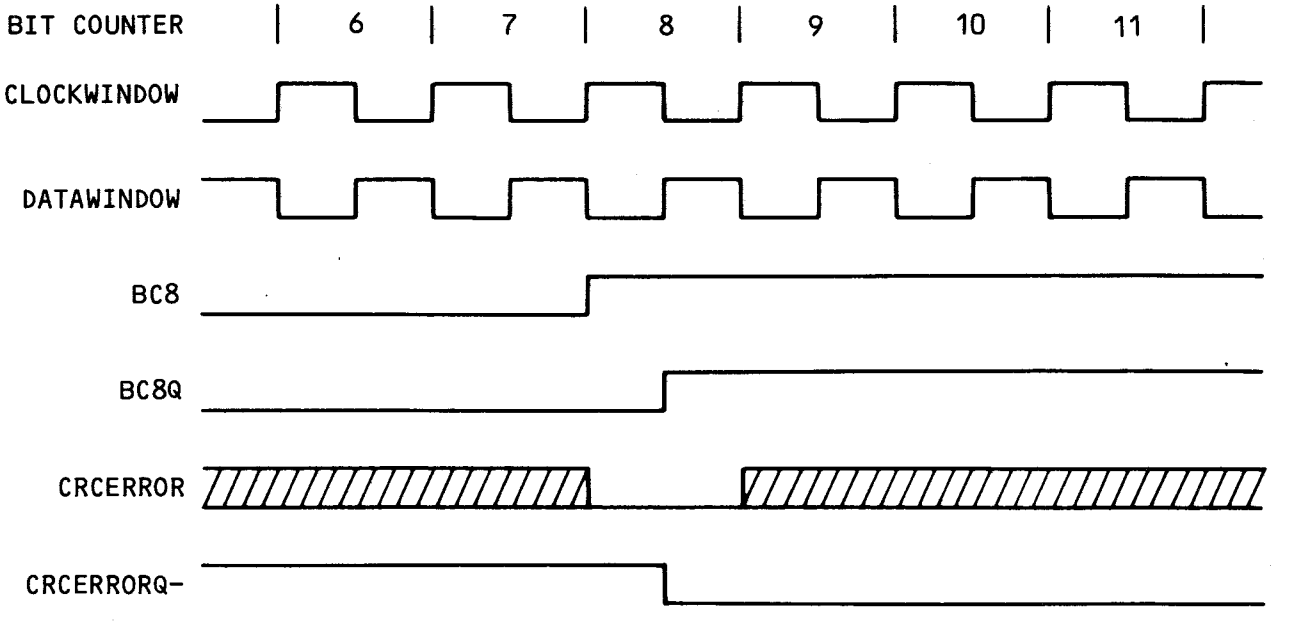


FIGURE 4-30. CRC ERROR LATCH TIMING

APPENDIX A

DISKETTE FORMATS AND CORRESPONDING DISK DRIVES

Diskette Formats	Disk Drive Formats			
	CDC 9404B	SA 400	SA 800	Qume DT-8
IBM Single Density/1 Side	X	X	X	X
IBM Double Density/1 Side		X	X	X
TI Double Density/1 Side			X	X
IBM Single Density/2 Sides				X
IBM Double Density/2 Sides				X
TI Double Density/2 Sides				X

		Hex Value
Word 0	DISK SIZE (Std=1) (0–7) | SURFACES (1) (8–15)	0101
Word 1	NUMBER OF TRACKS (77) (0–15)	004D
Word 2	HEAD STEP TIME (1000 10-us units) (0–15)	03E8
Word 3	STEP SETTLING TIME (1500 10-us units) (0–15)	05DC
Word 4	HEAD LOAD TIME (3500 10-us units) (0–15)	0DAC
Word 5	HEAD UNLOAD TIMEOUT (1000 ms) (0–15)	03E8
Word 6	DENSITY FM=0 (0–3) | SYNC TYPE IBM=0 (4–7) | INTERLACE FACTOR=0 (8–15)	0000
Word 7	SEC/TRACK=26 (0–5) | BYTES PER SECTOR=128 (6–15)	6880

Notes: 1. This is the default format for bootstrap load with jumper J8 installed.
2. For Shugart 800 only.

FIGURE A-1. IBM SINGLE DENSITY DEFINE DRIVE FORMAT BLOCK, SHEET 1 OF 3 (SHUGART 800 ONLY)

			Hex Value
Word 0	DISK SIZE (Std=1) (0–7)	SURFACES (1) (8–15)	0101
Word 1	NUMBER OF TRACKS (77) (0–15)		004D
Word 2	HEAD STEP TIME (2750 10-us units) (0–15)		0ABE
Word 3	STEP SETTLING TIME (1500 10-us units) (0–15)		05DC
Word 4	HEAD LOAD TIME (6000 10-us units) (0–15)		1770
Word 5	HEAD UNLOAD TIMEOUT (1000 ms) (0–15)		03E8
Word 6	DENSITY FM=0 (0–3)	SYNC TYPE IBM=0 (4–7)	INTERLACE FACTOR=0 (8–15)	0000
Word 7	SEC/TRACK=26 (0–5)	BYTES PER SECTOR=128 (6–15)	6880

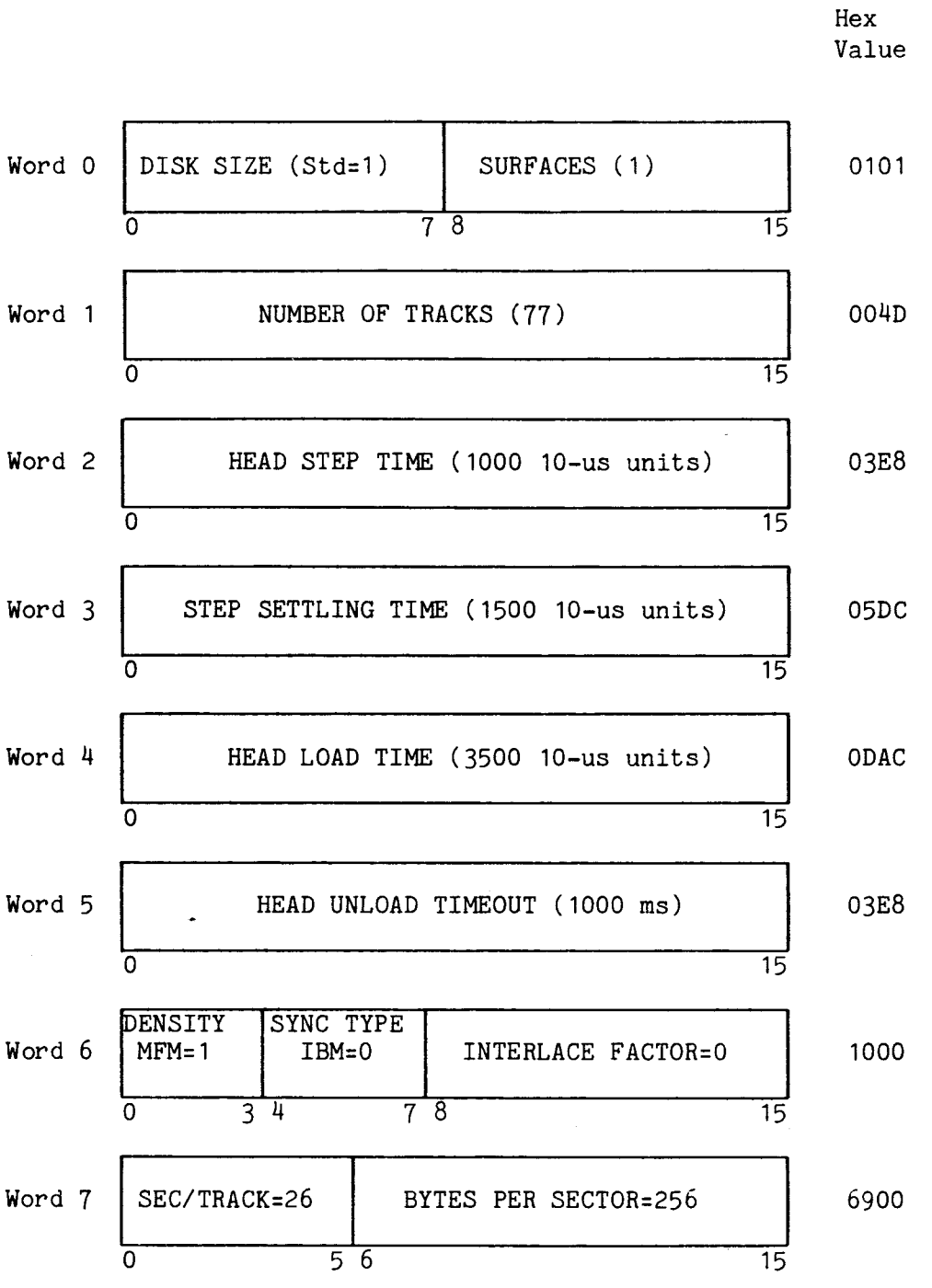
Note: For CDC 9404B only.

FIGURE A-1. IBM SINGLE DENSITY DEFINE DRIVE FORMAT BLOCK, SHEET 2 OF 3
(CDC 9404B ONLY)

			Hex Value
Word 0	DISK SIZE (Std=1) (0–7)	SURFACES (2) (8–15)	0102
Word 1	NUMBER OF TRACKS (77) (0–15)		004D
Word 2	HEAD STEP TIME (300 10-us units) (0–15)		012C
Word 3	STEP SETTLING TIME (1500 10-us units) (0–15)		05DC
Word 4	HEAD LOAD TIME (3500 10-us units) (0–15)		0DAC
Word 5	HEAD UNLOAD TIMEOUT (1000 ms) (0–15)		03E8
Word 6	DENSITY FM=0 (0–3)	SYNC TYPE IBM=0 (4–7)	INTERLACE FACTOR=0 (8–15)	0000
Word 7	SEC/TRACK=26 (0–5)	BYTES PER SECTOR=128 (6–15)	6880

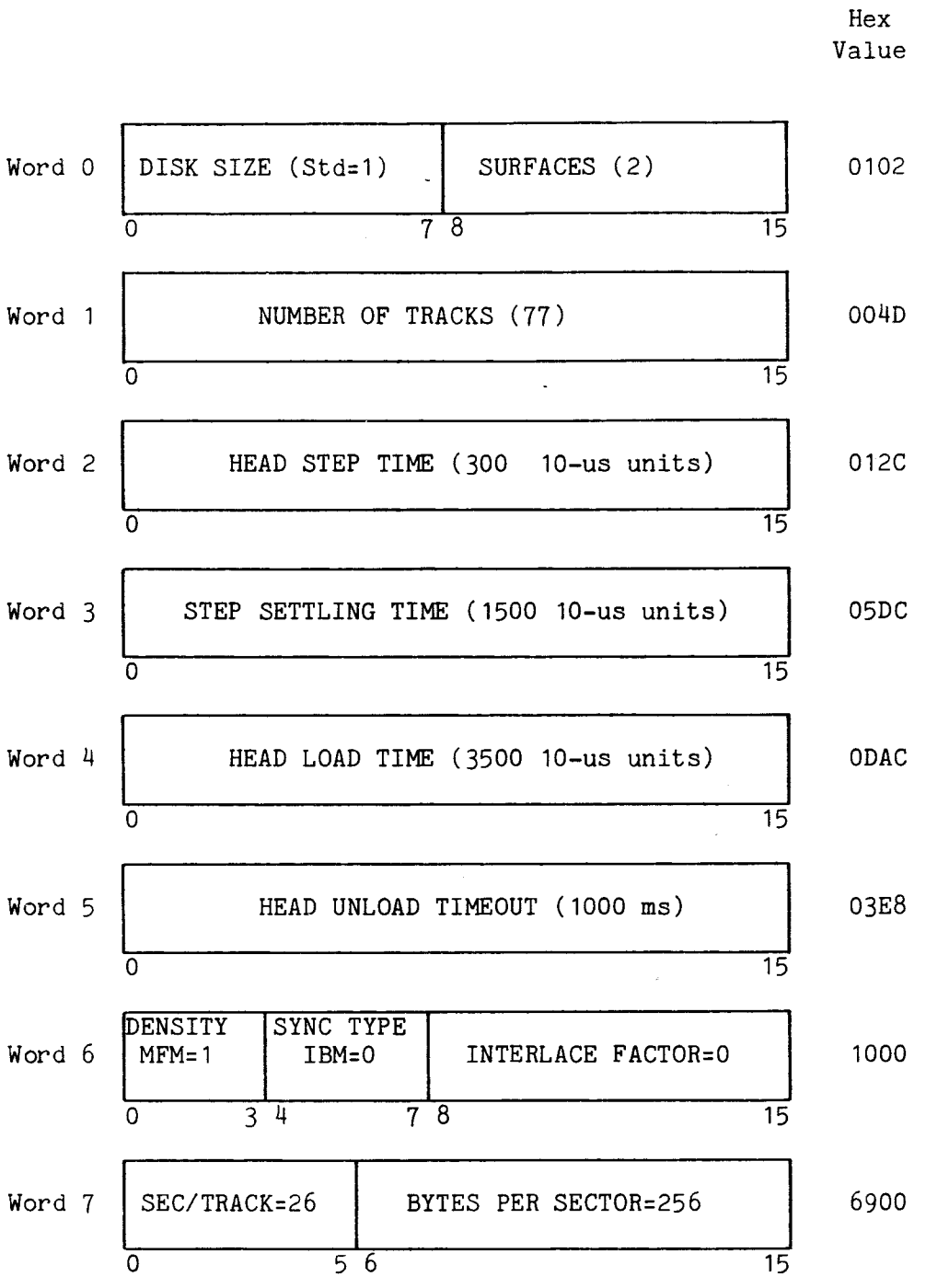
Note: For Qume DataTrak 8 only.

FIGURE A-1. IBM SINGLE DENSITY DEFINE DRIVE FORMAT BLOCK, SHEET 3 OF 3)
(QUME DATATRAK 8 ONLY)



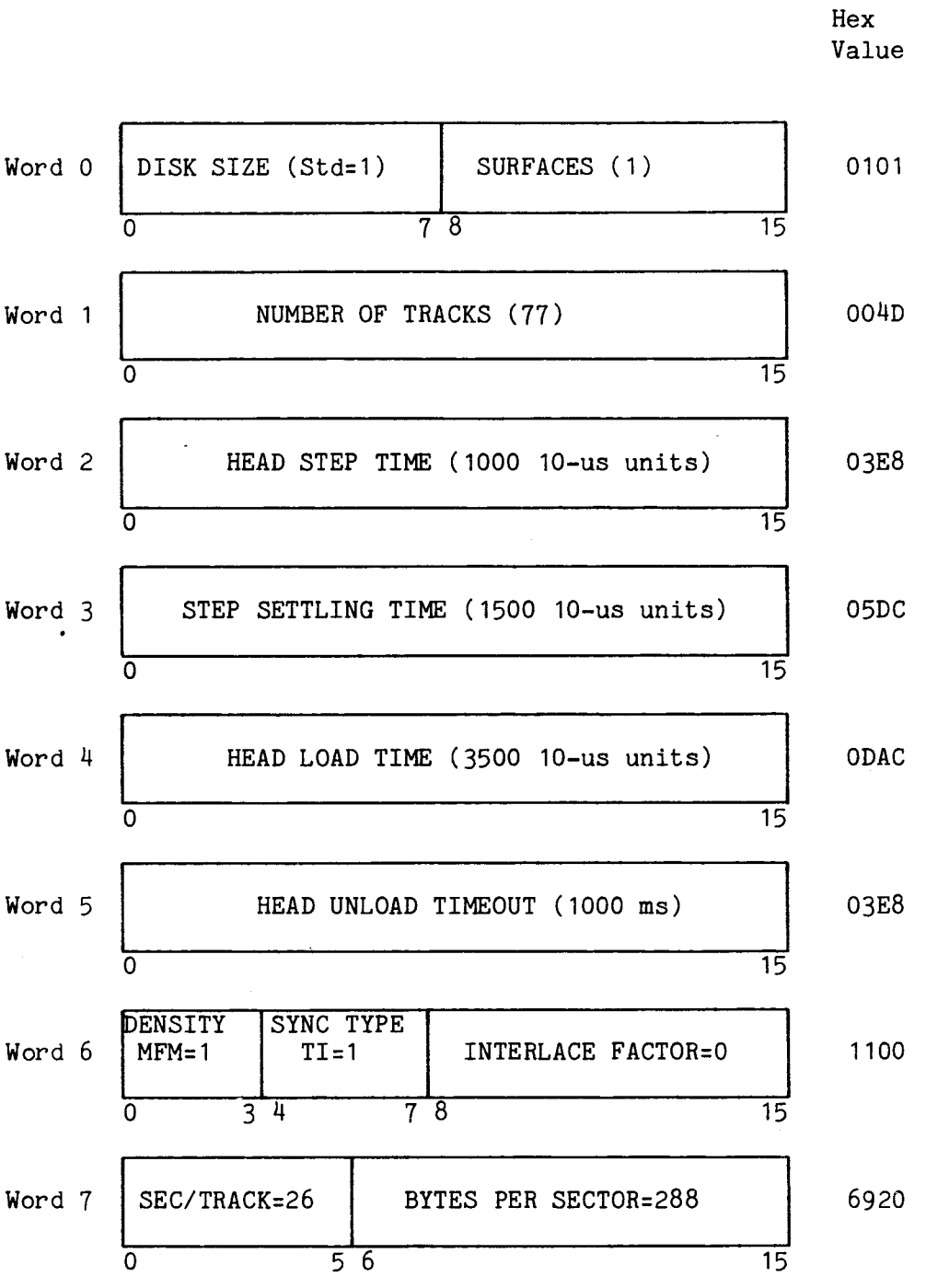
Note: For Shugart 800 only.

FIGURE A-2. IBM DOUBLE DENSITY DEFINE DRIVE FORMAT BLOCK, SHEET 1 OF 2 (SHUGART 800 ONLY)



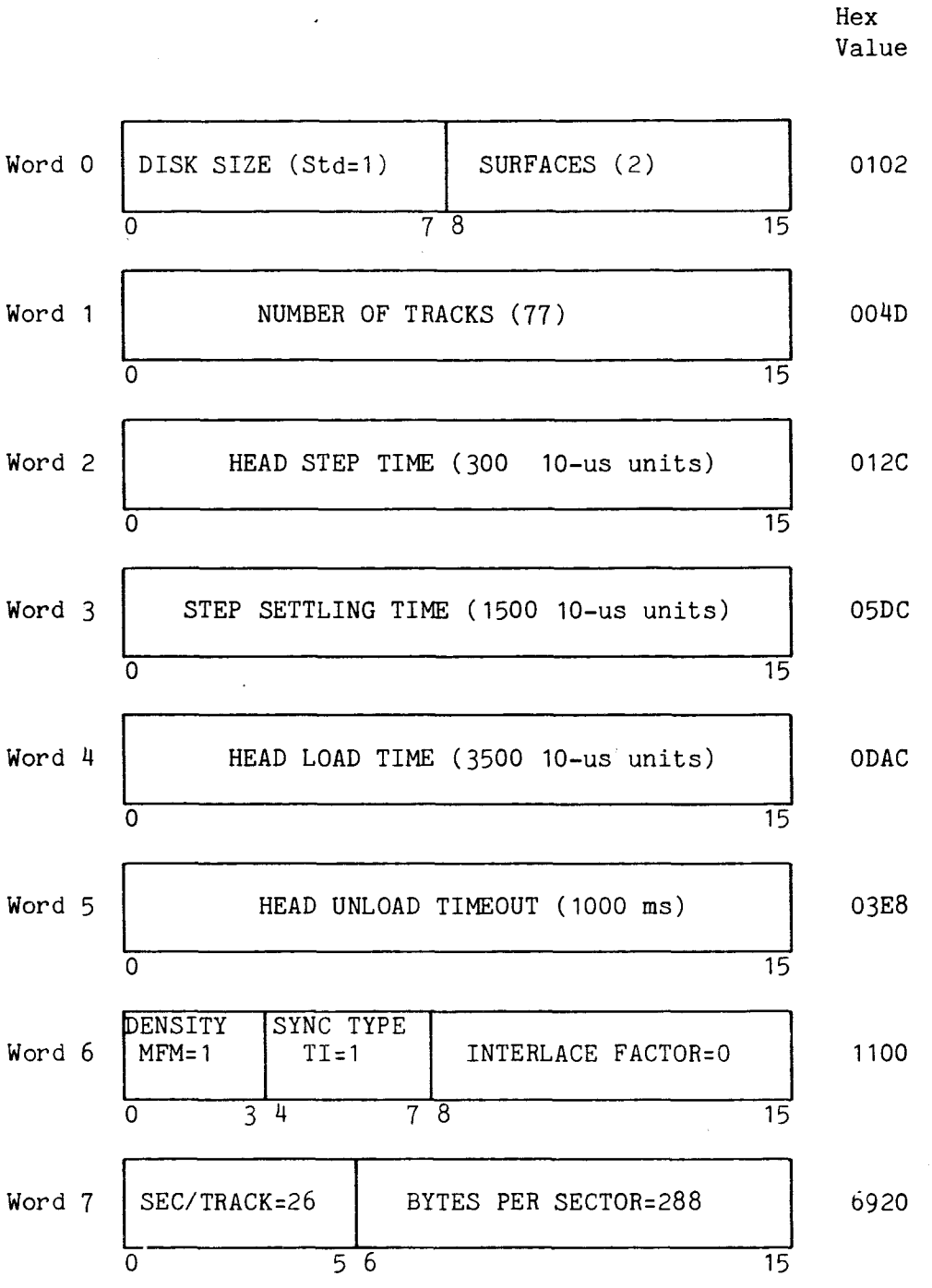
Note: For Qume DataTrak 8 only.

FIGURE A-2. IBM DOUBLE DENSITY DEFINE DRIVE FORMAT BLOCK, SHEET 2 OF 2
(QUME DATATRAK 8 ONLY)



Note: For Shugart 800 only.

FIGURE A-3. TI DOUBLE DENSITY DEFINE DRIVE FORMAT BLOCK, SHEET 1 OF 2
(FOR SHUGART 800 ONLY)



Note: For Qume DataTrak 8 only.

FIGURE A-3. TI DOUBLE DENSITY DEFINE DRIVE FORMAT BLOCK, SHEET 2 OF 2
(FOR DATATRAK 8 ONLY)

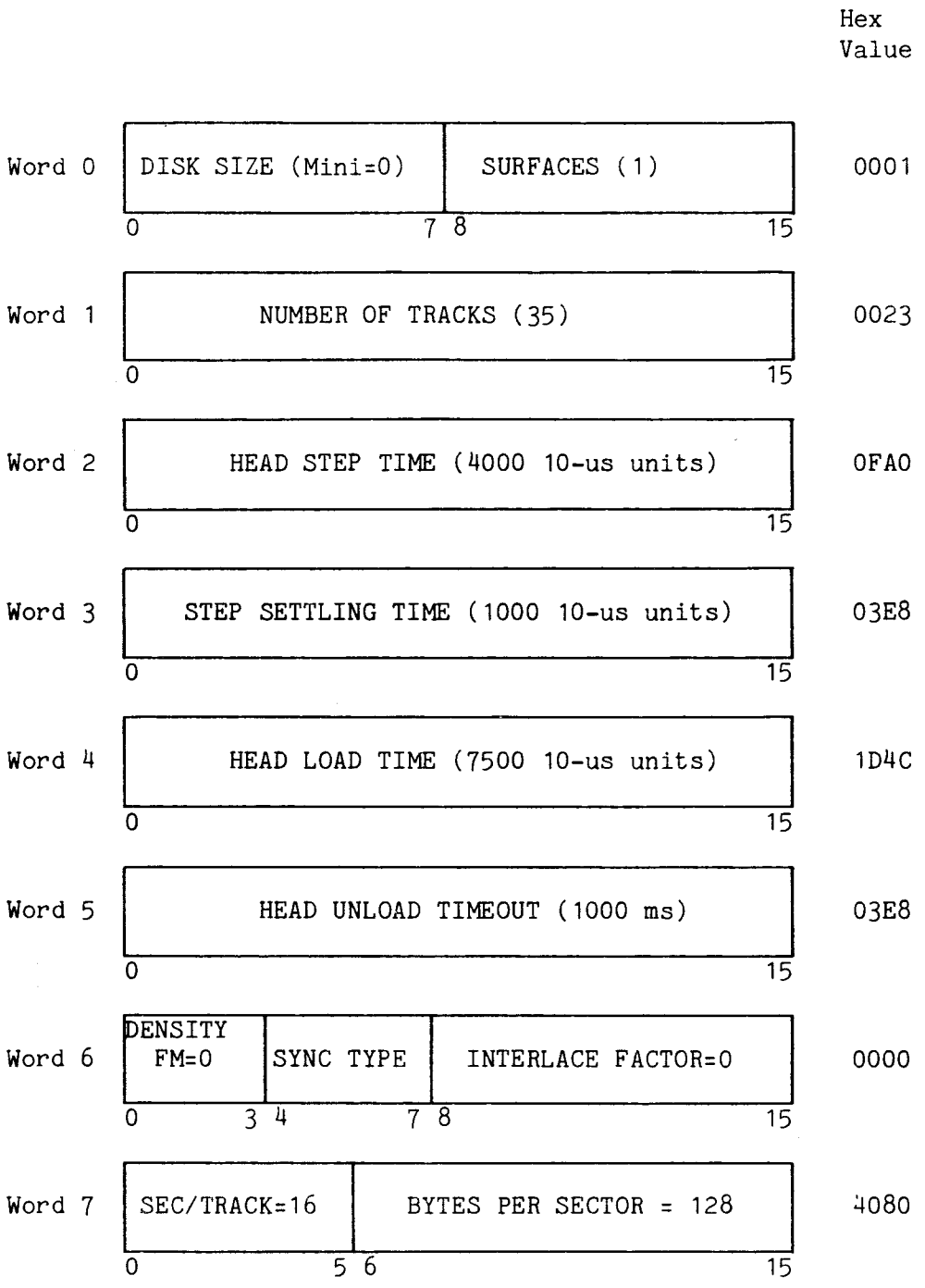


FIGURE A-4. MINI SINGLE DENSITY DEFINE DRIVE FORMAT BLOCK

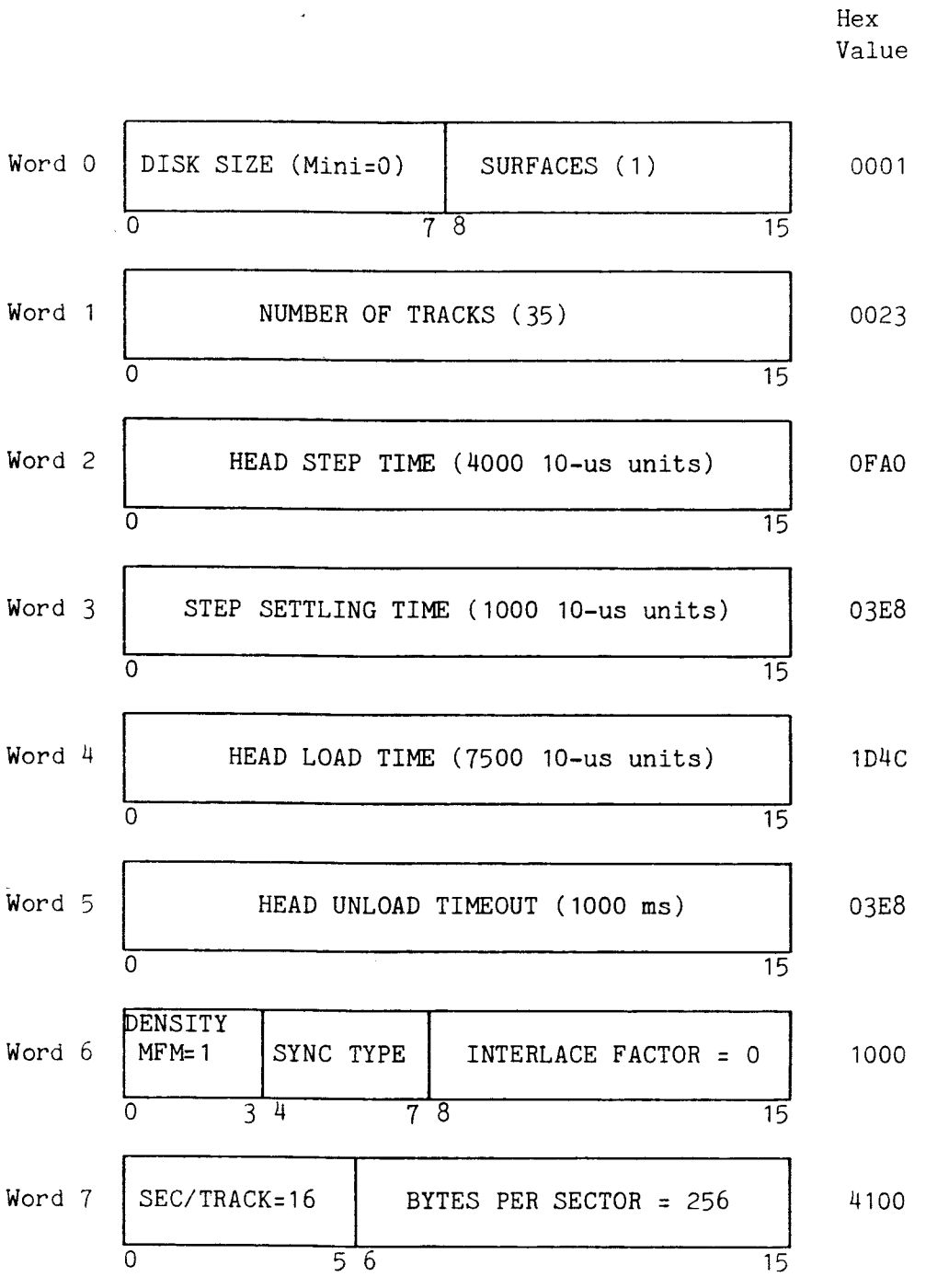


FIGURE A-5. MINI DOUBLE DENSITY DEFINE DRIVE FORMAT BLOCK

APPENDIX B

DISK DRIVE SPECIFICATIONS

TABLE B-1. MINI (5.25 INCH) FLOPPY DRIVE SPECIFICATIONS

	Single Density	Double Density	Measure
PERFORMANCE SPECIFICATIONS			
Diskette Capacity			
Unformatted			
Per Surface	875	1750	kilobits
Per Track	25	50	kilobits
IBM Format			
Per Surface	71,680	143,360	bytes
Per Track	2,048	4,096	bytes
Per Sector	128	256	bytes
Transfer Rate			
Total	125	250	kilobits/sec
Data	81	162	kilobits/sec
Rotational Latency (Avg)	100	100	milliseconds
Seek Time			
Track to Track	40	40	milliseconds
Average	463	463	milliseconds
Settling Time	10	10	milliseconds
Head Load Time	75	75	milliseconds
FUNCTIONAL SPECIFICATIONS			
Rotational Speed	300	300	rpm
Recording Density (Inside Track)	2581	5162	bpi
Flux Density	5162	5162	fci
Track Density	48	48	tpi
Tracks	35	35	
Sectors per Track (Soft)	16	16	
Index	1	1	
Encoding Method	FM	MFM	
Media Requirements (2 Sided)	SA104	SA104	

TABLE B-2. STANDARD (8 INCH) FLOPPY DRIVE SPECIFICATIONS

	Single Density	Double Density	Measure
PERFORMANCE SPECIFICATIONS			
Diskette Capacity			
Unformatted			
Per Surface	3.2	6.4	megabits
Per Track	41.7	83.3	kilobits
IBM Format			
Per Surface	256,256	512,512	bytes
Per Track	3,328	6,656	bytes
Per Sector	128	256	bytes
TI Format			
Per Surface		576,576	bytes
Per Track		7,488	bytes
Per Sector		288	bytes
Transfer Rate			
Total	250	500	kilobits/sec
Data	159	318	kilobits/sec
Rotational Latency (Avg)	83	83	milliseconds
Seek Time			
Track to Track	10	10	milliseconds
Average	260	260	milliseconds
Settling Time	8	8	milliseconds
Head Load Time	35	35	milliseconds
FUNCTIONAL SPECIFICAITIONS			
Rotational Speed	360	360	rpm
Recording Density (Inside Track)	3200	6400	bpi
Flux Density	6400	6400	fci
Track Density	48	48	tpi
Tracks	77	77	
Sectors per Track (Soft)	26	26	
Index	1	1	
Encoding Method	FM	MFM	
Media Requirements	SA100/IBM Diskette II	SA100/IBM Diskette II	

APPENDIX C

DISKETTE TRACK FORMATS

C.1 GENERAL

This appendix shows the format associated with the five formats compatible with the TM 990/303A floppy disk controller:

Format	Figure
Standard-size, single-density track format	C-1
IBM-compatible, standard-size, double-density track format	C-2
TI-compatible, standard-size, double-density track format	C-3
Mini-size. single-density track format	C-4
Mini-size, double-density track format	C-5

C.2 NOMENCLATURE

In the figures in this appendix, circled numbers are used to identify various fields and markers. These numbers represent the following:

(1) Sync Field. This field synchronizes the diskette drive circuitry to the information being read from the diskette.

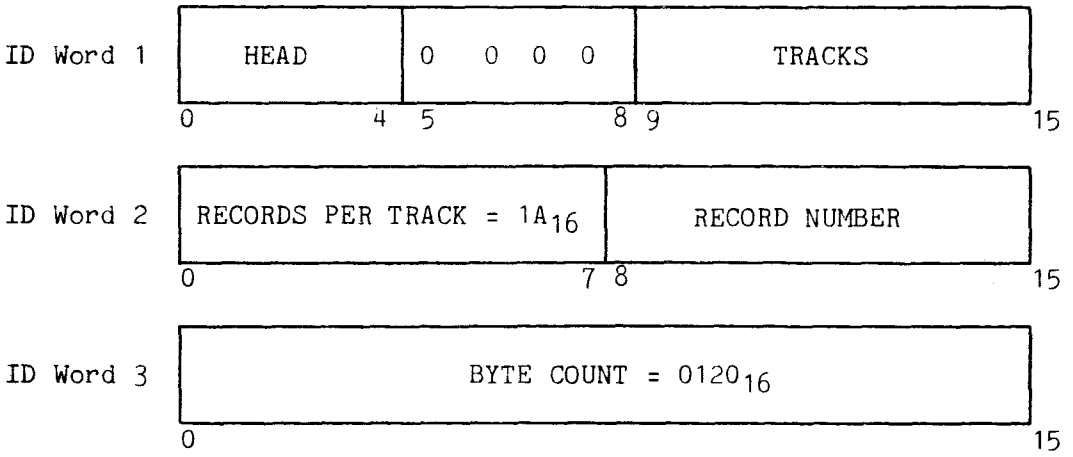
(2) AM1 (Address Marker 1). This marker identifies the information that follows as being the ID field.

(3) ID Field.

a. Formats other than TI double density. The ID field consists of four bytes of information identifying the address and size of the sector.

- First byte. Track number (00_{16} thru $4C_{16}$)
- Second byte. Head number (00_{16} = side 0, 01_{16} = side 1)
- Third byte. Record number with 1 record per sector (1 thru 26)
- Fourth byte. Physical record length in bytes:
  00_{16} = 128 bytes
  01_{16} = 256 bytes

b. TI double density format. Three words as follows:



(4) Cyclic Redundancy Check (CRC). The CRC is the 16-bit remainder value generated on a write data or write format operation by performing a polynomial division of the string of bits including the address mark and the data field using the following polynomial divisor:

$$X^{16} + X^{12} + X^{5} + 1$$

In addition, to reduce the possibility of a false CRC check, a partial remainder value of $FFFF_{16}$ is preset into the CRC generator prior to any CRC generation or checking operation.

During an ID location or read data operation, the address mark and following data, including the previously written CRC value, are again divided by the divisor polynomial. If the data does not contain any errors, the resulting remainder value in the CRC generator wil be 0000.

(5) Gap 2. This contains the fixed number of gap data bytes.

(6) AM2 (Address Marker 2). This identifies the field that follows as being a data record or a deleted data record.

(7) Data Record. A Data Record field contains the bytes of data.

(8) Gap 3. This contains two bytes of binary zeroes followed by a variable number of gap data bytes.

(9) Gap 4B. This is the pre-index gap which consists of a variable number of gap data bytes.

(10) Gap 4A. This is the post-index gap which consists of a fixed number of gap bytes.

(11) Track AM. This follows the index mark and identifies the start of a track.

(12) Gap 1. This consists of two bytes of zeroes followed by a fixed' number of gap data bytes.

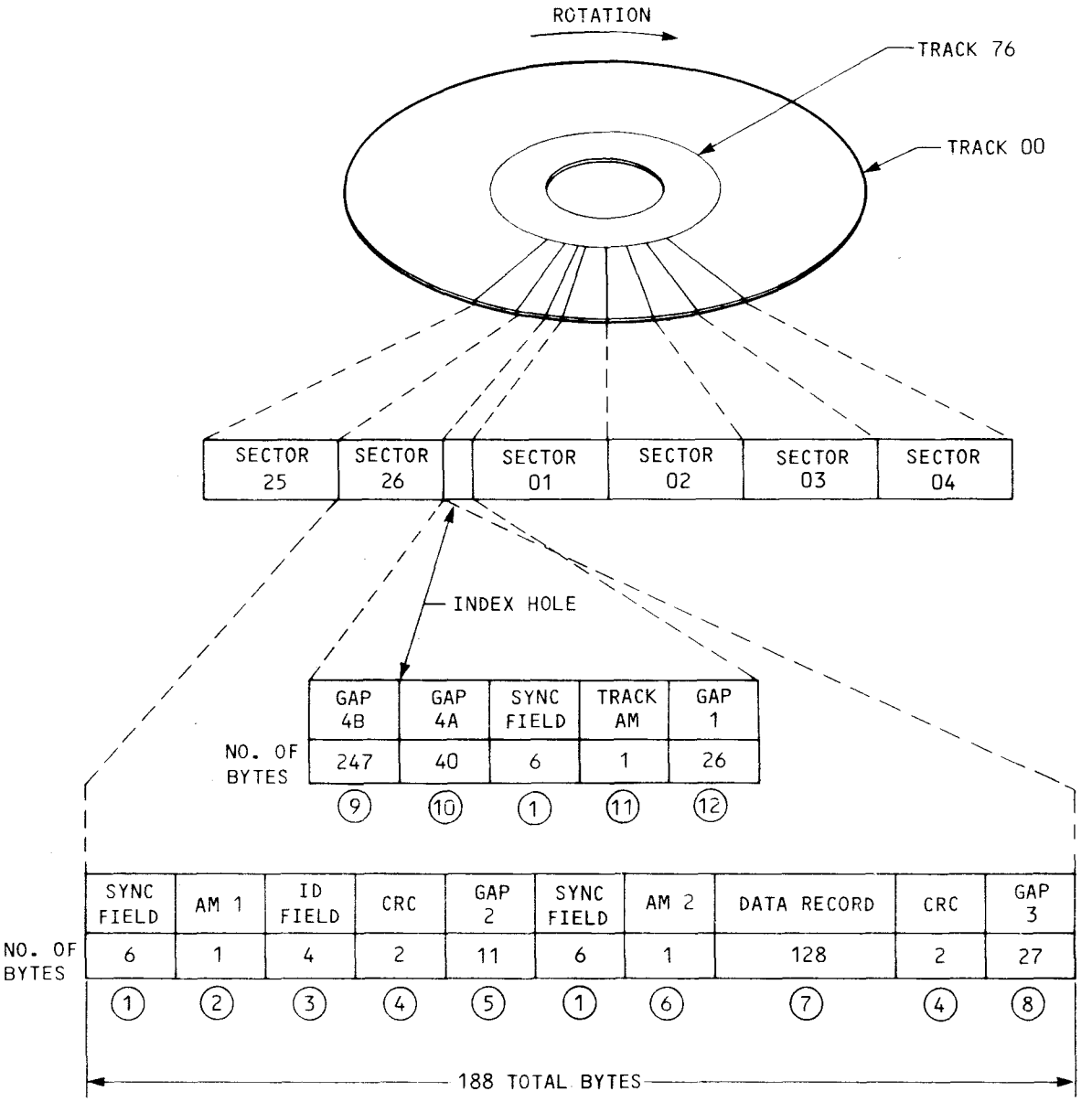
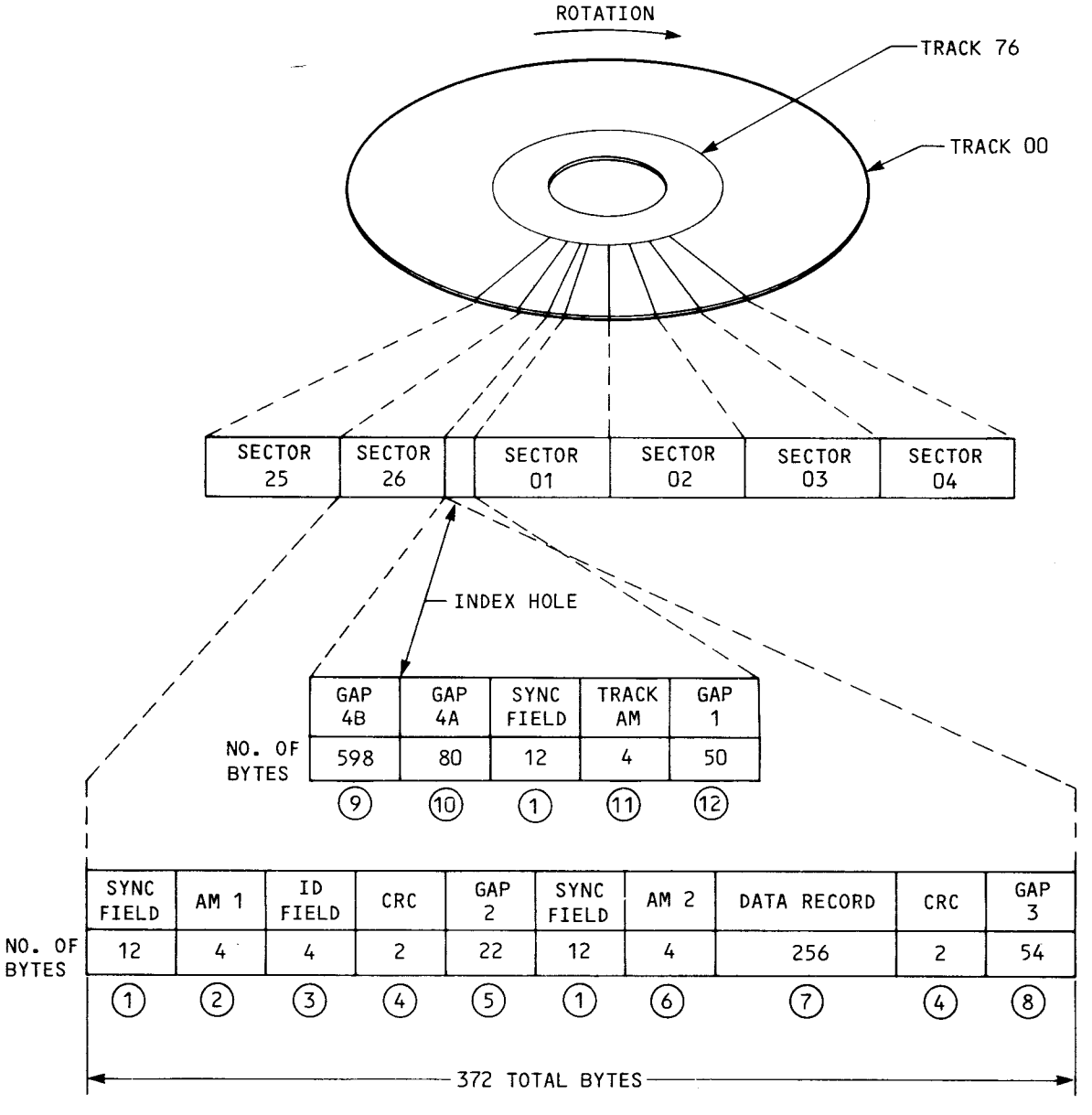


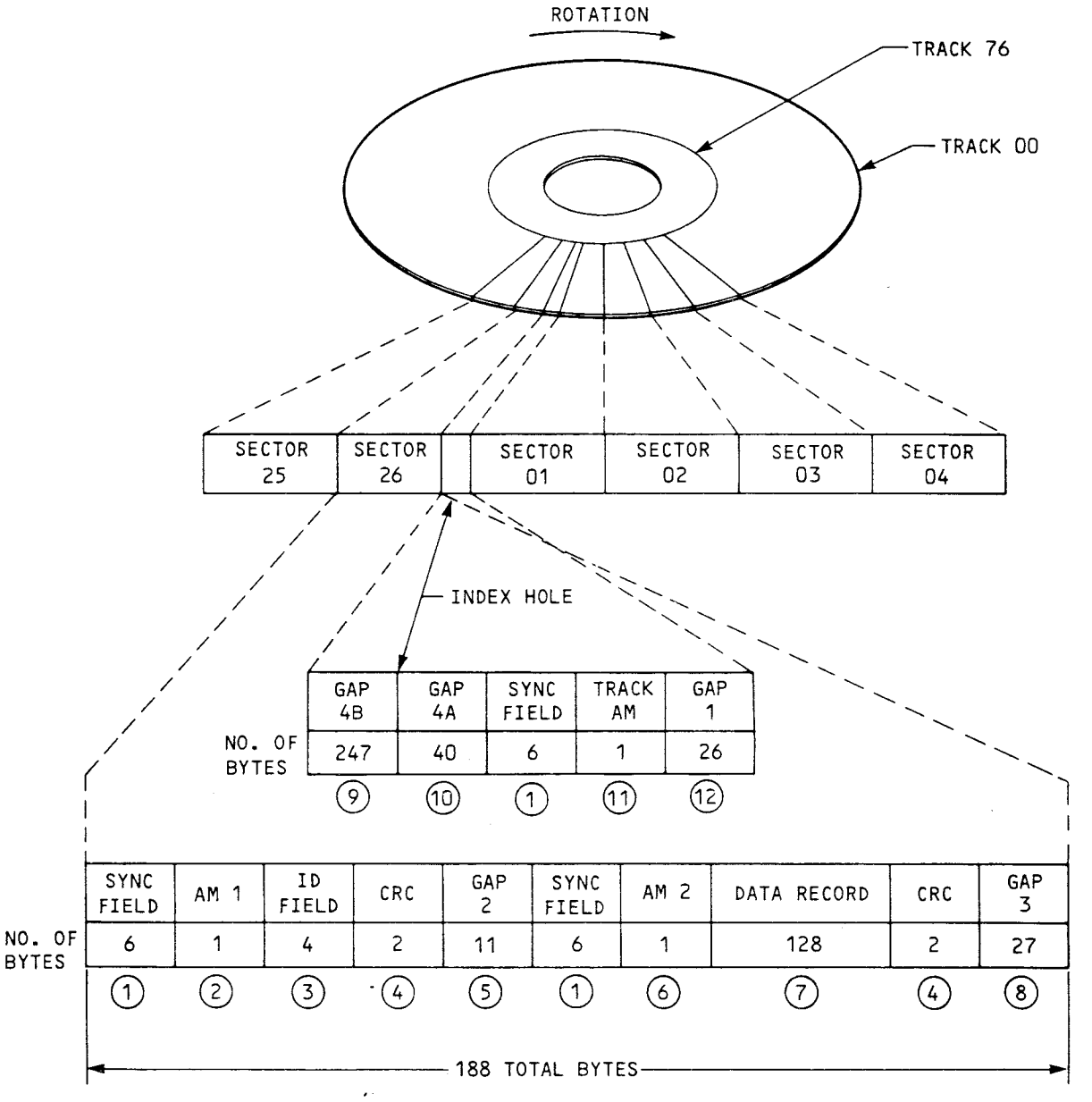
FIGURE C-1. STANDARD-SIZE, SINGLE-DENSITY TRACK FORMAT



NOTE

Shown above is for all tracks except track 0, side 0; track 0, side 0 shown in next page.

FIGURE C-2. IBM-COMPATIBLE STANDARD-SIZE DOUBLE DENSITY TRACK FORMAT (PAGE 1 OF 2)



NOTES

1. Shown above is track 0, side 0; other tracks on side 0 and all tracks on side 1 on previous page.

2. Shown are physical sector numbers. In this format for sector 0 only, only logical sectors 1 to 13 are present.

FIGURE C-2. IBM-COMPATIBLE STANDARD-SIZE DOUBLE DENSITY TRACK FORMAT (PAGE 2 OF 2)

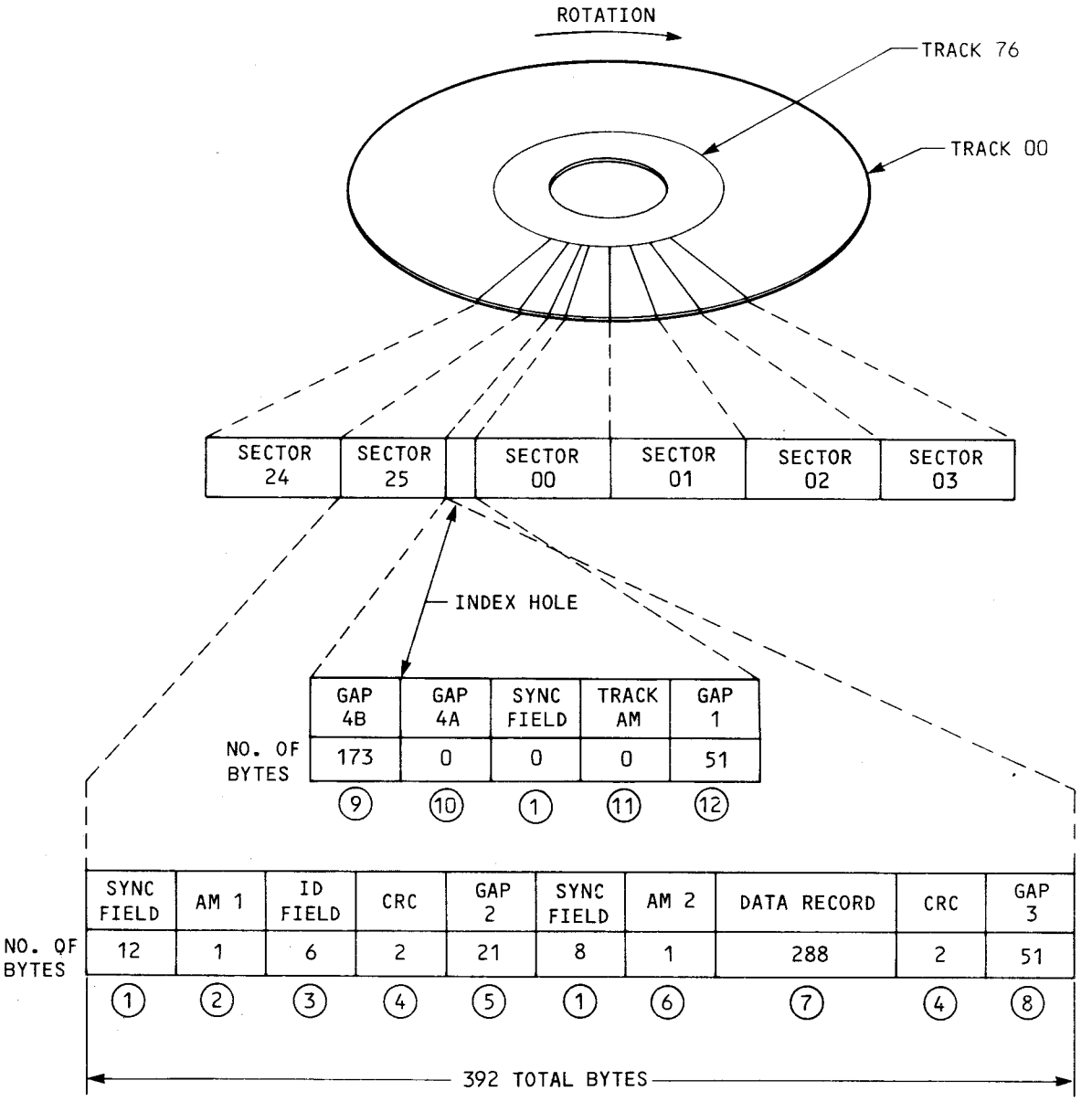


FIGURE C-3. TI-COMPATIBLE, STANDARD-SIZE, DOUBLE-DENSITY TRACK FORMAT

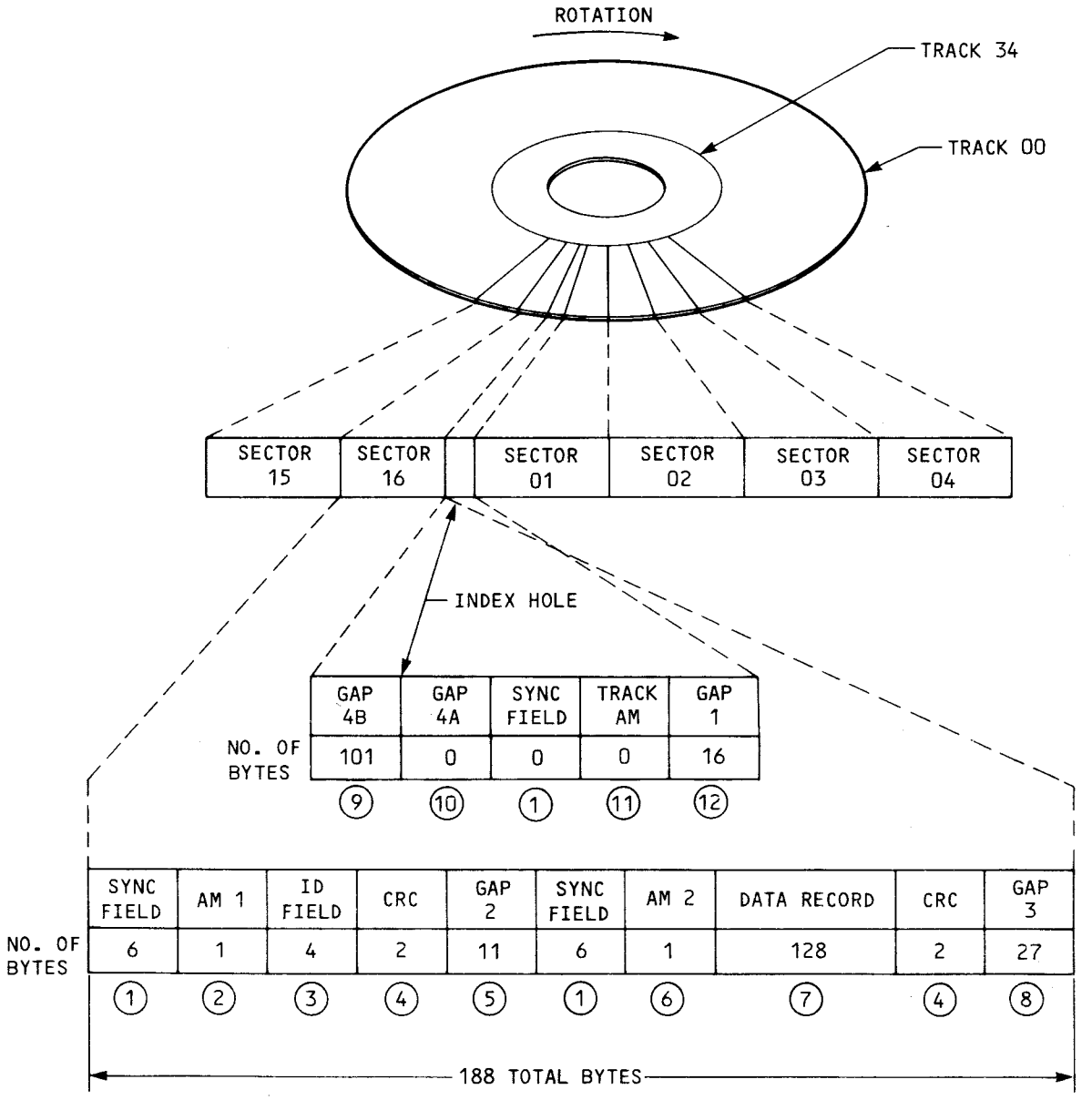


FIGURE C-4. MINI-SIZE, SINGLE-DENSITY TRACK FORMAT

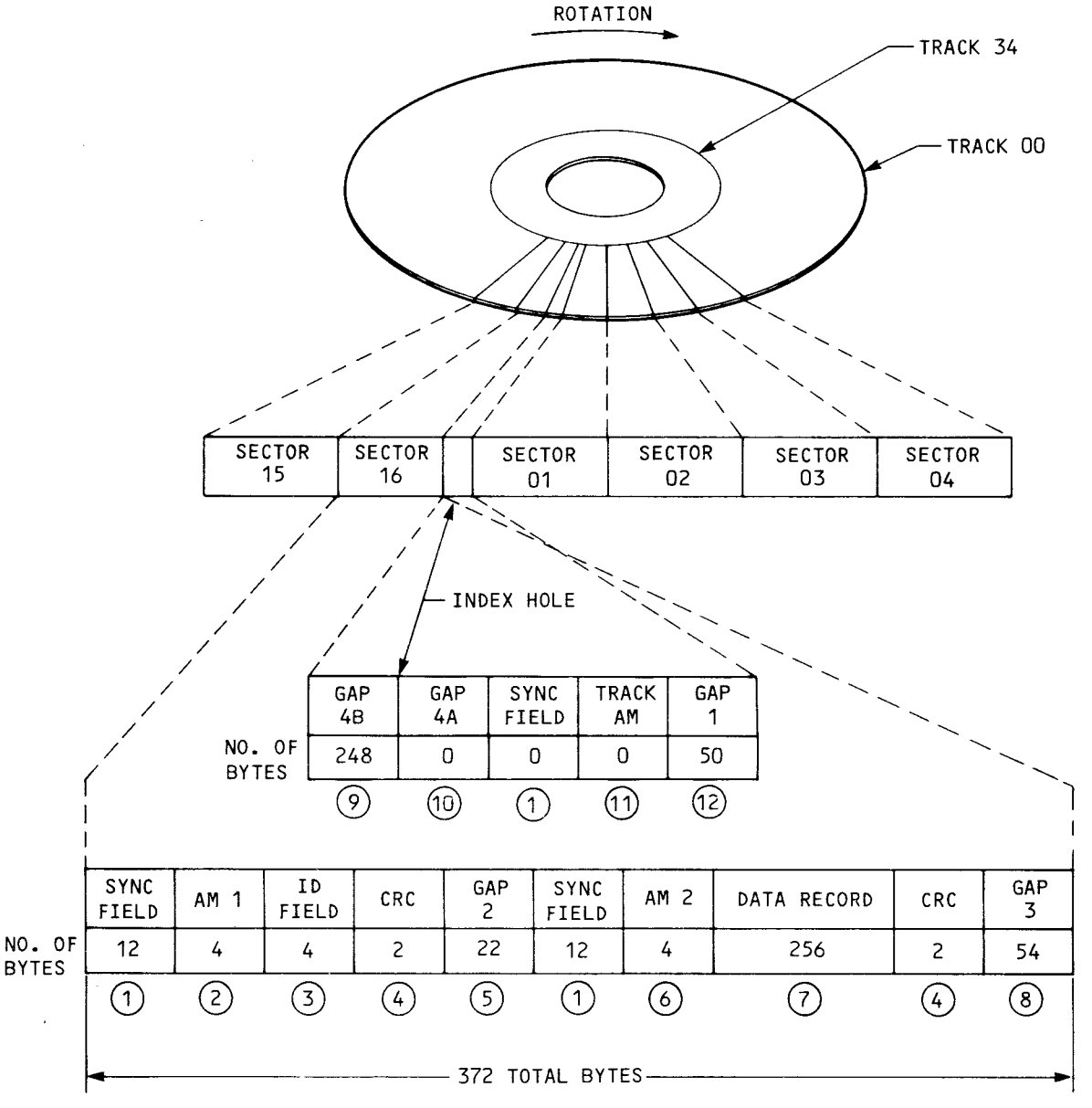


FIGURE C-5. MINI-SIZE, DOUBLE-DENSITY TRACK FORMAT

APPENDIX D

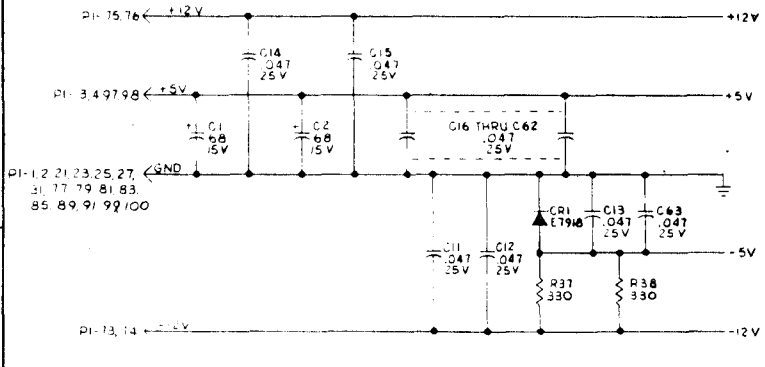
SCHEMATICS

NOTES: UNLESS OTHERWISE SPECIFIED

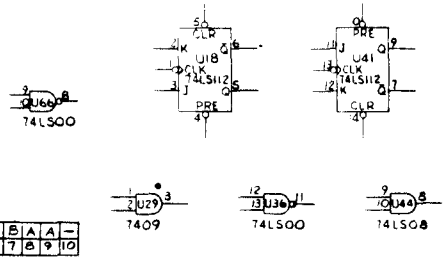
1. ALL CAPACITOR VALUES ARE IN MICROFARADS
2. ALL RESISTOR VALUES ARE IN OHMS
3. ALL RESISTORS TO BE .25 W, 5%
4. SOCKET AT LOCATION U60 IS USED FOR TEST ACCESS. CONNECTIONS ON U60-4 AND U60-10 ARE PROVIDED FOR TEST PURPOSE ONLY
5. COMPONENT LOCATION IS SOCKETED FOR USER'S OPTION
6. TMS 2716 AT POSITIONS U1 AND U2 MAY BE REPLACED WITH TMS 2516 OR TMS 2532 BY CUTTING AND RE-WIRING J4 J5 J6 AND J7

JUMPER CONFIGURATION	
JUMPER	POSITION
J1	E1, E3
J2	E5, E6
J3	E8, E56
J8	E23, E24
J9	E21, E22
J10	E18, E19
J11	E16, E17

REVISIONS			
REV	DESCRIPTION	DATE	APPROVED
A	CN447389 Hanson 10/79	1-18-80	
B	CN454395 Humphrey 2-22-80	2/28/80	



SPARES



REV STATUS OF SHEETS	REV	B	B	-	B	B	A	B	A	A	-
	SH	1	2	3	4	5	6	7	8	9	10

PARTS LIST

QTY	ITEM NO	PART OR IDENTIFYING NUMBER	NOMENCLATURE OR DESCRIPTION	PROCUREMENT SPECIFICATION	NOTES

SEQ NO	IDENT	SPEC	NO	ADDITIONAL	NOTES
	PROCESS			CLASSIFICATION	

PROCESSES — FOR CORRELATION TO GOVT-IND SPECIFICATIONS SEE TI DRAWING 7,2487

1600062	8117
1600061	8117
NEXT ASSY	USED ON
APPLICATION	

UNLESS OTHERWISE SPECIFIED
- DIMENSIONS ARE IN INCHES
- TOLERANCES ANGLES ±1° 3 PLACE DECIMALS ± .010 2 PLACE DECIMALS ± .02
- INTERPRET DRAWING PER MIL-D-1000
- REMOVE ALL BURRS AND SHARP EDGES
- CONCENTRICITY MACHINED DIAMETERS .010 FIM
- DIMENSIONAL LIMITS APPLY BEFORE PROCESSING
- PARENTHETICAL INFO FOR REF ONLY

HOLE TOLERANCE

.013 THRU .124	.004 .001	.125 THRU .250	.005 .001	.251 THRU .500	.006 .001
.501 THRU .750	.008 .001	.751 THRU 1.000	.010 .001	1.001 THRU 2.000	.012 .001

DRAWN		3/23/79
CHECKED		3/6/79
ENGR		8/13/79
		9/13/79
		8/15/79

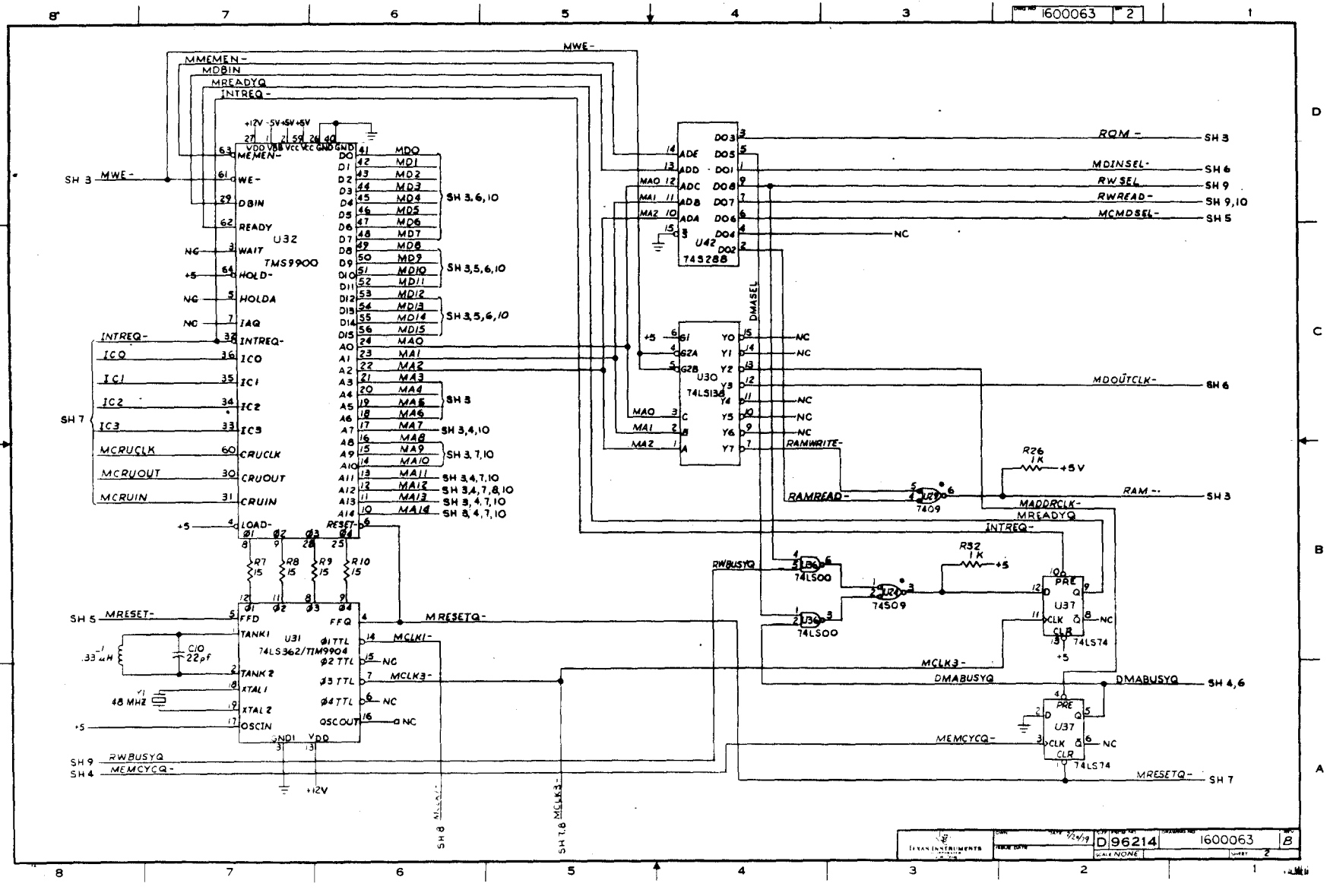
TEXAS INSTRUMENTS INCORPORATED Dallas Texas

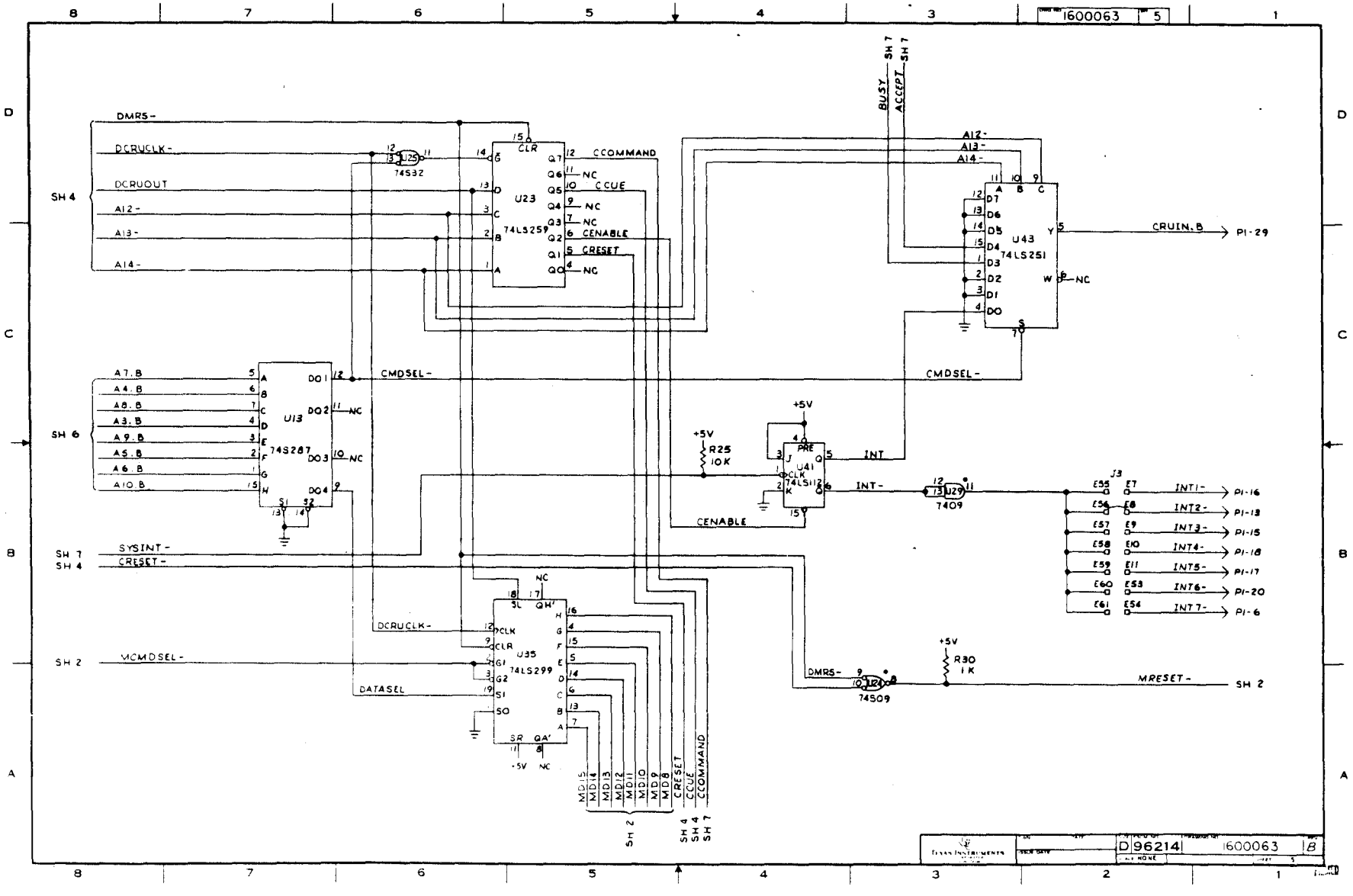
DIAGRAM, LOGIC
TM 990/303

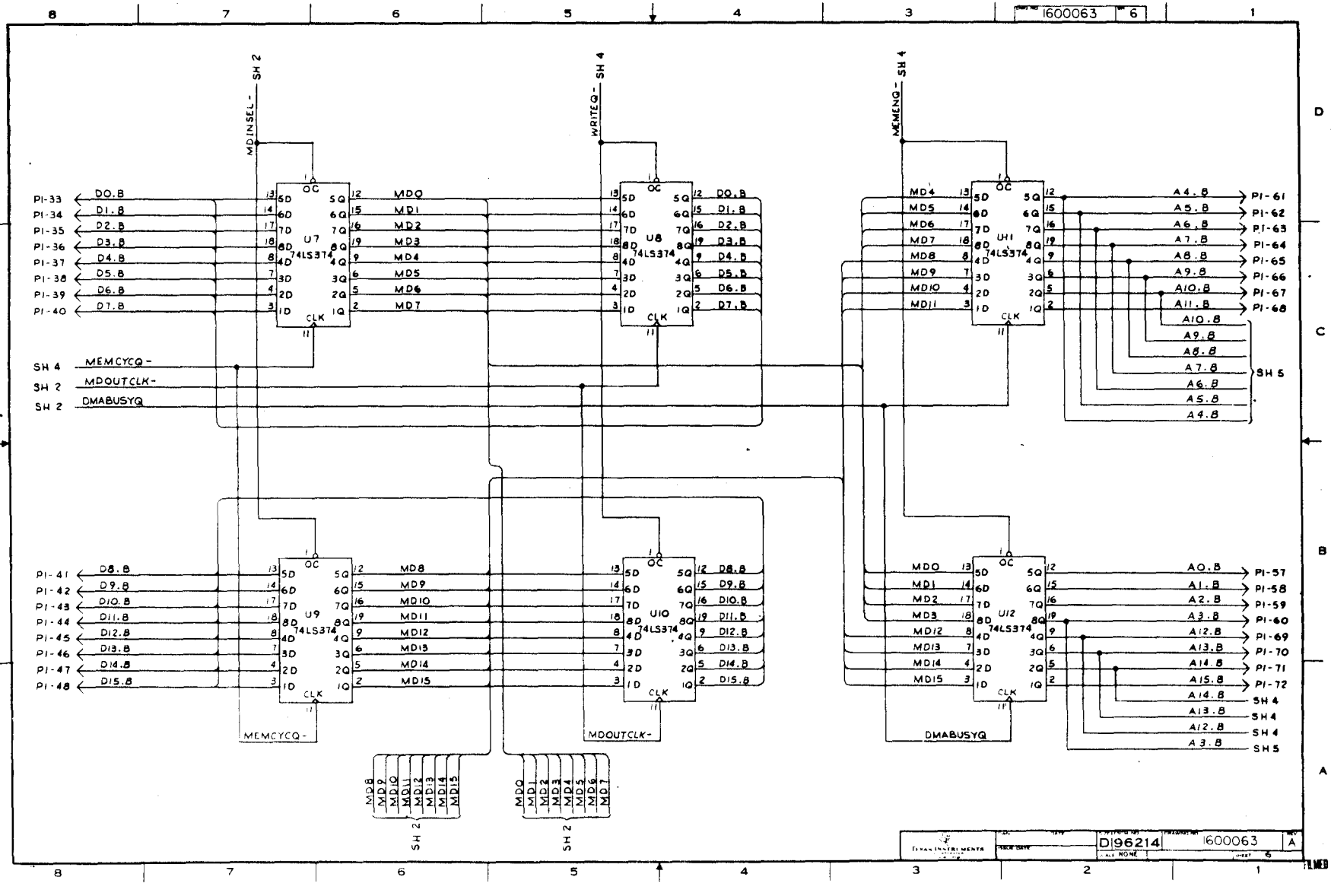
SIZE	CODE IDENT NO	DRAWING NO
D	96214	600063

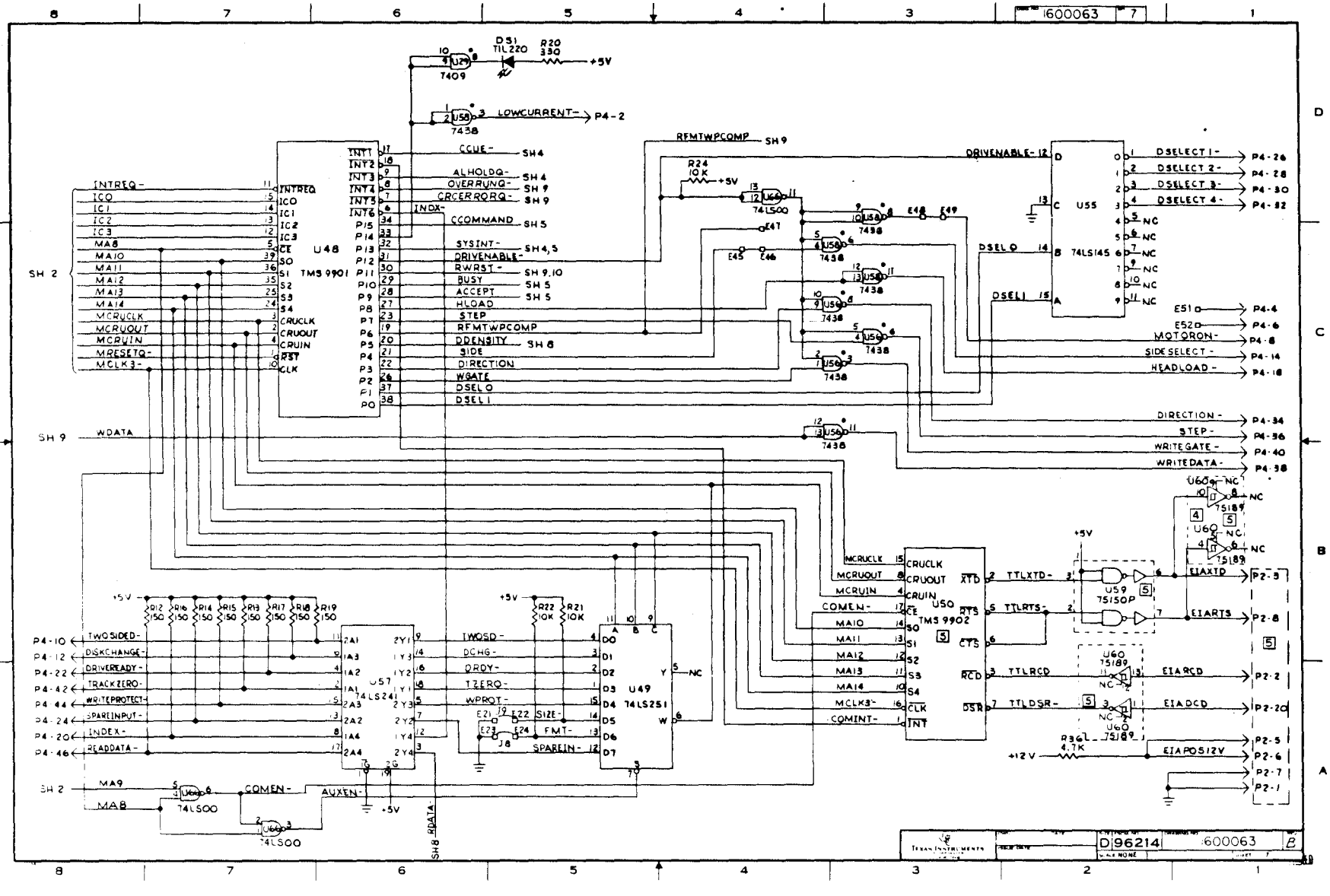
SCALE NONE SHEET 1 OF 10

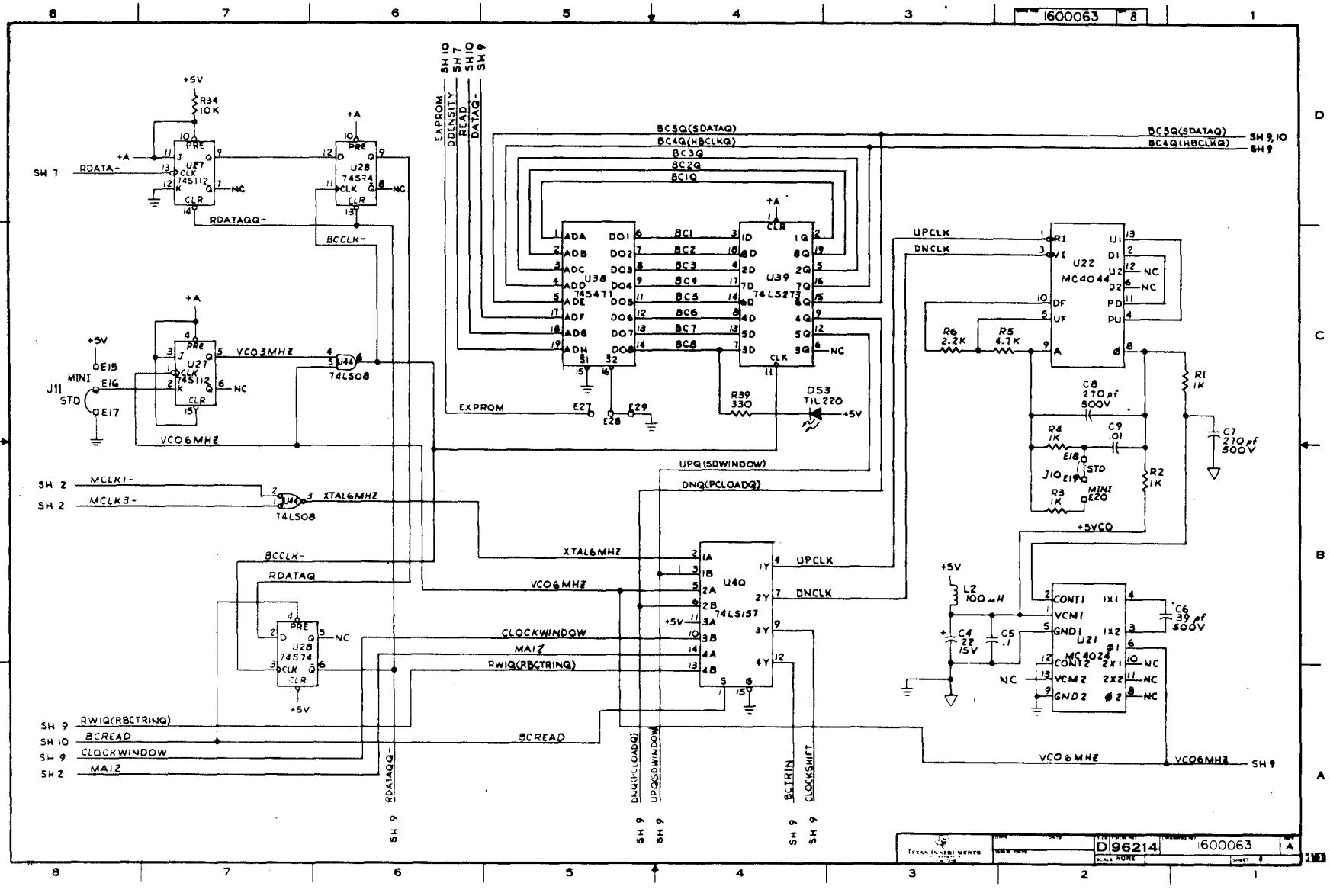
FILMED

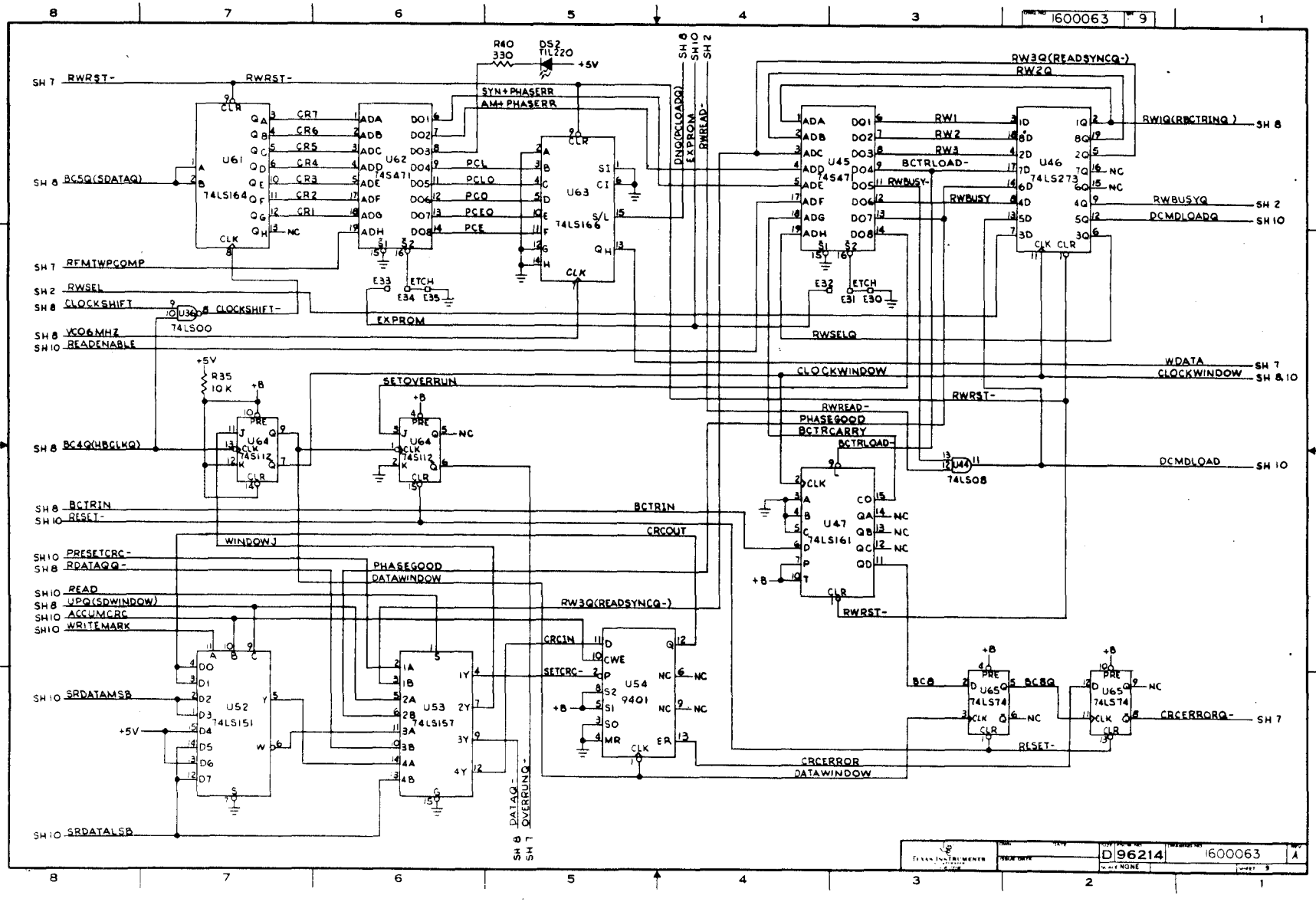

1600063
2
MMEMEN-
MDBIN
MREADYQ
INTREQ-
MWE-
U32
TMS9900
U31
74LS362/TIM9904
U42
74S288
U30
74LS138
DMASEL
RAMWRITE-
RAMREAD-
RAM-
ROM-
MDINSEL-
RWSEL
RWREAD-
MCMDSEL-
MDOUTCLK-
MADDRCLK-
MRESETQ-
MCLK1-
MCLK3-
DMABUSYQ
RWBUSYQ
MEMCYCQ-
MRESET-
R26 1K
R32 1K
74LS00
74S09
74LS74
U37
D96214
1600063
B

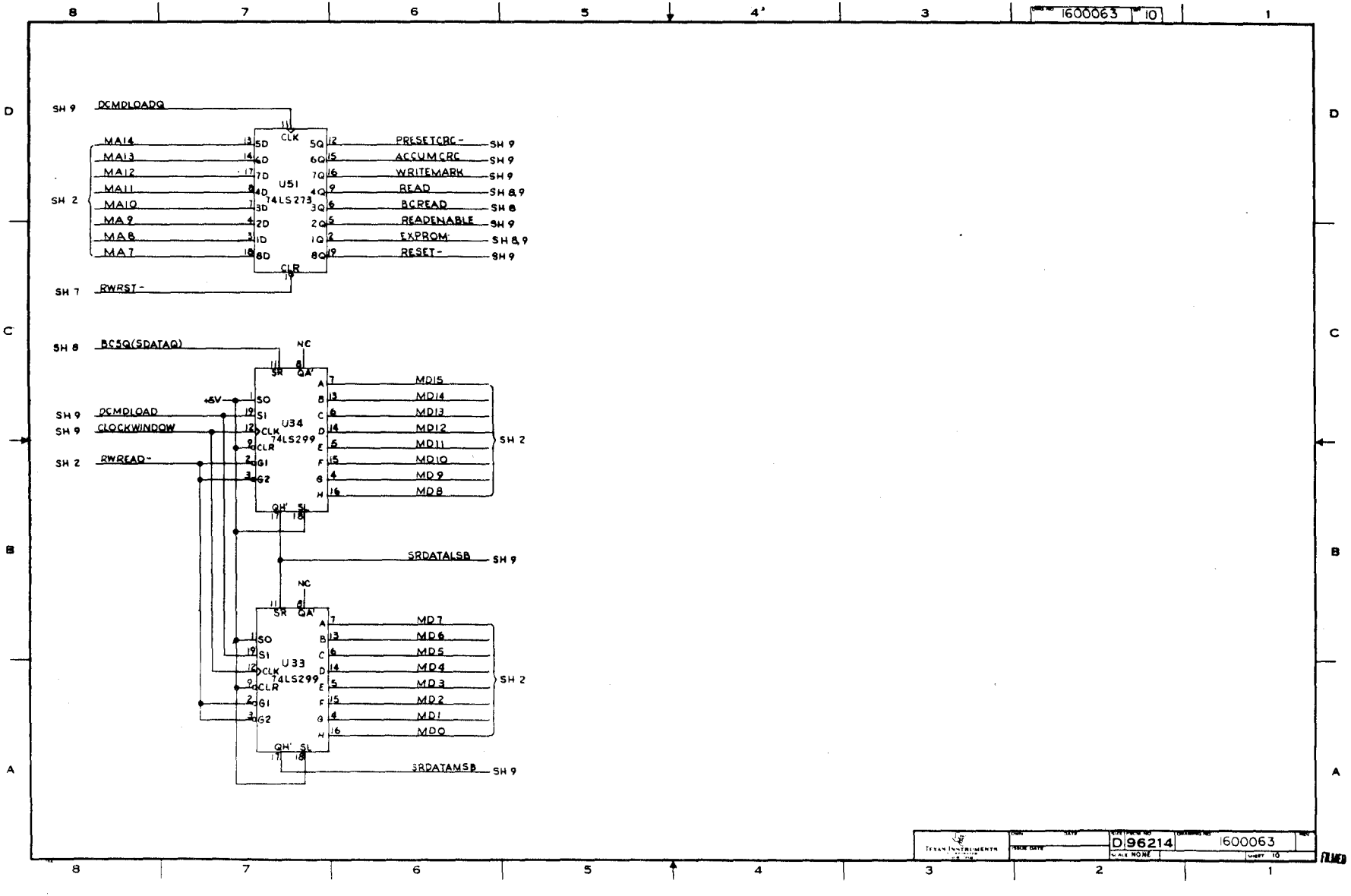

1600063
5
DMRS-
DCRUCLK-
DCRUOUT
A12-
A13-
A14-
SH 4
74S32
U25
U23
74LS259
CLR
CCOMMAND
CCUE
CENABLE
CRESET
NC
BUSY
ACCEPT
SH 7
U43
74LS251
CRUIN.B
PI-29
A7.B
A4.B
A8.B
A3.B
A9.B
A5.B
A6.B
A10.B
SH 6
U13
74S287
CMDSEL-
+5V
R25
10K
U41
74LS112
PRE
INT
INT-
7409
U29
J3
INT1-
INT2-
INT3-
INT4-
INT5-
INT6-
INT7-
PI-16
PI-13
PI-15
PI-18
PI-17
PI-20
PI-6
SYSINT-
CRESET-
SH 7
SH 4
U35
74LS299
DCRUCLK-
MCMDSEL-
SH 2
DATASEL
MD15
MD14
MD13
MD12
MD11
MD10
MD9
MD8
R30
1K
74S09
MRESET-
D96214
1600063
B


1600063
6
MDINSEL- SH 2
WRITEQ- SH 4
MEMENQ- SH 4
U7
U8
U11
U9
U10
U12
74LS374
OC
CLK
PI-33
PI-34
PI-35
PI-36
PI-37
PI-38
PI-39
PI-40
PI-41
PI-42
PI-43
PI-44
PI-45
PI-46
PI-47
PI-48
D0.B
D1.B
D2.B
D3.B
D4.B
D5.B
D6.B
D7.B
D8.B
D9.B
D10.B
D11.B
D12.B
D13.B
D14.B
D15.B
MD0
MD1
MD2
MD3
MD4
MD5
MD6
MD7
MD8
MD9
MD10
MD11
MD12
MD13
MD14
MD15
A4.B
A5.B
A6.B
A7.B
A8.B
A9.B
A10.B
A11.B
A0.B
A1.B
A2.B
A3.B
A12.B
A13.B
A14.B
A15.B
PI-61
PI-62
PI-63
PI-64
PI-65
PI-66
PI-67
PI-68
PI-57
PI-58
PI-59
PI-60
PI-69
PI-70
PI-71
PI-72
SH 4
SH 5
SH 2
MEMCYCQ-
MDOUTCLK-
DMABUSYQ
D96214
1600063
A


1600063
D31 TIL220
R20 330
+5V
7409
LOWCURRENT- P4-2
7438
U48 TMS 9901
INTREQ-
ICO
IC1
IC2
IC3
MA8
MA10
MA11
MA12
MA13
MA14
MCRUCLK
MCRUOUT
MCRUIN
MRESETO-
MCLK3-
SH 2
SH 9
WDATA
CCUE- SH4
ALHOLDO- SH 4
OVERRUNO- SH 9
CRCERRORO- SH 9
INDX-
CCOMMAND SH 5
SYSINT- SH 4,5
DRIVENABLE-
RWRST- SH 9,10
BUSY SH 5
ACCEPT SH 5
HLOAD
STEP
RFMTWPCOMP
DDENSITY SH 8
SIDE
DIRECTION
WGATE
DSEL0
DSEL1
RFMTWPCOMP SH9
R24 10K
74LS00
U55 74LS145
DSELECT1- P4-26
DSELECT2- P4-28
DSELECT3- P4-30
DSELECT4- P4-32
E51 P4-4
E52 P4-6
MOTORON- P4-8
SIDESELECT- P4-14
HEADLOAD- P4-18
DIRECTION- P4-34
STEP- P4-36
WRITEGATE- P4-40
WRITEDATA- P4-38
U60 75189
U59 75150P
U50 TMS 9902
TTLXTD-
TTLRTS-
TTLRCD
TTLDSR-
EIAXTD P2-3
EIARTS P2-8
EIARCD P2-2
EIADCD P2-20
EIAPOS12V P2-5 P2-6 P2-7 P2-1
R36 4.7K
+12V
R12 150
R16 150
R14 150
R15 150
R13 150
R17 150
R18 150
R19 150
TWOSIDED- P4-10
DISKCHANGE- P4-12
DRIVEREADY- P4-22
TRACKZERO- P4-42
WRITEPROTECT- P4-44
SPAREINPUT- P4-24
INDEX- P4-20
READDATA- P4-46
U57 74LS241
R22 10K
R21 10K
TWOSD-
DCHG-
DRDY-
TZERO-
WPROT-
SIZE-
FMT-
SPAREIN-
U49 74LS251
MA9
MA8
COMEN-
AUXEN-
RDATA-
D96214
B


1600063
8
+5V
R34
10K
+A
SH 7
RDATA-
U27
74S112
CLR
NC
RDATAQQ-
U28
74S74
BCCLK-
EXPROM
DENSITY
READ
DATAQ-
SH 10
SH 7
SH10
SH 9
BC5Q(SDATAQ)
BC4Q(HBCLKQ)
BC3Q
BC2Q
BC1Q
U38
74S471
U39
74LS273
DS3
TIL220
R39
330
E27
E28
E29
VCO3MHZ
U44
74LS08
J11
MINI
STD
E15
E16
E17
VCO6MHZ
SH 2
MCLK1-
MCLK3-
XTAL6MHZ
UPQ(SDWINDOW)
DNQ(PCLOADQ)
U40
74LS157
UPCLK
DNCLK
CLOCKWINDOW
MAI2
RWIQ(RBCTRINQ)
BCREAD
CLOCKSHIFT
BCTRIN
U22
MC4044
R6
2.2K
R5
4.7K
C8
270pf
500V
C9
.01
R4
1K
R3
1K
R2
1K
R1
1K
C7
270pf
500V
J10
+5VCO
L2
100μH
C4
22
15V
C5
.1
U21
MC4024
CONT1
VCM1
GND1
CONT2
VCM2
GND2
C6
39pf
500V
VCO6MHZ
SH 9
SH 10
D96214
1600063
A
NONE


1600063
9
SH 7 RWRST-
RWRST-
R40 330
DS2 TIL220
+5V
SYN+PHASERR
AM+PHASERR
SH 8 BC5Q(SDATAQ)
U61 74LS164
U62 74S471
U63 74LS166
SH 7 RFMTWPCOMP
SH 2 RWSEL
SH 8 CLOCKSHIFT
CLOCKSHIFT-
U36 74LS00
SH 8 VCO6MHZ
SH 10 READENABLE
EXPROM
ETCH
E33 E34 E35
DNQ(PCLOAD)
EXPROM
RWREAD-
SH 8
SH 10
SH 2
RW3Q(READSYNCQ-)
RW2Q
U45 74S47
U46 74LS273
RW1
RW2
RW3
BCTRLOAD-
RWBUSY-
RWBUSY
RW1Q(RBCTRINQ)
SH 8
RWBUSYQ
SH 2
DCMDLOADQ
SH 10
E32 E31 E30
RWSELQ
WDATA
SH 7
CLOCKWINDOW
SH 8 10
R35 10K
U64 74S112
SETOVERRUN
SH 8 BC4Q(HBCLKQ)
RWREAD-
PHASEGOOD
BCTRCARRY
BCTRLOAD-
U44 74LS08
DCMDLOAD
SH 10
U47 74LS161
SH 8 BCTRIN
SH 10 RESET-
BCTRIN
CRCOUT
WINDOWJ
SH 10 PRESETCRC-
SH 8 RDATAQ-
PHASEGOOD
DATAWINDOW
SH 10 READ
SH 8 UPQ(SDWINDOW)
SH 10 ACCUMCRC
SH 10 WRITEMARK
RW3Q(READSYNCQ-)
U52 74LS151
U53 74LS157
U54 9401
CRCIN
SETCRC-
SH 10 SRDATAMSB
SH 10 SRDATALSB
DATAQ-
OVERRUNQ-
SH 8
SH 7
BC8
U65 74LS74
BC8Q
CRCERRORQ-
SH 7
RESET-
CRCERROR
DATAWINDOW
D96214
1600063
A
SCALE NONE
SHEET 9


1600063
10
SH 9
DCMPLOADQ
MA14
MA13
MA12
MA11
MA10
MA9
MA8
MA7
SH 2
U51
74LS273
CLK
CLR
5D
6D
7D
4D
3D
2D
1D
8D
5Q
6Q
7Q
4Q
3Q
2Q
1Q
8Q
PRESETCRC-
ACCUMCRC
WRITEMARK
READ
BCREAD
READENABLE
EXPROM-
RESET-
SH 7
RWRST-
SH 8
BCSQ(SDATAQ)
NC
U34
74LS299
SR
QA'
SO
S1
CLK
CLR
G1
G2
QH'
SL
+5V
DCMPLOAD
CLOCKWINDOW
RWREAD-
MD15
MD14
MD13
MD12
MD11
MD10
MD9
MD8
SRDATALSB
U33
MD7
MD6
MD5
MD4
MD3
MD2
MD1
MD0
SRDATAMSB
TEXAS INSTRUMENTS
D96214
1600063
SCALE NONE
SHEET 10

APPENDIX E

PROGRAMMING PROM FOR UNIQUE CRU ADDRESS

E.1 GENERAL

A program in the 74LS287 PROM in socket U13 of the TM 990/303A board is used to specify the CRU base addresses for communication through the Communication Register Unit (CRU) as described in section 3.3. Two base addresses are used, one to transfer address bytes of the Command List (hardware base address 0108_{16} or software base address 0210_{16}) and one to transfer command signals (hardware base address 0110_{16} or software base address 0220_{16}). These addresses are the result of the PROM monitoring address lines A3 to A10 and thus enabling/disabling the data transfer signal (DATASEL) or the command transfer signal (CMDSEL-) by its programmed contents. This means that the user can replace this PROM with another programmed PROM in order to use a different

To Disk Controller	CRU Bit and (Hardware Base Addresses)	From Disk Controller	Displacement From Hardware Base Addr
LSB	Base + 0 (100)	0	
↑	1	0	
Command	2	0	
List	3	0	
Address	4	0	
	5	0	
↓	6	0	
MSB	7	0	
LSB	Base + 8 (108)	0	0
↑	9	0	1
Command	A	0	2
List	B	0	3
Address	C	0	4
	D	0	5
↓	E	0	6
MSB	F	0	7
COMMAND	Base + 10 (110)	0	8
0	11	0	9
CUE	12	0	10
0	13	ACCEPT	11
0	14	BUSY	12
INTERRRUPT ENABLE	15	0	13
RESET	16	0	14
0	Base + 17 (117)	INTERRUPT ISSUED	15
COMMAND	18 (118)	0	
0	19	0	
CUE	1A	0	
0	1B	ACCEPT	
0	1C	BUSY	
INTERRRUPT ENABLE	1D	0	
RESET	1E	0	
0	Base + 1F (11F)	INTERRUPT ISSUED	

FIGURE E-1. CRU ADDRESS SCHEME FOR TRANSFERRING COMMAND LIST ADDRESS AS SHIPPED FROM FACTORY

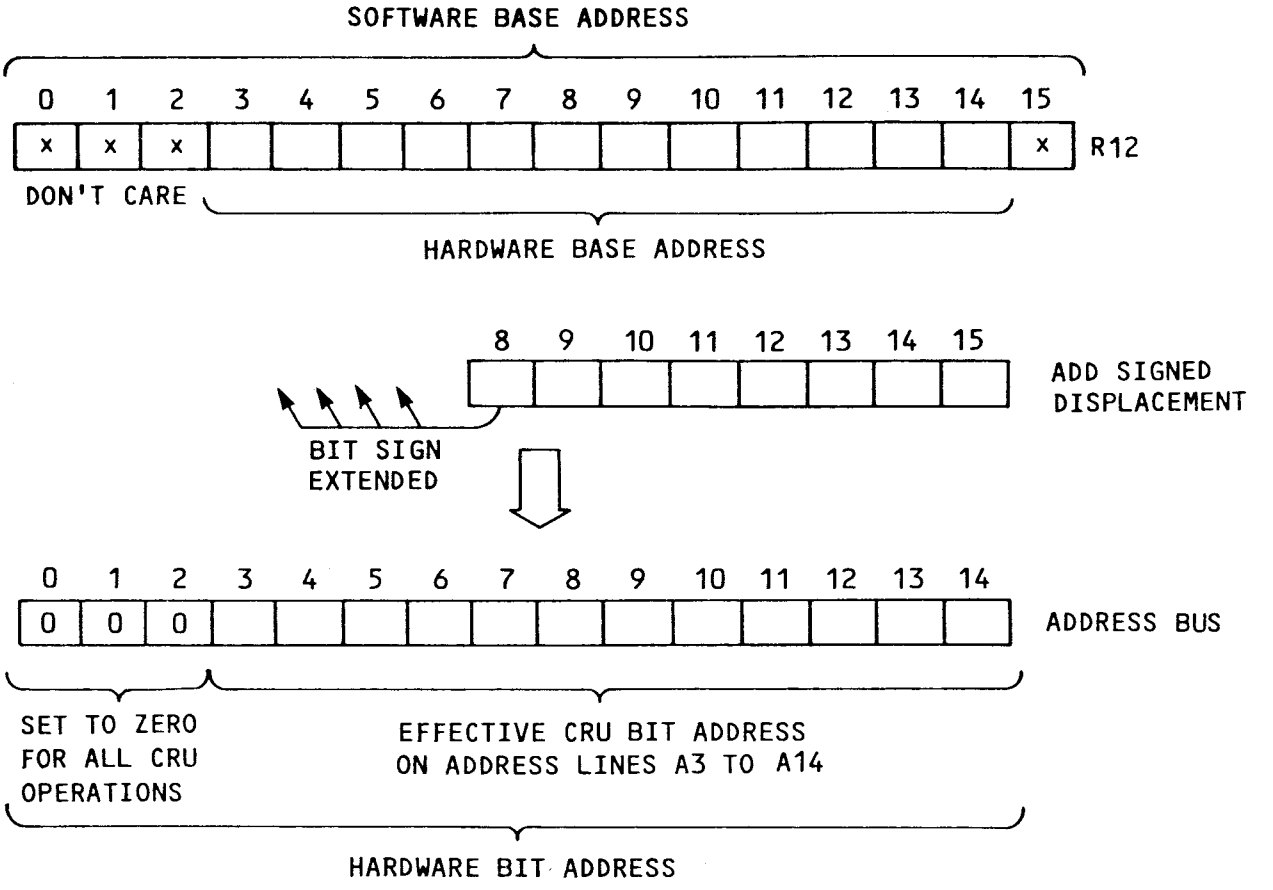


FIGURE E-2. CRU ADDRESS NOMENCLATURE

CRU addressing scheme (the scheme as shipped is shown in Figure E-1). The relationship between the address bus, hardware base address, and software base address is shown in Figure E-2.

Note in Figure E-1 that the first eight CRU bits (at hardware base address 0100_{16}) are the address (data) bytes to be transferred serially to the disk controller. The next eight CRU bits (hardware base address 0108_{16}) are a repeat of the first eight bits. This is the same for the second 16 CRU bits which contain the command data for the address-byte transfer -- the first eight bits (hardware base address 0110_{16}) are repeated in the second eight bits (hardware base address 0118_{16}). The reason for the first eight bits being repeated is that address line A11 is not monitored in this address scheme. Only address lines A3 to A10 are decoded by the PROM at U13; thus, the value on A11 is not taken into consideration for the CRU address. If the value on A3 to A10 remains the same while A11 is toggled, the PROM output remains the same. Because of this, there are 16 bits in the center of the 32-bit scheme that appear for programming ease to be contiguous and which can handle both the data (address bytes) and command transfer through the CRU.

E.2 PROM INPUT/OUTPUT

Input to the PROM consists of the address lines A3 (most significant bit or MSB) to A10 (least significant bit or LSB). Output of the PROM consists of a four-bit "nibble," of which only the MSB (DO4) and the LSB (DO1) are connected to the board logic. The middle outputs (DO2 and DO3) are not connected. Input and output pins on the PROM are shown in Figure E-3.

Data outputs are:

- DO4 is DATASEL for transfer of the data address byte
- DO1 is CMDSEL- for transfer of command bits.

The arrangement of bus address line to PROM address input is not straightforward; that is, the most significant bus address line A3.B is not connected to the most significant PROM address input line H. Because of this, the user must take care in interpreting his address line values corresponding to the internal PROM address. This correspondence of bus address line to PROM address is shown in Figure E-4.

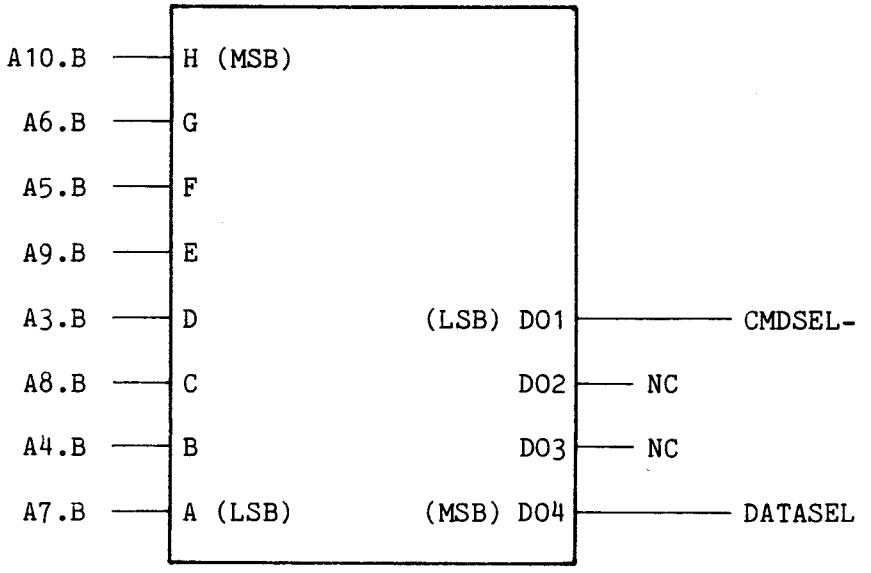


FIGURE E-3. PROM U13 ADDRESS INPUT AND DATA OUTPUT PINS

E.3 REQUIRED DATA OUTPUT

In order to effect correct data transfer and command transfer, the following values must be on DO1 and DO4 (DO2 and DO3 can be treated as "don't cares").

- Select data transfer: DO1 (CMDSEL-) = 1
  DO4 (DATALEL) = 1
- Select command transfer: DO1 (CMDSEL-) = 0
  DO4 (DATASEL) = 0
- All other times: DO1 (CMDSEL-) = 1
  DO4 (DATASEL) = 0

In other words, for data transfer, the nibble output will be $1XX1_2$ (with X as a don't care), for command transfer, the nibble output will be $0XX0_2$, and for all other times the nibble will be $0XX1_2$ (using zeroes for don't cares, these would be respectively the values 9, 0, 1).

E.4 CONSIDERATIONS

The software base address must be a value between 0020_{16} (hardware base address 0010_{16}) and $1FE0_{16}$ (hardware base address $0FF0_{16}$). Note that the nibble in R12 consisting of bits 8 to 11 must be an even value (i.e., bit 11 is not decoded).

Because address line A11.B is not decoded, the user selects a CRU area consisting of 16 CRU bits for each address input, even though only eight bits are to be transferred. This results in the duplication of purpose of the CRU process as shown in Figure E-1. As shown in that figure, a software base address of 0200_{16} or 0210_{16} can be used for transferring the Command List address. Thus a total CRU address space of 32 CRU bits is needed for both the data transfer and the command transfer (each at their own CRU address). As shipped, the data transfer uses CRU software base address 0200_{16} and command transfer uses CRU software base address 0220_{16}. However, for convenience, software base addresses 0210_{16} (for data) and 0220_{16} (for command) can be used for the programmer to make one contiguous CRU address space.

E.5 DETERMINE THE PROM ADDRESSES TO BE CODED

Use the following steps to determine to PROM addresses and their respective codes:

1) Compute the desired CRU hardware base address (in register 12, this is bits 3 to 14) or CRU software base address (all 16 bits).

2) Insert the hardware base address into the 12 blanks of R12 as shown at the top of Figure E-4. Note that only the values in R12 bits 3 to 10 are relevent since only these will be decoded by the PROM address input lines. Do not enter values for R12 bits 0-3 or 11-15.

3) Following the lines between R12 and the PROM, copy the same binary values into the PROM address input boxes. The resulting eight-bit value will be the PROM address at which to program the respective nibble contents as explained in section E.3.

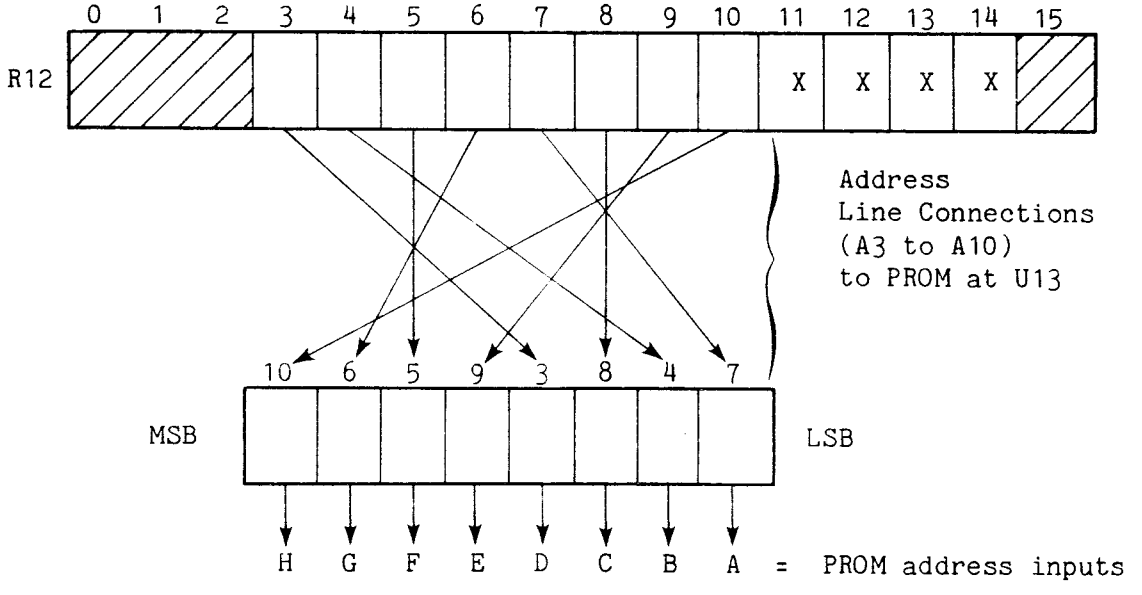


FIGURE E-4. INTERPRETING HARDWARE BASE ADDRESS AS ADDRESS INPUT TO PROM AT U13

E.5 EXAMPLE 1

- Desired software base address for data transfer: 0320_{16}
- Desired software base address for command transfer: 0340_{16}

The results are shown in Figure E-5.

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R12				0	0	0	1	1	0	0	1	X	X	X	X	

Address
Line Connections
(A3 to A10)
to PROM at U13

	10	6	5	9	3	8	4	7	
MSB	1	1	0	0	0	0	0	1	LSB
	H	G	F	E	D	C	B	A	= PROM address inputs

Result: Program PROM address $C1_{16}$ with 1001_2

(a) Determine Data Transfer PROM Nibble Contents (Sftwr Base Addr 0320_{16})

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R12				0	0	0	1	1	0	1	0	X	X	X	X	

Address
Line Connections
(A3 to A10)
to PROM at U13

	10	6	5	9	3	8	4	7	
MSB	0	1	0	1	0	0	0	1	LSB
	H	G	F	E	D	C	B	A	= PROM address inputs

Result: Program PROM address 51_{16} with 0000_2.

(b) Determine Command Transfer PROM Nibble Contents (Sftwr Base Addr 0340_{16})

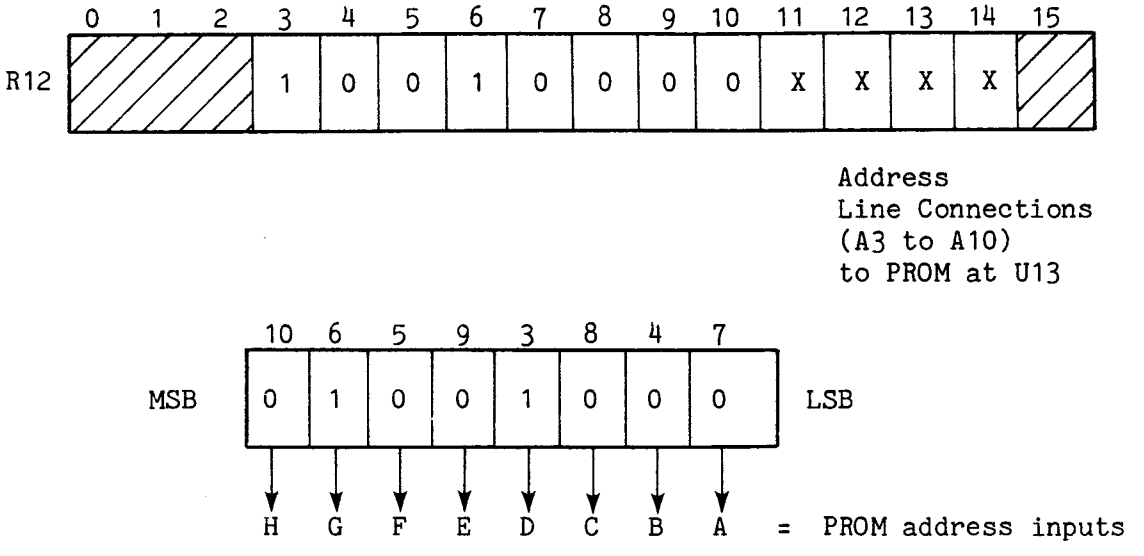
<u>Note: program all other PROM addresses with 0001_2.</u>

FIGURE E-5. EXAMPLE 1 RESULTS

E.6 EXAMPLE 2

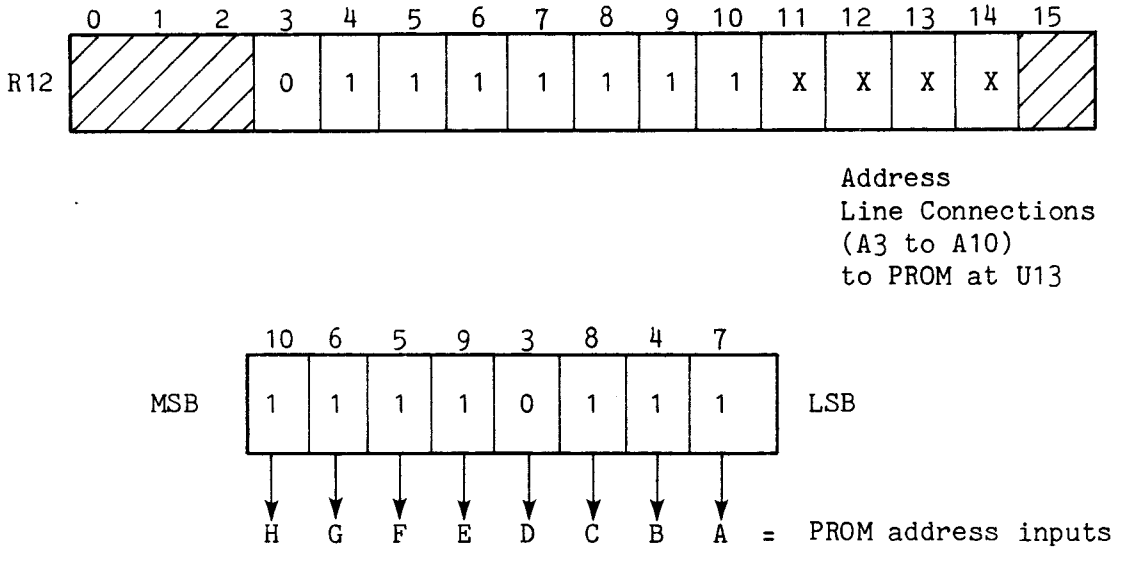
- Desired software base address for data transfer: 1200_{16}
- Desired software base address for command transfer: $0FE0_{16}$

The results are shown in Figure E-6.



Result: Program PROM address 48_{16} with 1001_2

(a) Determine Data Transfer PROM Nibble Contents (Sftwr Base Addr 1200_{16})



Result: Program PROM address $F7_{16}$ with 0000_2.

(b) Determine Command Transfer PROM Nibble Contents (Sftwr Base Addr $0FE0_{16}$)

<u>Note: Program all other PROM addresses with 0001_2.</u>

FIGURE E-6. EXAMPLE 2 RESULTS

APPENDIX F

PIN LIST FOR CONTROLLER-TO-DRIVE CABLES

TABLE F-1. PIN LIST FOR TM 990/527 CABLE FOR MODEL 800 DRIVE

Pin at Connector P4	Pin at Disk Drive	Signal
P4-2	2	LOW CURRENT (Not used by Shugart models)
P4-4	4	E51 (unpopulated jumper pin)
P4-6	6	E52 (unpopulated jumper pin)
P4-8	8	MOTORON-
P4-10	10	TWOSIDED-
P4-12	12	DISKCHANGE-
P4-14	14	SIDESELECT-
P4-18	18	HEADLOAD-
P4-20	20	INDEX-
P4-22	22	DRIVEREADY-
P4-24	24	SPAREINPUT-
P4-26	26	DSELECT1-
P4-28	28	DSELECT2-
P4-30	30	DSELECT3-
P4-32	32	DSELECT4-
P4-34	34	DIRECTION-
P4-36	36	STEP-
P4-38	38	WRITEDATA-
P4-40	40	WRITEGATE-
P4-42	42	TRACKZERO-
P4-44	44	WRITEPROTECT-
P4-46	46	READDATA-

NOTES:

1. All odd-numbered lines are tied to ground. Odd-numbered pins at connector P4 are on the bottom side of the PC board.

2. Note that the dash number at connector P4 is the pin number at the disk drive edge connector.

3. Jumper pins E50, E51, and E52 are <u>not</u> provided on the board as shipped; instead plated through holes are connected to the pins at connector P4. These are provided for future use.

4. Pin 2 at connector P4 is connected to the color-coded ribbon-cable edge.

TABLE F-2. PIN LIST FOR TM 990/535 CABLE FOR MODEL 400 DRIVE

50-Pin[1] Connector P4	34-Pin[2] Output at P5	Signal
P4-8	P5-16	MOTORON-
P4-20	P5-8	INDEX-
P4-26	P5-10	DSELECT1-
P4-28	P5-12	DSELECT2-
P4-30	P5-14	DSELECT3-
P4-34	P5-18	DIRECTION-
P4-36	P5-20	STEP-
P4-38	P5-22	WRITEDATA-
P4-40	P5-24	WRITEGATE-
P4-42	P5-26	TRACKZERO-
P4-44	P5-28	WRITEPROTECT-
P4-46	P5-30	READDATA-

NOTES

1. PC board 1600134-0001 is connected to the 50-pin connector at P4 to provide cross-over wiring to a 34-pin output compatible with the 34-pin connector at the model 400 (mini) disk drive. This interface board is included as part of the 34-wire cable, TM 990/335. P5 is the 34-pin output of this interface board.

2. The dash number at P5 is the corresponding pin number at the disk drive connector. P5 is the 34-pin output of this interface board.

3. Odd-numbered pins are connected to ground only. Those odd-numbered ground pins in the cable are P4-7 (P5-1), P4-11 (P5-3), P4-19 (P5-5), P4-21 (P5-7), and the odd numbers from P4-25 to P4-45 (P5-9 to P5-29) inclusive.

4. Pin 2 at connector P4 is connected to the color-coded ribbon-cable edge.

APPENDIX G

PARTS LIST

Symbol	Description	Qty
C01, C02	Capacitor, 68.0 uFd, 15 V, 10%	2
C03, C04	Capacitor, 22.0 uFd, 15 V, 10%	2
C05	Capacitor, 0.10 uFd, 50 V, 5%, ceramic	1
C06	Capacitor, 39.0 pFd, 500 V, 5%, mica	1
C07, C08	Capacitor, 270.0 pFd, 500 V, 5%, mica	2
C09	Capacitor, 0.010 uFd, 100 V, 10%, ceramic	1
C10	Capacitor, 22.0 pFd, 200 V, 10%, ceramic	1
C11-C63	Capacitor, 0.047 uFd, 500 V, +80%/-20%, axial lead	53
CR1	Diode, silicon zener, 1%, 5V (E7918)	1
DS1-DS3	Optoelectronic device, TIL 220	3
L1	Coil, RF, 0.33 uH, phenolic core	1
L2	Coil, RF, 100 uH, 4.5 ohm, 133 mA	
R01-R04, R23, R26, R28, R30-R32	Resistor, 1.0 kilohm, 5%, 0.25 W	10
R5, R36	Resistor, 4.7 kilohm, 5%, 0.25 W	2
R6	Resistor, 2.2 kilohm, 5%, 0.25 W	1
R07-R10	Resistor, 15 ohm, 5%, 0.25 W	4
R11	Resistor, 68 ohm, 5%, 0.25 W	1
R12-R19	Resistor, 150 ohm, 5%, 0.25 W	8
R20, R37-R40	Resistor, 330 ohm, 5%, 0.25 W	5
R21, R22, R24, R25 R27, R29, R33, R34, R35	Resistor, 10 kilohm, 5%, 0.25 W	9

Symbol	Description	Qty
U01	EPROM, TMS 2716	1
U02	EPROM, TMS 2716	1
U03-U06	Static RAM, TMS 4045 (alternate is 2114)	4
U07-U12, U14	IC. SN74LS374N	7
U13	PROM, CRU decode, 74S287	1
U15	IC, SN74S241N	1
U16	IC, SN74LS240N	1
U17	Network, SN74LS132N	1
U18, U41	Network, SN74LS112N	2
U19, U20	Network, SN74S114N	2
U21	IC, MC4024P	1
U22	IC, MC4044P	1
U23	IC, SN74LS259N	1
U24	IC, SN74S09N	1
U25	Network, SN74S32N	1
U26	Network, SN74S02N	1
U27, U64	Network, SN74S112N	2
U28	Network, SN74S74N	1
U29	Network, SN7409	1
U30	Network, SN74LS138N	1
U31	TIM 9904 four-phase clock generator driver	1
U32	Microprocessor, TMS 9900	1
U33, U34, U35	IC, SN74LS299N	3
U36, U66	Network, SN74LS00N	2
U37, U65	Network, SN74LS74N	2
U38	PROM, data separator, 74S471	1
U39, U46, U51	IC, SN74LS273N	3
U40, U53	Network, SN74LS157N	2
U42	PROM, processor decode, 74S288	1
U43, U49	IC, SN74LS251N	2
U44	Network, SN74LS08N	1
U45	PROM, read/write controller, 74S471	1
U47	IC, SN74LS161N	1
U48	TMS 9901 programmable systems interface	1
U52	Network, SN74LS151N	1
U54	IC, CRC Generator, FCD 9401	1
U55	Network, SN74LS145	1
U56, U58	Network, SN7438N	2
U57	IC, SN74LS241N	1
U61	IC, SN74LS164N	1
U62	PROM, sync/precomp,74S471	1
U63	IC, SN74LS166N	1
Y1	Crystal, quartz, 48 MHz, 0.005 %, 3d overtone (HC-18U)	1

Jumper Plugs These plugs are available from:

Berg Electronic, Inc.
Rt. 3
New Cumberland, Penn. 17070

Berg part number 65474-005

APPENDIX H

DEMONSTRATION SOFTWARE

H.1 GENERAL

Optional TM 990/425 demonstration software is available for verifying the correct operation of the TM 990/303A floppy disk controller. The software is provided on two TMS 2716 EPROM chips which can be plugged into the EPROM memory areas on the TM 990/10X microcomputer board or the TM 990/201 memory board. The demonstration software has two RAM memory requirements: 1) The CPU board must be configured for TIBUG (i.e., RAM at $F800_{16}$-$FFFF_{16}$), and 2) a 4K block of RAM beginning at $F000_{16}$ or lower. This software is executed under the TIBUG monitor and uses the I/O utilities provided by the monitor. An assembly listing of the demo software is provided as part of this appendix.

The software is completely position independent in that it can be plugged into any location on the address map (except those locations which are reserved such as TIBUG workspaces, interrupt vectors, or load vectors). The entry point is the first address occupied by the EPROM module. The 4K block of RAM can start on any 4K memory boundry except those used by TIBUG and the demonstration software.

The TM 990/303A demonstration-software structure is shown in Figure H-1.

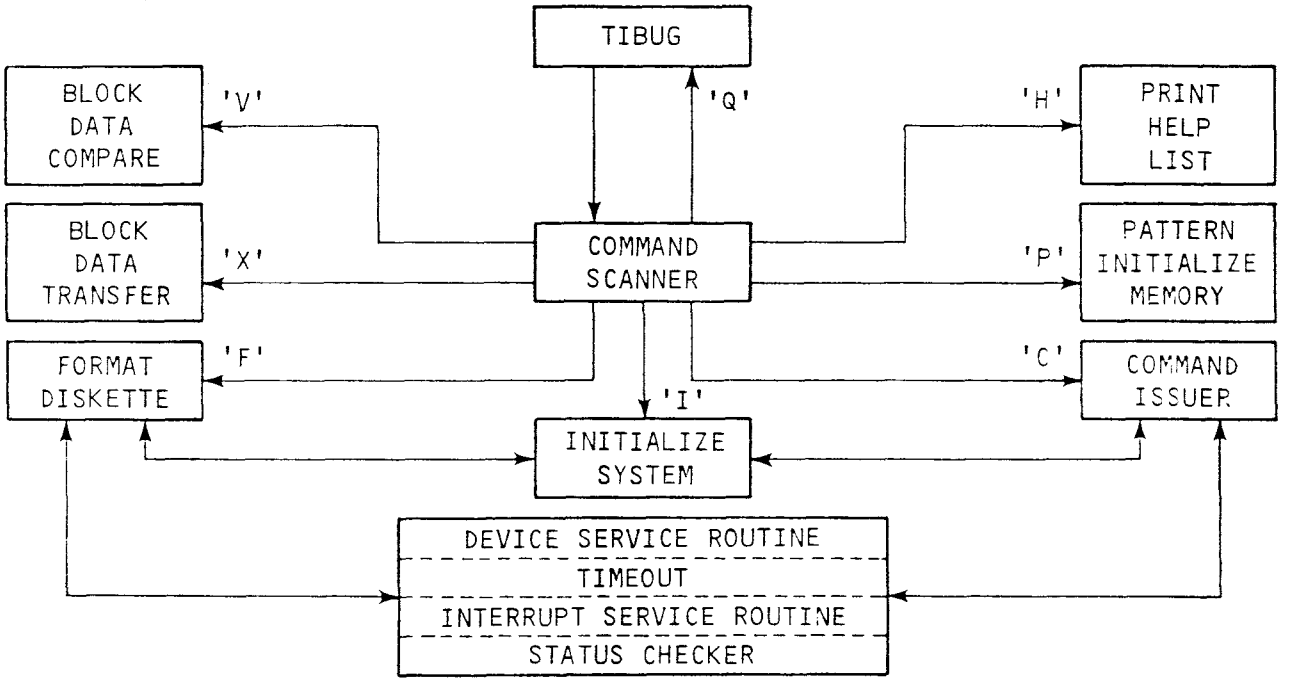


FIGURE H-1. TM 990/303A DEMONSTRATION SOFTWARE STRUCTURE

H.2 INSTALLATION

The EPROM's can be inserted on either the TM 990/10X microcomputer board or on the TM 990/201 memory board.

H.2.1 Installation on Microcomputer Board

a. Turn off power to the system. Remove TM 990/10X board.

b. Remove any EPROM's installed in sockets U43 or U45. Leave the TIBUG EPROM's installed in sockets U42 and U44.

c. Set jumper J2 to 2716.

d. Set jumper J4 to 16 (indicating 2716). Jumper J3 should remain set at 08 (setting for the TIBUG EPROM's).

e. Install the demo EPROM marked U43 in socket U43. Install the demo EPROM marked U45 in socket U45.

f. Install the board into the system and reapply power.

g. Call up the TIBUG monitor (toggle the microcomputer board RESET switch and press the character A on the keyboard).

h. Using the R command, set the Program Counter (P=) to 1000_{16} or to the first location in the demonstration software.

i. Using the E command, execute the software demonstration program.

```
?R
W=FFC6
P=2000 1000
?E

??
```

H.2.2 Installation on the TM 990/201 Memory Board

a. Turn off power to the system. Remove the memory board.

b. On the memory board, place the two TMS 2716 demonstration software chips on adjacent horizontal EPROM sockets (e.g., U56 and U64) with the lowest numbered chip going to the lowest numbered socket (e.g., the U43-marked chip in U56 and the U45-marked chip in U64).

c. Install jumper J2 on the memory board to the SLOW position.

d. Set switches 1 to 4 at S1 to a configuration corresponding to placement of the chips in the memory map. For example, if the sockets U56 and U64 are used (this is EBLK7), then any one of the settings can be used (see the TM 990/201 memory board user's guide). EBLK7 is mapped

into every memory map configuration selected by switches 1 to 4. The only difference is the beginning address. For example, with switches 1-4 set to OFF-ON-ON-ON, EBLK7 starts at address 2000. This being the case, the demonstration software can be executed with the following interaction with TIBUG.

```
?R
W=FFC6
P=01A4  2000
?E
```

H.3 DEMONSTRATION SOFTWARE COMMANDS

When the demonstration software is executed, it outputs an opening message that prompts (??) the user to enter one of the eight one-character commands. The eight commands are explained in the following paragraphs.

The opening banner message is shown below:

```
?E
TM 990/303 DEMO SOFTWARE  REL. 1.1  02/28/80
EPROM AT >2000  DEMO RAM AT >E000
??
```

The location of the EPROM and RAM being used are printed so that the user can compare the listings to the contents of memory.

H.3.1 Help Command (H)

To obtain a list of the eight one character commands observed by the demonstration software, enter the H command. The following list will be output:

```
??H
COMMANDS

H = HELP, PRINT HELP MESSAGE
Q = QUIT, RETURN TO TIBUG
I = INITIALIZE DEMO SOFTWARE, INTERACTIVE USER INITIALIZATION
C = COMMAND ISSUER, USER MAY INTERACTIVELY ISSUE COMMANDS TO THE
    DISK CONTROLLER
X = BLOCK DATA TRANSFER
P = INITIALIZE RAM MEMORY WITH PATTERN
V = MEMORY TO MEMORY DATA COMPARE
F = FORMAT DISKETTE
??
```

H.3.2 Initialize Demo Software Command (I)

This command allows the user to initialize demonstration software parameters. This command will be entered when an 'I' is typed in response to the prompt (??).

The following six questions will be asked with the operator entries shown.

```
??I
UNIT TO BE TESTED (0/1/2/3,DEF=0) ?
SOFTWARE CRU BASE OF /303 (DEF = >200 ) ?
USE MASS STORAGE MODE (Y/N,DEF=Y) ?
USE INTERRUPTS (Y/N,DEF=N) ?
HALT ON ERROR (Y/N,DEF=Y) ?
PRINT ON ERROR (Y/N,DEF=Y) ? Y
??
```

If the operator does not enter one of the acceptable entries (e.g., Y/N, 0/1/2/3) the question will be asked again. There are no default answers for these questions. If the operator does not initialize the system prior to executing the FORMAT or COMMAND ISSUER commands, the initialization questions will be asked. The operator will have to initialize the system every time the demo software in entered from the TIBUG monitor.

H.3.2.1 UNIT TO BE TESTED (0/1/2/3,DEF=0) ?

The unit to be tested refers to the disk drive the demo software will issue commands to in the FORMAT command. The controller is able to talk to a maximum of four (DS1 to DS4 as jumpered) drives provided they are connected to the controller via the daisy chain ribbon cable. If the user specifies a disk drive that is not present, errors will occur during the execution of commands. Units 0 to 3 correspond to DS1 to DS3.

H.3.2.2 SOFTWARE CRU BASE OF /303 (DEF = >200) ?

The host system has a CRU interface to the TM 990/303A board. The default (as shipped) CRU address of the the TM 990/303A board is hexadecimal 200. If the user decides to move the the TM 990/303A board to another position in the CRU map, this question provides a means of testing the board at the new location.

H.3.2.3 USE MASS STORAGE MODE (Y/N,DEF=Y) ?

Data on the floppy disk may be addressed in two modes; mass storage mode, and the physical mode. The answer to this question will determine what questions will be asked when using the COMMAND ISSUER (C) command in the demonstration software.

H.3.2.4 USE INTERRUPTS (Y/N,DEF=N) ?

The use of interrupts is primarily dependent upon the system software. The response to this question will also determine if commands issued via the COMMAND ISSUER will use interrupts. When executing with interrupts the controller will generate an interrupt upon completion of each command. When using the demo software the user should have jumper J3 on the TM 990/303A board set for interrupt 2. The demo software will be able to report whether errors occurred when they were expected. The demo software will also fill the intermediate interrupt vectors with the correct address of the demo software interrupt service routine.

H.3.2.5 HALT ON ERROR (Y/N,DEF=Y) ?

An affirmative answer to this question will cause control to return to the demo command scanner when an error occurrs when a command is sent to the controller. This question affects commands issued to the controller from both the FORMAT and COMMAND ISSUER routines. When running the FORMAT command it is desirable to halt when errors occur. However when using the COMMAND ISSUER and looking at signals with the oscilliscope it may be desirable to continue looping even though an error has occurred.

H.3.2.6 PRINT ON ERROR (Y/N,DEF=Y) ?

An affirmative answer to this question will allow error messages to be printed when an error occurs. When monitoring signals with an oscilliscope it may be desirable to not print messages since this would make synchronization harder.

H.3.3 QUIT (Q)

This command allows the operator to return to the TIBUG monitor.

H.3.4 COMMAND ISSUER (C)

This command allows the user to interactively issue commands to the disk controller. The user may build up to 10 command lists prior to issuing them to the disk controller. The user will be prompted by the demo software to enter all the information required to build up a command list. The unit specified in the Initialize (I) command will not be used in building lists for this command(C). Failure to initialize the system prior to using the Command Issuer, will cause the system initialization questions to be asked. Shown below are the questions as asked by the Command Issuer using both storage modes of data addressing.

Mass storage mode:

```
??C
COMMAND # (0- >10) ? 0
UNIT # (0/1/2/3,DEF=0) ? 0
DATA VERIFY ON READ/WRITE (Y/N,DEF=N) ? N
MSW DISK STORAGE ADDRESS (0- >7FFF) ? 0
LSW DISK STORAGE ADDRESS (0- >FFFE) ? 0
BYTE COUNT ? 0
MEMORY MAP ADDRESS (0- >F,DEF=F) ? F
MEMORY ADDRESS (0- >FFFE) ? 4000
COMMAND # (0- >10) ? Carriage return
CHECK STATUS (Y/N,DEF=Y) ? Y
LOOP THRU COMMAND CHAIN (Y/N,DEF=N) ? N
EXECUTE (Y/N,DEF=Y) ? Y
??
```

Physical storage mode:

```
??C
COMMAND # (0- >10) ? 0
UNIT # (0/1/2/3,DEF=0) ? 0
DATA VERIFY ON READ/WRITE (Y/N,DEF=N) ? N
TRACK NUMBER (0- >4C OR 0- >22) ? 0
SURFACE (0 OR 1, DEF = 0) ? 0
SECTOR (1- >1A OR 1- >10) ? 1
BYTE COUNT ?
MEMORY MAP ADDRESS (0- >F,DEF=F) ? F
MEMORY ADDRESS (0- >FFFE) ? 4000
COMMAND # (0- >10) ? Carriage return
CHECK STATUS (Y/N,DEF=Y) ? Y
LOOP THRU COMMAND CHAIN (Y/N,DEF=N) ? N
EXECUTE (Y/N,DEF=Y) ? Y
??
```

When the demo software types out a prompt(??) after a command has been issued to the controller the user may inspect the command list issued to the controller by looking at memory starting at the RAM memory location printed in the opening banner.

H.3.4.1 COMMAND # (0- >10) ?

The command number refers to one of the 17 commands that may be issued to the disk controller. These commands are discussed in section 3.4.3 of this manual. If a larger value than 10 hexadecimal is entered, the user will be asked the question again. The value is placed in the most significant byte of word 2 of the command list. If a carriage return is entered, the building of command lists cease. If a carriage return is entered as the response to the first time this question is asked, the user will be prompted (??) again. If a carriage return is entered after one or more command lists have been built the 'status, loop, and execute' questions will be asked. The demo software will automatically handle the chain pointers (words 8 & 9) if more than one list is built up.

H.3.4.2 UNIT (0/1/2/3,DEF=0) ?

The unit entered is the disk drive the particular command list will be issued to. This allows the user to issue command lists to different drives when chaining command lists (H.3.4.10). If the value entered is larger than 3, the question will be asked again. The value entered here will be placed in the two least significant bits of word 2 of the command list.

H.3.4.3 DATA VERIFY ON READ/WRITE (Y/N,DEF=N) ?

An affirmative answer to this question makes bit 9 in word 2 of the command list equal to a one. This bit is interrogated by the read and write data commands to compare the data read from or written to the disk to the data in host memory. All other commands ignore this bit. If a 'Y' or 'N' is not entered as the answer, the question will be asked again.

H.3.4.4.1 MSW DISK STORAGE ADDRESS (0- >7FFF) ?

If the user decided to use the mass storage mode this address is the most significant 16 bits of the 31 bit mass storage address used to determine where the head is positioned and the data is transferred to or from. If the value is larger than 7FFF hexadecimal is entered, only the 15 least significant bits will be used as an address.

H.3.4.4.2 TRACK NUMBER (0- >4C OR 0- >22) ?

If the user decides to use the physical storage mode of addressing this question asks for the track number. This track number is the physical track location where the head will be positioned. This value is logically 'OR'ed with hexadecimal 8000 and placed in Word 3 of the command list. The numbers shown in parentheses are the allowable tracks for a standard and mini size diskette respectively using IBM formats.

H.3.4.5.1 LSW DISK STORAGE ADDRESS (0- >7FFE) ?

When using the mass storage mode of addressing, this value is the least significant 16 bits of the 31-bit address that is used to determine head positioning and data transfer address. If a value is entered in which the least significant bit is a 1, it will be ignored because the address needs to be on a word boundary. The user is cautioned that the 31 bit mass storage address must be on a sector boundary.

H.3.4.5.2 SURFACE (0 OR 1,DEF=0) ?

The physical storage mode of addressing requires that the user decide which side of the disk is going to be addressed. The user should be aware of the type and model of drive that is being used. All drives will have a surface 0. This value will be placed in the most significant byte of Word 4 of the command list that is being built by the demonstration software.

H.3.4.5.3 SECTOR (1- >1A OR 1- >10) ?

In the physical storage mode Word 4 also contains the sector number in the least significant byte. When this hexadecimal number is entered it is combined with the surface and placed in Word 4 of the command list. Hexadecimal 1A and 10 are the maximum sectors/track on a standard and mini diskette respectively.

H.3.4.6 BYTE COUNT ?

The byte count refers to the amount of data to bt transferred to or from the disk controller. When specified in a read or write operation, this hexadecimal value should be a multiple of one sectors data length.

H.3.4.7 MEMORY MAP ADDRESS (0- >F,DEF=F) ?

The memory map address is the four most significant bits of the twenty bit address space where data will be transferred to or from by the controller. When the demo software is used with the TM 990/201 memory board, this value should be zero. When used with the TM 990/203 memory board this value should be F. If a map address larger than hexadecimal F is entered, the question will be asked again.

H.3.4.8 MEMORY ADDRESS (0->FFFE) ?

This memory address is the least significant 16 bits of the memory storage address where data will be transferred to or from the controller. The demo software will automatically force the least significant bit to zero (word boundary). The user should be cautioned not to transfer data from the controller to addresses occupied by EPROM and RAM workspace areas used by the demo software and TIBUG.

H.3.4.9 CHECK STATUS (Y/N,DEF=Y) ?

An affirmative answer to this question will cause the status bits in command list words 0 and 1 to be examined. If there are errors and the 'PRINT ON ERROR' questioned was answered 'Y', the error message will be printed. If neither a 'Y' or 'N' is entered, the question will be asked again.

H.3.4.10 LOOP THRU COMMAND CHAIN (Y/N,DEF=N) ?

This question gives the user the choice of executing the built command list(s) only once or over and over. If the lists are executed repeatedly it can only be stopped by halting on an error if an error occurs or by depressing the ESCape key on the users terminal. This looping mechanism allows the user to look at signals with an oscilliscope. If an allowable response(Y/N) is not entered the question will be repeated.

H.3.4.11 EXECUTE (Y/N,DEF=Y) ?

This is the last question the user must answer before the chain of command lists can be executed. If the user answers with a 'N' the user will be prompted(??) and the command list(s) will not be executed. This last question is an escape if an error was made in the building of the command lists.

H.3.4.12 Error Messages.

The TM 990/303A demonstration software will report errors when an unexpected condition exists between the host system and controller, or when an error is reported back from the disk controller. For error messages to be reported on the user terminal, the operator must give an affirmative answer to the 'PRINT ON ERROR (Y/N)?' question asked during initialization (H.3.2.6). Any operation with the controller may be halted if the user gives an affirmative answer to the 'HALT ON ERROR' question asked during initialization (H.3.2.5). By answering negative to both of these questions the user may set up command loops for scoping electrical signals without time between disk commands for error printing. Many error messages do not indicate a malfunction of the disk controller, but could be caused by improper configuration of the system. This should be able to determine if the controller board is operational by examining the LED indicator lights on the edge of the board (see section 2.9 for LED use).

H.3.4.12.1 CONTROLLER WOULD NOT ACCEPT LIST ADDRESS

This error occurs when the host attempts to transfer the address of the command list to the controller via the Communications Register Unit (CRU). If the controller does not respond to this address transfer initiation the above message will be printed on the user's terminal. Prior to the address transfer the controller 'BUSY' signal should indicate a not busy state. Some possible causes of this error are: a) no disk controller board is in the system, b) the CRU address at which the controller responds may different than that used in the device service routine, c) the controller board may be malfunctioning. The user may attempt to correct this state by resetting the controller via the CRU.

H.3.4.12.2 CONTROLLER BUSY- COMMAND NOT ISSUED

This error occurs when the host device service routine attempts to transfer a command list address but the 'CONTROLLER BUSY' signal generated by the disk controller indicates the controller is busy and cannot accept the list address. One possible cause of this error is attempting to initiate a list address transfer while the disk controller is processing another command list. Another possible cause of this error is a disk controller malfunction. The user can attempt to rectify this condition by resetting the disk controller.

H.3.4.12.3 COMMAND DID NOT COMPLETE IN 25 SECONDS

This error occurs when the disk controller fails to turn on the 'OPERATION COMPLETE' bit in Word 0 of the command list in 25 seconds after the list address was transferred to the controller. Most commands should be completed by the disk controller in a few seconds, however 25 seconds is adequate time to allow the disk controller to attempt several retries to overcome error conditions. If the controller does not complete a command in 25 seconds the controller will also be in the busy state and must be reset. One cause of this error is a controller malfunction.

H.3.4.12.4 COMMAND DID NOT INTERRUPT UPON COMPLETION

This error occurs when a command completes but does not interrupt the host. To issue commands which interrupt the host upon completion, the user must answer affirmative to the initialization question 'EXECUTE USING INTERRUPTS' (H.3.2.4). If this is not done this error will never occur. If the user desires to use interrupts, he should make sure all conditions allowing interrupts are set up correctly. This error is an indication of controller malfunction or improper interrupt jumper setup.

H.3.4.12.5 UNEXPECTED INTERRUPT FROM CONTROLLER

This error occurs when an interrupt from the controller occurs but is not expected. If the user does not desire interrupts from the controller he needs to answer the question discussed in the previous section negatively. Again the user should make sure that the conditions for interrupts to occur are not setup. This error is an indication of a disk controller or system configuration malfunction.

H.3.4.12.6 NO RAM EXISTS, CANNOT USE DEMO

This error can occur immediately after the demonstration software is entered from TIBUG. This error identifies an error in the memory map configuration in that the demonstration software could not find a 4K block of memory to use. If this error occurs, the user should turn the power off, extract the CPU and memory boards and recheck the jumpers and switch configurations. This error does not indicate any problem with the controller or disk drive.

H.3.4.12.7 BAD EPROM, CANNOT USE DEMO

This error can also occur immediately after the demonstration software is entered from TIBUG. This error indicates that the contents of the demonstration software EPROMs are incorrect. The contents of the EPROMs are verified by performing a checksum on the contents of the EPROM. When this occurs the user should not use the demonstration software because the results will be unpredictable.

H.3.4.12.8

```
*** ERROR UNIT: X  COMMAND: XX
HOST ADDR: XXXXX  STORAGE ADDR: XXXXXXXX  LENGTH: XXXX  DRIVE STATUS: XX

BITS:
OC  ER  IE  DE  ID  OR  SE  UE  OL  WP  SI  ST  BC
 X   X   X   X   X   X   X   X   X   X   X   X   X
```

This error message is returned to the user when an error occurs when using the "C" command to issue commands to the disk controller. This error message will only be printed if the "PRINT ON ERRORS" prompt was answered affirmatively during initialization. The first line of the message tells the user what disk drive is being addressed and what command was issued to it. This is a great deal of help if several commands are chained together which deal with more than one unit. The data for this line is extracted from word 2 of the command list. The second line shows the source or destination address in the host memory where data is written to or from. The host address is extracted from words 6 and 7 of the command list. The second line also shows the disk storage address for the attempted command. This is the contents of words 3 and 4 of the command list. The length shown represents the amount of actual data transferred. This is the value returned to the controller in word 5. The drive status on the second line is returned by the controller and represents the lines of the disk drive as read by the controller (see section 3.4). The last three lines of this error message are a breakout of the error bits in words 0 and 1 of the command list which are returned from the controller. Section 3.4 provides an in-depth discussion of these error bits. The value of the error bits (0 or 1) will appear direcly below the error bit synonyms.

H.3.5 FORMAT DISKETTE (F)

This command allows the user to format diskettes on the drive specified during initialization. This command will only format single sided diskettes. If the initialization questions have not been answered prior to entering this command, they will be asked. Shown below are the prompts for the format diskette command.

```
??F
NUMBER OF SIDES (1 OR 2,DEF=1) ?
SIZE OF DISKETTE (8 OR 5 IN,DEF=8) ?
FORMAT: 0 = IBM SINGLE, 1 = IBM DOUBLE, 2 = TI DOUBLE (DEF=0)  ?
```

With the information gained from initialization and the above prompts command lists will be built to define the drive parameters and format the diskette.

H.3.5.1 NUMBER OF SIDES (1 OR 2,DEF=1) ?

This question refers to the number of usable sides on the diskette to be formatted. The combination of two sides and mini diskettes is illegal, and the user will be prompted again.

H.3.5.2 SIZE OF DISKETTE (8 OR 5 IN,DEF=8) ?

The answer to this question tells the demonstration software whether a mini (5 in.) or standard (8 in.) diskette is being used. This information also defines the number of tracks to be formatted. If a value other than 5 or 8 is entered, the prompt will be asked again.

H.3.5.3 FORMAT: 0 =IBM SINGLE, 1 = IBM DOUBLE, 2 = TI DOUBLE (DEF=0) ?

The type of format specified determines how the disk is addressed, the amount of data per sector, and the storage capacity of the diskette. If a value larger than 2 is entered this prompt is asked again. The diskette size of 5 inches and the TI DOUBLE format is an illegal combination and will cause both prompts to be asked again. After both prompts have been answered the demonstration software will attempt to format the diskette.

H.3.6 BLOCK TRANSFER ROUTINE (X)

This command is entered from the demonstration software command scanner by entering the character 'X'. This command allows the user to move a block of data from one memory location to another. The user will be prompted for the source address, destination address, and the number of bytes to be transferred. The user should terminate all entries with a space character. All entries are hexadecinal and must be an even number. Shown below are the prompts for this command.

```
??X
SOURCE = XXXX  DEST = YYYY  LENGTH = LLLL
??
```

The user should terminate all entries with a space character.

H.3.7 INITIALIZE RAM MEMORY WITH PATTERN (P)

This command is entered from the demonstration software command scanner by entering the character 'P'. This command allows the user to initialize memory with a data pattern. This command is useful for initializing memory before read and write operations. The user will be prompted for a memory address, byte count, and data pattern. All entries are in hexadecimal and are terminated with a space character. The start address and byte count must be even numbers. Shown below are the prompts for this command.

```
??P
ADDR1 = XXXX  LENGTH = LLLL  PATTERN = PPPP
??
```

H.3.8 MEMORY TO MEMORY DATA COMPARE (V)

This command is entered from the demonstration software command scanner by entering the character 'V'. This command allows the user to verify the contents of two blocks of memory by comparing the contents of the two blocks on a word by word basis. If two corresponding memory locations do not have the same contents an error message is printed. This command is very useful in comparing data after write and read operations. The user will be prompted for the starting addresses of the two blocks of data and the number of bytes to be compared. All entries are in hexadecimal and terminated by the space character. Shown below are the prompts for this command and the error print out.

```
??V
ADDR1 = XXXX  ADDR2 = YYYY  LENGTH = LLLL

**ERROR
ADDR1 = XX42  DATA = ABCD  ADDR2 = YY42  DATA = ABCC
??
```

H.4 USING THE DEMO SOFTWARE'S DEVICE SERVICE ROUTINE

The user may take advantage of the demonstration software's Device Service Routine (DSR) by calling it as a subroutine. This enables the user to test his own software with the floppy disk controller without having to write his

own DSR. By using the demo software's DSR the status checker and interrupt service routines may be utilized. To use the DSR the user must first initialize all the variables which the demo software manipulates during entry and the Initialize (I) command. These variables are:

- ROMBAS — Start address of the EPROM
- RAMBAS — Start address of the DEMO RAM
- BAS303 — Base CRU address of the TM 990/303A
- UNIT — Don't care
- INITFL — Determine if interrupts are to be used
- HALTFL — Halt if an error is encountered
- PRNTFL — Print if an error is encountered
- STORFL — Don't care
- UNITFL — Will insert own unit

This initialization can be done by entering the demo software and executing the Initialize (I) command. To interface to the DSR, the user's calling routine must be set up in a certain manner. The requirements are:

- R0 must contain the address of the command list
- R9 must contain the demo software's RAM base
- A subroutine vector must be defined with a workspace and program counter.

Shown below is the initialization process and a code segment which could use the DSR.

Initialize Segment

```
?R
W= FFC6
P= 01A4 1000
?E
TM990/303 DEMO SOFTWARE REL. 1.1 02/28/80
EPROM AT >1000  DEMO RAM AT >F000
??I
UNIT TO BE TESTED (0/1/2/3,DEF=0) ? 0
SOFTWARE CRU BASE OF /303 (DEF= >200) ? 200
USE MASS STORAGE MODE (Y/N,DEF=Y) ? N
USE INTERRUPTS (Y/N,DEF=Y) ? N
HALT ON ERROR (Y/N,DEF=Y) ? Y
PRINT ON ERROR (Y/N,DEF=Y) ? Y
??Q

?R
W=  FFC6
P=  1000 2000      Enter start address of user's code
?E
```

User's Code Segment

```
COMISR EQU  >15A4          Define address of 'COMISR'
RAMBAS EQU  >F226          Define address of RAMBAS
       ----
ENTRY  ----
       ----
       ----
       LI   R0,CMDLST      Get command list address
       MOV  @RAMBAS,R9     Get base of demo RAM
       ----
       BLWP @EXCMD         Go execute command
       ----
       ----
CMDLST BSS  20             Command list
WPCODE BSS  32             Workspace for DSR
EXCMD  DATA WPCODE,COMISR ** The address of 'COMSIR'
                           should be taken from the
                           demo software listing.
       END
```

Note: The user should be cautioned not to use the same RAM memory area that the DEMO Software uses.

```
                IDT  '303 DEMO'
        ********************************************************
        *                RAM SYSTEM DEFINITION                 *
        *                                                      *
        *  This module contains all the RAM definitions for the *
        *  TM990/303 floppy disk demo software.                *
        *                                                      *
        ********************************************************
        *
        *======> BUFFER DEFINITIONS
        *
0000    RAMST  EQU  0000        START OF DEMO RAM DEFINITION
0000    COMBUF EQU  RAMST       BUFFER COMMANDS ISSUED FROM
00C8    BUF    EQU  COMBUF+200  BUFFER COMMMANDS ARE BUILT IN
        *
        *======> WORKSPACE DEFINITIONS
        *
0190    WP2    EQU  BUF+200     1ST DYNAMIC WORKSPACE
01B0    WP3    EQU  WP2+32      2ND DYNAMIC WORKSPACE
01D0    WP4    EQU  WP3+32      3RD DYNAMIC WORKSPACE
01F0    WP5    EQU  WP4+32      4TH DYNAMIC WORKSPACE
        *
        *======> FLAG DEFINITIONS
        *
0210    INITFL EQU  WP5+32      INITIALIZE FLAG, 1234 = YES
0212    INTFLG EQU  INITFL+2    INTERRUPT FLAG
0214    INTOCC EQU  INTFLG+2    INTERRUPT OCCURRED FLAG
0216    HALTFL EQU  INTOCC+2    HALT ON ERROR FLAG
0218    PRNTFL EQU  HALTFL+2    PRINT ON ERROR FLAG
021A    ERRFLG EQU  PRNTFL+2    SYSTEM ERROR FLAG
021C    INTEXP EQU  ERRFLG+2    INTERRUPT EXPECTED FLAG
021E    LOOPFL EQU  INTEXP+2    LOOP ON CHAIN FLAG
0220    UNITFL EQU  LOOPFL+2    DETERMINES IF INIT UNIT INSERTED
0222    STATFL EQU  UNITFL+2    STATUS CHECK FLAG
FD20    EXAREA EQU  WP1+32      EXECUTABLE AREA
        *
        *======> GENERAL PURPOSE VARIABLES
        *
0224    ROMBAS EQU  STATFL+2    BASE OF EPROM
0226    RAMBAS EQU  ROMBAS+2    BASE OF RAM
0228    BAS303 EQU  RAMBAS+2    CRU BASE OF /303 BOARD
022A    UNIT   EQU  BAS303+2    UNIT UNDER TEST
022C    STORFL EQU  UNIT+2      STORAGE MODE FLAG
        *                       0 = PHYSICAL, NOT 0 = MASS
FD00    WP1    EQU  >FD00       ONLY KNOWN WORKSPACE
        *
        *======> XOP DECLARATIONS
        *
                DXOP PRINT,14   MESSAGE PRINT
                DXOP INCHAR,11  INPUT A CHARACTER
                DXOP HEXOUT,10  OUTPUT A HEX #
                DXOP HEXIN,9    INPUT A HEX #
                DXOP CH1OUT,8   OUTPUT 1 HEX XHARACTER
```

```
                ****************************************************
                *              VALUE EQUATES                       *
                ****************************************************
000A  CHNCTR EQU  10          MAX # OF COMMAND LISTS
0010  MAXCMD EQU  >10         MAX CMD CODE
0003  MAXUNT EQU  3           MAX UNIT #
0040  VFYBIT EQU  >40         VERIFY BIT
000F  MAXMAP EQU  >F          MAX MAP ALLOWED
800F  CHNBIT EQU  >800F       CHAIN BIT IN WORD 8 WITH MAP >F
FFF8  MINUS8 EQU  -8          NEG 8
0008  MAXWDS EQU  8           # WORDS IN A CMD LIST TO BE MOVED
000C  BUSY   EQU  >C          CRU BUSY BIT
000B  ACCEPT EQU  >B          CRU ACCEPT BIT
0008  COMAND EQU  >8          CRU COMMAND BIT
000A  CUE    EQU  >A          CRU CUE BIT
000D  INTBIT EQU  >D          INTERRUPT ENABLE BIT TO /303
0100  CRU01  EQU  >100        CRU ADDR OF TMS9901
000E  CLKINT EQU  >E          TMS9901 CLOCK INTERRUPT VECTOR + 2
0460  BRANCH EQU  >460        UNCONDITIONAL BRANCH OPCODE
0064  TICKCT EQU  100         25 SECONDS
0003  CLKBIT EQU  3           CLOCK INTERRUPT BIT
0000  INTCLK EQU  0           CLOCK/INT MODE BIT
0002  INT303 EQU  2           /303 INTERRUPT BIT
0080  INTEN  EQU  >80         INTERRUPT ENABLE BIT
000A  FLPVEC EQU  >A          FLOPPY INTERRUPT VECTOR + 2
0010  WAIT   EQU  >10         WAIT VALUE
0003  THREE  EQU  3           HEX 3
E05B  STAT1  EQU  >E05B       PRIMARY STATUS BITS
AB00  STAT2  EQU  >AB00       SECONDARY STATUS BITS
0001  ONE    EQU  1           VALUE 1
0003  HEX3   EQU  3           HEXADECIMAL 3
5900  YES    EQU  'Y'*256     ASCII Y
4E00  NO     EQU  'N'*256     ASCII N
0140  MAXCRU EQU  >140        LOWEST ALLOWABLE CRU ADDRESS
0200  DEFCRU EQU  >200        DEFAULT CRU BASE
045B  RETURN EQU  >045B       RETURN=B *R11 OPCODE
FFFE  MINUS2 EQU  -2          VALUE NEG 2
0080  TIBUG  EQU  >80         TIBUG ENTRY ADDRESS
000D  CR     EQU  >D          CARRIAGE RETURN
000A  LF     EQU  >A          LINE FEED
0007  BELL   EQU  7           BELL CODE
AAAA  PATTRN EQU  >AAAA       RAM PATTERN
F000  RAMBEG EQU  >F000       FIRST POSSIBLE RAM ADDR
1B00  ESCAPE EQU  >1B00       ASCII ESCAPE
2000  SPACE  EQU  >2000       ASCII SPACE
0D00  CARRET EQU  >0D00       ASCII CARRIAGE RETURN
0022  MINMAX EQU  34          MAX TRACK FOR MINI
004C  STDMAX EQU  76          MAX TRACK FOR STD
CDF5  MOV1   EQU  >CDF5       OPCODE FOR 'MOV *R5+,*R7+'
CD47  MOV2   EQU  >CD47       OPCODE FOR 'MOV *R7,*R5+'
```

```
                 ******************************************************
                 *                   COMMAND SCANNER                  *
                 *                                                    *
                 *  This module is entered via TIBUG. An example is shown  *
                 *  below. This module handles the self relocation of      *
                 *  EPROMs, the RAM search, input from the user, prints the *
                 *  'HELP' list, and return to TIBUG.                      *
                 *                                                    *
                 *                                                    *
                 *  Entry: ? R (cr)                                    *
                 *         W = (space)                                 *
                 *         P = (start address of EPROM)(cr)            *
                 *         ? E                                         *
                 *                                                    *
                 ******************************************************
0000 0300        SCAN00 LIMI 0            SHUT DOWN INTERRUPTS
     0002 0000
0004 02E0               LWPI WP1          SET UP WORKSPACE
     0006 FD00
0008 04C5               CLR  R5           CLEAR THE ACCUMULATOR
000A 020A               LI   R10,SCAN00   LOOK AT START ADDRESS
     000C 0000'
000E 1302               JEQ  SCAN01       IF NOT, BIASED, JUMP
0010 04CA               CLR  R10          DEBUG-LOADER HAS BIAS, CLEAR BASE
0012 1012               JMP  SCAN03       GO TO PROGRAM
0014 020A        SCAN01 LI   R10,RETURN   SET UP 'RT' OPCODE
     0016 045B
0018 068A               BL   R10          PC VALUE TO R11
001A 022B        SCAN02 AI   R11,SCAN00-SCAN02 PC-10 = EPROM START
     001C FFE6
001E C28B               MOV  R11,R10      R10 = BASE VALUE FOR WP
                 *
                 *======> DO EPROM CHECKSUM
                 *
0020 C04A               MOV  R10,R1       GET START OF EPROM
0022 C08A               MOV  R10,R2       CALCULATE END OF EPROM
0024 0222               AI   R2,ENDEMO    ADD IN LENGTH OF CODE
     0026 0FF8'
0028             SCAN09
0028 A171               A    *R1+,R5      ADD IN EPROM VALUE
002A 8081               C    R1,R2        ARE WE DONE ?
002C 16FD               JNE  SCAN09       IF NO, ADD AGAIN
002E C145               MOV  R5,R5        WAS THE SUM 0 ?
0030 1303               JEQ  SCAN03       IF YES, GO ON
0032 2FAA               PRINT @BADROM(R10) PRINT BAD ROM MSG
     0034 0294'
0036 1045               JMP  QUIT         GO BACK TO TIBUG
                 *
                 *======> INITIALIZE THE /203 PARITY
                 *
0038 020C        SCAN03 LI   R12,CRU01    GET 9901 CRU BASE
     003A 0100
003C 1E17               SBZ  23           SET UP /203 RESET
003E CD55        SCAN10 MOV  *R5,*R5+     DO READ/WRITE
0040 0285               CI   R5,WP1       ARE WE DONE ?
     0042 FD00
0044 16FC               JNE  SCAN10       IF NO, JUMP
0046 1D17               SBO  23           RESET /203
                 *
                 *======> DO RAM SEARCH
```

```
0159                *
0160 0048 020C          LI   R12,PATTRN   SET UP PATTERN
     004A AAAA
0161 004C 0209          LI   R9,RAMBEG    SET UP 1st RAM ADDR
     004E F000
0162 0050 C64C  SCAN07 MOV  R12,*R9       SET UP PATTERN
0163 0052 0705          SETO R5           PUT BUS ALL ONE WAY
0164 0054 864C          C    R12,*R9      IS THE PATTERN THERE ?
0165 0056 1306          JEQ  SCAN06       IF YES, JUMP
0166 0058 0229          AI   R9,->1000    SUBTRACT >1000
     005A F000
0167 005C 16F9          JNE  SCAN07       IF NOT AT ZERO, JUMP
0168 005E 2FAA          PRINT @NORAM(R10) PRINT THE ERROR MESSAGE
     0060 0273'
0169 0062 102F          JMP  QUIT         GO TO TIBUG
0170 0064 CA49  SCAN06 MOV  R9,@RAMBAS(R9) SAVE THE VALUE OF RAM BLOCK
     0066 0226
0171 0068 CA4A          MOV  R10,@ROMBAS(R9) SAVE THE VALUE FOR ALL WPs
     006A 0224
0172 006C 0201          LI   R1,>1234     CHECK FOR INIT ALREADY
     006E 1234
0173      006E' INTVAL EQU  $-2
0174 0070 8A41          C    R1,@INITFL(R9) ARE WE INITIALIZED ?
     0072 0210
0175 0074 1302          JEQ  SCAN08       IF YES, JUMP
0176 0076 04E9          CLR  @INITFL(R9)  CLEAR THE INITIALIZE FLAG
     0078 0210
0177 007A 2FAA  SCAN08 PRINT @MSG1ST(R10) PRINT THE HEADER MSG
     007C 0243'
0178 007E 2FAA          PRINT @ROMMSG(R10) PRINT ROM MESSAGE
     0080 02B1'
0179 0082 2E8A          HEXOUT R10
0180 0084 2FAA          PRINT @RAMMSG(R10) PRINT RAM MESSAGE
     0086 02BC'
0181 0088 2E89          HEXOUT R9
0182                *
0183                *======> COMMAND SCANNER
0184                *
0185 008A 02E0  SCAN04 LWPI WP1           SET UP WP AGAIN
     008C FD00
0186 008E 2FAA          PRINT @PROMPT(R10) PUT OUT '??'
     0090 00ED'
0187 0092 2EC1          INCHAR R1         GET THE INPUT CHAR
0188 0094 0204          LI   R4,MINUS2    SET UP INDEX CTR
     0096 FFFE
0189 0098 0203          LI   R3,CMDLST    GET ADDRESS OF CHAR LIST
     009A 00E4'
0190 009C A0CA          A    R10,R3       ADD IN BASE OF EPROM
0191 009E D033  SCAN05 MOVB *R3+,R0       GET CHAR FROM TABLE
0192 00A0 13F4          JEQ  SCAN04       IF END OF TABLE, PROMPT AGAIN
0193 00A2 05C4          INCT R4           ADVANCE INDEX CTR
0194 00A4 9001          CB   R1,R0        DOES INPUT MATCH TABLE ?
0195 00A6 16FB          JNE  SCAN05       IF NO, CHECK NEXT CHAR
0196 00A8 2FAA          PRINT @CRLF(R10)  DO A CR,LF
     00AA 0270'
0197 00AC 0205          LI   R5,CMDADR    ADD IN TABLE ADDRESS
     00AE 00D4'
0198 00B0 A14A          A    R10,R5       ADD IN THE BASE
0199 00B2 A105          A    R5,R4        ADD BASE OF TABLE TO OFFSET
0200 00B4 C114          MOV  *R4,R4       GET CONTENTS OF TABLE
```

```
00B6 A10A        A    R10,R4       ADD IN BASE OF EPROM
00B8 0203        LI   R3,WP2       GET TARGET WP
00BA 0190
00BC A0C9        A    R9,R3        ADJUST THE RAM START
00BE 0403        BLWP R3           GO TO ROUTINE
00C0 10E4        JMP  SCAN04       GO TO SCANNER
                 *
                 *======> QUIT COMMAND
                 *
00C2 0460  QUIT  B    @TIBUG       GO TO THE MONITOR
00C4 0080
                 *
                 *======> HELP COMMAND
                 *
00C6 C26D  HELP  MOV  @18(R13),R9  GET THE RAM BASE
00C8 0012
00CA C2A9        MOV  @ROMBAS(R9),R10 GET THE BASE ADDRESS OF EPROM
00CC 0224
00CE 2FAA        PRINT @HLPMSG(R10) PRINT THE HELP MESSAGE
00D0 00F3'
00D2 0380        RTWP              RETURN TO COMMAND SCANNER
```

```
0218                *
0219                *======> LIST OF COMMANDS
0220                *
0221 00D4 0880' CMDADR DATA ISSCMD,HELP,INIT,QUIT,XFER00,INIT10
     00D6 00C6'
     00D8 02CE'
     00DA 00C2'
     00DC 0C32'
     00DE 0C8C'
0222 00E0 0CCC'        DATA VEFY00,FORMAT
     00E2 0D7C'
0223 00E4   43  CMDLST BYTE 'C','H','I','Q','X','P','V','F',0
     00E5   48
     00E6   49
     00E7   51
     00E8   58
     00E9   50
     00EA   56
     00EB   46
     00EC   00
0224                *
0225                *=======> MESSAGES
0226                *
0227 00ED   07  PROMPT BYTE BELL,CR,LF,'?','?',0
     00EE   0D
     00EF   0A
     00F0   3F
     00F1   3F
     00F2   00
0228 00F3   07  HLPMSG BYTE BELL,CR,LF
     00F4   0D
     00F5   0A
0229 00F6   43         TEXT 'COMMANDS:'
0230 00FF   0D         BYTE CR,LF,LF
     0100   0A
     0101   0A
0231 0102   48         TEXT 'H = HELP, PRINT HELP LIST'
0232 011B   0D         BYTE CR,LF
     011C   0A
0233 011D   51         TEXT 'Q = QUIT, RETURN TO TIBUG'
0234 0136   0D         BYTE CR,LF
     0137   0A
0235 0138   49         TEXT 'I = INITIALIZE DEMO SOFTWARE, INTERACTIVE'
0236 0161   20         TEXT ' USER INITIALIZATION'
0237 0175   0D         BYTE CR,LF
     0176   0A
0238 0177   43         TEXT 'C = COMMAND ISSUER, USER MAY INTERACTIVELY'
0239 01A1   20         TEXT ' ISSUE COMMANDS TO DISK CONTROLLER'
0240 01C3   0D         BYTE CR,LF
     01C4   0A
0241 01C5   58         TEXT 'X = BLOCK DATA TRANSFER '
0242 01DD   0D         BYTE CR,LF
     01DE   0A
0243 01DF   50         TEXT 'P = INITIALIZE RAM MEMORY WITH PATTERN '
0244 0206   0D         BYTE CR,LF
     0207   0A
0245 0208   56         TEXT 'V = MEMORY TO MEMORY DATA COMPARE '
0246 022A   0D         BYTE CR,LF
     022B   0A
0247 022C   46         TEXT 'F = FORMAT DISKETTE '
```

```
0248 0240   0D          BYTE CR,LF,0
     0241   0A
     0242   00
0249 0243   0D  MSG1ST BYTE CR,LF
     0244   0A
0250 0245   54          TEXT 'TM990/303 DEMO SOFTWARE  REL. 1.1  02/28/80'
0251 0270   0D  CRLF   BYTE CR,LF,0
     0271   0A
     0272   00
0252 0273   4E  NORAM  TEXT 'NO RAM EXISTS, CANNOT USE DEMO'
0253 0291   0D          BYTE CR,LF,0
     0292   0A
     0293   00
0254 0294   42  BADROM TEXT 'BAD EPROM, CANNOT USE DEMO'
0255 02AE   0D          BYTE CR,LF,0
     02AF   0A
     02B0   00
0256 02B1   45  ROMMSG TEXT 'EPROM AT >'
0257 02BB   00          BYTE 0
0258 02BC   20  RAMMSG TEXT '   DEMO RAM AT >'
0259 02CC   00          BYTE 0
```

```
0262              ******************************************************
0263              *              INITIALIZE PARAMETERS                 *
0264              *                                                    *
0265              *  This module initializes the parameters used by the *
0266              *  command issuer subroutine. This section of code is *
0267              *  entered three ways; thru the command scanner, from *
0268              *  the format module and from the command issue module. *
0269              *  The last two are caused when those commands are   *
0270              *  attempted but the parameters have not been previously *
0271              *  initialized.                                      *
0272              *                                                    *
0273              ******************************************************
0274 02CE                EVEN
0275 02CE C26D    INIT   MOV  @18(R13),R9  GET THE RAM BASE ADDR FROM
     02D0 0012
0276              *                        R9 OF OLD WP
0277 02D2 C2A9           MOV  @ROMBAS(R9),R10 GET THE ROM BASE ADDR
     02D4 0224
0278 02D6 04C7    INIT00 CLR  R7           INITIALIZE UNIT VALUE
0279 02D8 2FAA           PRINT @UNTMSG(R10) PRINT THE UNIT QUESTION
     02DA 037A'
0280 02DC 06AA           BL   @INSUB(R10)  MOVE CODE & EXECUTE
     02DE 0D64'
0281 02E0 2E47           HEXIN R7          GET THE UNIT
0282 02E2 02E6'          DATA INIT01       PROCESS THE TERMINATOR
0283 02E4 02D6'          DATA INIT00       IF NOT HEX ASK AGAIN
0284 02E6 0287    INIT01 CI   R7,HEX3      IS THE # TOO LARGE ?
     02E8 0003
0285 02EA 15F5           JGT  INIT00       IF YES, GET ANOTHER ONE
0286 02EC CA47           MOV  R7,@UNIT(R9) STORE UNIT NUMBER
     02EE 022A
0287 02F0 0207    INIT02 LI   R7,DEFCRU    SET UP DEFAULT CRU BASE
     02F2 0200
0288 02F4 2FAA           PRINT @CRUMSG(R10) ASK FOR CRU BASE
     02F6 0404'
0289 02F8 06AA           BL   @INSUB(R10)  MOVE CODE & EXECUTE
     02FA 0D64'
0290 02FC 2E47           HEXIN R7          GET THE CRU BASE
0291 02FE 0302'          DATA INIT03       PROCESS THE TERMINATOR
0292 0300 02F0'          DATA INIT02       IF NOT HEX, ASK AGAIN
0293 0302 0287    INIT03 CI   R7,MAXCRU    IS VALUE TOO LOW ?
     0304 0140
0294 0306 11F4           JLT  INIT02       IF YES, JUMP
0295 0308 0227           AI   R7,>10       ADJUST CRU BASE
     030A 0010
0296 030C CA47           MOV  R7,@BAS303(R9) SAVE THE CRU BASE
     030E 0228
0297 0310 0208           LI   R8,STORFL    GET ADDR OF STORAGE MODE FLAG
     0312 022C
0298 0314 020C           LI   R12,STORMG   GET ADDR OF STORAGE MESSAGE
     0316 0430'
0299 0318 0700           SETO R0           SET UP DEFAULT
0300 031A 06AA           BL   @ASKSUB(R10) GO INPUT STORAGE
     031C 0350'
0301 031E 0208           LI   R8,INTFLG    GET THE ADDR OF INTERRUPT FLAG
     0320 0212
0302 0322 020C           LI   R12,INTMSG   GET THE ADDR OF INTERRUPT MESSAGE
     0324 03A2'
0303 0326 04C0           CLR  R0           SET UP DEFAULT
0304 0328 06AA           BL   @ASKSUB(R10) GO INPUT INTERRUPT FLAG
```

```
     032A 0350'
0305 032C 0208        LI   R8,HALTFL     GET THE ADDR OF HALT FLAG
     032E 0216
0306 0330 020C        LI   R12,HLTMSG    GET THE ADDR OF HALT MESSAGE
     0332 03C3'
0307 0334 0700        SETO R0            SET UP DEFAULT
0308 0336 06AA        BL   @ASKSUB(R10)  GO INPUT HALT FLAG
     0338 0350'
0309 033A 0208        LI   R8,PRNTFL     GET THE ADDR OF PRINT FLAG
     033C 0218
0310 033E 020C        LI   R12,PRTMSG    GET THE ADDR OF PRINT MESSAGE
     0340 03E3'
0311 0342 0700        SETO R0            SET UP DEFAULT
0312 0344 06AA        BL   @ASKSUB(R10)  GO INPUT PRINT FLAG
     0346 0350'
0313 0348 CA6A        MOV  @INTVAL(R10),@INITFL(R9) SET THE INITIALIZE FLAG
     034A 006E'
     034C 0210
0314 034E 0380        RTWP               RETURN TO COMMAND SCANNER
```

```
                 ******************************************************
                 *                  ASKSUB SUBROUTINE                 *
                 ******************************************************
0350 A209  ASKSUB A    R9,R8        ADJUST RAM BASE
0352 C600         MOV  R0,*R8       SET UP DEFAULT
0354 A30A         A    R10,R12      GET REAL ADDR OF MESSAGE
0356 2F9C  ASKSU1 PRINT *R12        PRINT THE MESSAGE
0358 2EC0         INCHAR R0         INPUT THE ANSWER
035A 0280         CI   R0,SPACE     IS IT A SPACE TERMINATOR ?
     035C 2000
035E 130C         JEQ  ASKSU2       IF YES, JUMP
0360 0280         CI   R0,CARRET    IS IT A RETURN TERMINATOR ?
     0362 0D00
0364 1309         JEQ  ASKSU2       IF YES, JUMP
0366 0280         CI   R0,YES       IS IT A 'Y' ?
     0368 5900
036A 1602         JNE  ASKSU3       IF NO, JUMP
036C 0718         SETO *R8          TURN ON FLAG
036E 1004         JMP  ASKSU2       GO TO EXIT
0370 0280  ASKSU3 CI   R0,NO        IS IT A 'N' ?
     0372 4E00
0374 16F0         JNE  ASKSU1       IF NO, ASK QUESTION AGAIN
0376 04D8         CLR  *R8          CLEAR THE FLAG
0378 045B  ASKSU2 RT                RETURN
                 ******************************************************
                 *                INITIALIZE MESSAGES                 *
                 ******************************************************
037A   0D  UNTMSG BYTE CR,LF
     037B   0A
037C   55         TEXT 'UNIT TO BE TESTED (0/1/2/3,DEF=0) ? '
03A0   07         BYTE BELL,0
     03A1   00
03A2   0D  INTMSG BYTE CR,LF
     03A3   0A
03A4   55         TEXT 'USE INTERRUPTS (Y/N,DEF=N) ? '
03C1   07         BYTE BELL,0
     03C2   00
03C3   0D  HLTMSG BYTE CR,LF
     03C4   0A
03C5   48         TEXT 'HALT ON ERROR (Y/N,DEF=Y) ? '
03E1   07         BYTE BELL,0
     03E2   00
03E3   0D  PRTMSG BYTE CR,LF
     03E4   0A
03E5   50         TEXT 'PRINT ON ERROR (Y/N,DEF=Y) ? '
0402   07         BYTE BELL,0
     0403   00
0404   0D  CRUMSG BYTE CR,LF
     0405   0A
0406   53         TEXT 'SOFTWARE CRU BASE OF /303 (DEF= >200) ? '
042E   07         BYTE BELL,0
     042F   00
0430   0D  STORMG BYTE CR,LF
     0431   0A
0432   55         TEXT 'USE MASS STORAGE MODE (Y/N,DEF=Y) ? '
0456   07         BYTE BELL,0
     0457   00
```

```
0359                 ******************************************************
0360                 *                STATUS CHECKER SUBROUTINE           *
0361                 *                                                    *
0362                 *  This subroutine checks the status of the command just *
0363                 *  completed. This subroutine is called by the command  *
0364                 *  issuer subroutine only if the system status check flag *
0365                 *  is on. This routine will set the system error flag if *
0366                 *  there was an error from the execution of the command. *
0367                 *  This flag is used to determine whether to halt on   *
0368                 *  errors. If the print on error flag is on the error  *
0369                 *  message will be printed indicating the relevant     *
0370                 *  information about the command.                      *
0371                 *                                                    *
0372                 ******************************************************
0373 0458                EVEN
0374 0458 C26D  STATUS MOV  @18(R13),R9  GET RAM BASE
     045A 0012
0375 045C C2A9         MOV  @ROMBAS(R9),R10 GET THE ROMBAS OF THE EPROM
     045E 0224
0376 0460 04E9         CLR  @ERRFLG(R9)  CLEAR THE SYSTEM ERROR FLAG
     0462 021A
0377 0464 C11D         MOV  *R13,R4      GET THE ADDRESS OF THE COMMAND LIST
0378 0466 C214         MOV  *R4,R8       GET THE PRIMARY STATUS
0379 0468 0A18         SLA  R8,1         MOVE ERROR BIT INTO MSB
0380 046A 1101         JLT  STAT03       IF ERROR BIT = '1', JUMP
0381 046C 0380  STAT02 RTWP              RETURN
0382 046E 0729  STAT03 SETO @ERRFLG(R9)  TURN ON SYSTEM ERROR FLAG
     0470 021A
0383 0472 C029         MOV  @PRNTFL(R9),R0 IS THE PRINT FLAG ON ?
     0474 0218
0384 0476 13FA         JEQ  STAT02       IF NO, GO TO EXIT
0385 0478 2FAA         PRINT @ERRUNT(R10) PRINT THE UNIT MESAGE
     047A 0506'
0386 047C C224         MOV  @4(R4),R8    GET THE WORD WITH UNIT/CMD
     047E 0004
0387 0480 C1C8         MOV  R8,R7        PUT INTO WORKING REGISTER
0388 0482 0247         ANDI R7,THREE     STRIP OFF THE UNIT
     0484 0003
0389 0486 2E07         CH1OUT R7         PUT OUT THE UNIT
0390 0488 2FAA         PRINT @CMDMSG(R10) PRINT THE COMMAND
     048A 051A'
0391 048C 0BC8         SRC  R8,12        SET UP LEFT CHARACTER
0392 048E 2E08         CH1OUT R8         PUT OUT LEFT CHARACTER
0393 0490 09C8         SRL  R8,12        SET UP RIGHT CHARACTER
0394 0492 2E08         CH1OUT R8         PUT OUT RIGHT CHARACTER
0395 0494 2FAA         PRINT @HOST(R10)  PRINT 'HOST ADDR'
     0496 0568'
0396 0498 2E24         CH1OUT @12(R4)    PRINT HOST STATUS
     049A 000C
0397 049C 2EA4         HEXOUT @14(R4)
     049E 000E
0398 04A0 2FAA         PRINT @DRIVE(R10) PRINT 'STORAGE ADDR'
     04A2 0576'
0399 04A4 2EA4         HEXOUT @6(R4)     PRINT STORAGE ADDR
     04A6 0006
0400 04A8 2EA4         HEXOUT @8(R4)
     04AA 0008
0401 04AC 2FAA         PRINT @LENGTH(R10) PRINT 'LENGTH'
     04AE 0587'
0402 04B0 2EA4         HEXOUT @10(R4)    PRINT THE BYTE COUNT
```

```
     04B2 000A
0403 04B4 2FAA          PRINT @DRIVST(R10) PRINT 'DRIVE STATUS'
     04B6 0592'
0404 04B8 C024          MOV  @2(R4),R0     GET THE H/W STATUS
     04BA 0002
0405 04BC 0B40          SRC  R0,4          RIGHT JUSTIFY LEFT CHARACTER
0406 04BE 2E00          CHIOUT R0          PUT OUT LEFT CHARACTER
0407 04C0 09C0          SRL  R0,12         RIGHT JUSTIFY RIGHT CHARACTER
0408 04C2 2E00          CHIOUT R0          PUT OUT RIGHT CHARACTER
0409 04C4 2FAA          PRINT @BITS(R10)   PRINT 'BITS'
     04C6 0526'
0410 04C8 0705          SETO R5            SET UP FLAG
0411 04CA 0208          LI   R8,STAT1      SET UP BIT SELECTOR
     04CC E05B
0412 04CE C1D4          MOV  *R4,R7        GET THE PRIMARY STATUS
0413 04D0 0206  STAT04  LI   R6,ONE        SET UP VALUE
     04D2 0001
0414 04D4 C208          MOV  R8,R8         ARE WE DONE WITH THIS WORD ?
0415 04D6 130D          JEQ  STAT05        IF YES, JUMP
0416 04D8 0A18          SLA  R8,1          DO WE CHECK THIS BIT ?
0417 04DA 1802          JOC  STAT06        IF YES, JUMP
0418 04DC 0A17          SLA  R7,1          MOVE BIT OVER
0419 04DE 10F8          JMP  STAT04        GO CHECK NEXT BIT
0420 04E0 0A17  STAT06  SLA  R7,1          MOVE DATA BIT INTO CARRY
0421 04E2 1801          JOC  STAT07        IF A '1', JUMP
0422 04E4 04C6          CLR  R6            SET DATA TO ALL ZERO'S
0423 04E6 2FAA  STAT07  PRINT @SPACE1(R10) PUT OUT 1 SPACEARACTER
     04E8 0566'
0424 04EA 2E06          CHIOUT R6          PUT OUT RIGHT CHARACTER
0425 04EC 2FAA          PRINT @SPACE2(R10) PRINT 2 SPACES
     04EE 0565'
0426 04F0 10EF          JMP  STAT04        GO DO NEXT BIT
0427 04F2 0545  STAT05  INV  R5            CHANGE THE FLAG
0428 04F4 1605          JNE  STAT08        IF DONE, JUMP
0429 04F6 C1E4          MOV  @2(R4),R7     GET SECONDARY STATUS
     04F8 0002
0430 04FA 0208          LI   R8,STAT2      SET UP BIT SELECTOR
     04FC AB00
0431 04FE 10E8          JMP  STAT04        GO PROCESS 2ND WORD
0432 0500 2FAA  STAT08  PRINT @CRLF(R10)   DO A CR,LF
     0502 0270'
0433 0504 0380          RTWP               RETURN
```

```
                 ************************************************************
                 *              STATUS CHECKER MESSAGES                     *
                 ************************************************************
0506  0D  ERRUNT BYTE CR,LF,LF
0507  0A
0508  0A
0509  2A         TEXT '*** ERROR UNIT: '
0519  00         BYTE 0
051A  20  CMDMSG TEXT '  COMMAND: '
0525  00         BYTE 0
0526  0D  BITS   BYTE CR,LF,LF
0527  0A
0528  0A
0529  42         TEXT 'BITS:'
052E  0D         BYTE CR,LF
052F  0A
0530  4F         TEXT 'OC  ER  IE  DE  ID  OR  SE  UE  OL  WP  SI'
055A  20         TEXT '  ST  BC'
0562  0D         BYTE CR,LF,0
0563  0A
0564  00
0565  20  SPACE2 BYTE ' '
0566  20  SPACE1 BYTE ' ',0
0567  00
0568  0D  HOST   BYTE CR,LF
0569  0A
056A  48         TEXT 'HOST ADDR: '
0575  00         BYTE 0
0576  20  DRIVE  TEXT '  STORAGE ADDR: '
0586  00         BYTE 0
0587  20  LENGTH TEXT '  LENGTH: '
0591  00         BYTE 0
0592  20  DRIVST TEXT '  DRIVE STATUS: '
05A2  00         BYTE 0
```

```
0462                 ******************************************************
0463                 *                COMMAND ISSUER SUBROUTINE           *
0464                 *                                                    *
0465                 * This subroutine initializes the transfer of the command *
0466                 * list to the disk controller. The command list is first *
0467                 * moved to a buffer used by this routine. This is done so *
0468                 * that commands can be initiated from EPROM. This    *
0469                 * subroutine also determines whether status bits are *
0470                 * checked. The 'HALT ON ERROR' flag is also checked in *
0471                 * this routine. This subroutine is called by the format *
0472                 * module and the issue command module.               *
0473                 *                                                    *
0474                 * This is the DSR that user software may call. Appendix H *
0475                 * of the TM990/303 manual describes this interface. A *
0476                 * example of this interface is the 'FORMAT' command. *
0477                 *                                                    *
0478                 ******************************************************
0479 05A4                EVEN
0480 05A4 C26D  COMISR MOV  @18(R13),R9  R9 = RAM BASE
     05A6 0012
0481 05A8 C2A9         MOV  @ROMBAS(R9),R10 GET THE BASE OF THE EPROM
     05AA 0224
0482 05AC C21D  COMI01 MOV  *R13,R8      GET ADDR OF 1ST LIST
0483 05AE 0207         LI   R7,COMBUF    GET ADDR OF CMD ISSUER BUFFER
     05B0 0000
0484 05B2 A1C9         A    R9,R7        ADJUST FOR RAM
0485 05B4 0206  COMI02 LI   R6,MAXWDS    GET # OF WDS TO BE XFERRED
     05B6 0008
0486 05B8 CDF8  COMI03 MOV  *R8+,*R7+    MOVE DATA
0487 05BA 0606         DEC  R6           DECREMENT THE CTR
0488 05BC 16FD         JNE  COMI03       IF NOT DONE, KEEP MOVING
0489 05BE 04C1         CLR  R1           CLEAR THE UNIT
0490 05C0 C029         MOV  @UNITFL(R9),R0 DO WE INSERT THE UNIT
     05C2 0220
0491 05C4 1302         JEQ  COMI20       IF NO, JUMP
0492 05C6 C069         MOV  @UNIT(R9),R1 SET UP THE UNIT #
     05C8 022A
0493 05CA C029  COMI20 MOV  @INTFLG(R9),R0 DO WE USE INTERRUPTS ?
     05CC 0212
0494 05CE 1302         JEQ  COMI05       IF NO, JUMP
0495 05D0 0221         AI   R1,INTEN     SET UP INTERRUPT ENABLE BIT
     05D2 0080
0496 05D4 E9C1  COMI05 SOC  R1,@-12(R7)  TURN ON UNIT & INT BIT
     05D6 FFF4
0497 05D8 CDF8         MOV  *R8+,*R7+    DO WE CHAIN ?
0498 05DA 1102         JLT  COMI04       IF YES, JUMP
0499 05DC 04D7         CLR  *R7          ZERO THE LAST WORD
0500 05DE 1005         JMP  COMI06       GO SET UP XFER
0501 05E0 C107  COMI04 MOV  R7,R4        SAVE ADDR OF LAST CHAIN WORD
0502 05E2 05C7         INCT R7           R7 POINTS TO NEW LIST
0503 05E4 C507         MOV  R7,*R4       UPDATE CHAIN POINTER
0504 05E6 C218         MOV  *R8,R8       GET NEXT CHAIN ADDR
0505 05E8 10E5         JMP  COMI02       GO MOVE NEXT LIST
0506 05EA 0208  COMI06 LI   R8,COMBUF    GET ADDR OF 1ST LIST
     05EC 0000
0507 05EE A209         A    R9,R8        ADJUST FOR RAM
0508 05F0 C329         MOV  @BAS303(R9),R12 SET UP CRU ADDR OF TM900/303
     05F2 0228
0509 05F4 1F0C         TB   BUSY         IS THE CONTROLLER BUSY ?
0510 05F6 1606         JNE  COMI08       IF NO, JUMP
```

```
05F8 C029          MOV  @PRNTFL(R9),R0 DO WE PRINT ?
05FA 0218
05FC 1302          JEQ  COMI07        IF NO, JUMP
05FE 2FAA          PRINT @BSYMSG(R10) PRINT 'CONTROLLER BUSY'
0600 07A8'
0602 0380  COMI07 RTWP                EXIT SUBROUTINE
           *
           *======> SET UP INTERRUPT SERVICE ROUTINES
           *
0604 C060  COMI08 MOV  @CLKINT,R1     GET ADDR OF CLOCK INT VECTOR
0606 000E
0608 0202          LI   R2,BRANCH     GET BRANCH OPCODE
060A 0460
060C 0203          LI   R3,XFERIN     GET PC
060E 0722'
0610 CC42          MOV  R2,*R1+       SET UP INTERMEDIATE VECTOR
0612 A0CA          A    R10,R3        ADJUST FOR EPROM ADDR
0614 C443          MOV  R3,*R1
0616 C060          MOV  @FLPVEC,R1    GET ADDR OF VECTOR 2
0618 000A
061A CC42          MOV  R2,*R1+       SET UP BRANCH COMMAND
061C 0203          LI   R3,CONINT     GET ADDR OF CONTROLLER ISR
061E 0774'
0620 A0CA          A    R10,R3        ADD IN EPROM BASE
0622 C443          MOV  R3,*R1        PUT REAL ADDR IN VECTOR
0624 0200          LI   R0,>F00       SET UP MAP BITS
0626 0F00
0628 0207          LI   R7,COMBUF     SET UP LSB ADDR
062A 0000
062C A1C9          A    R9,R7         ADJUST FOR RAM
062E 020C          LI   R12,CRU01     GET CRU ADDR OF CLOCK
0630 0100
0632 33EA          LDCR @CLKVAL(R10),15 LOAD THE CLOCK VALUE
0634 074A'
0636 1E00          SBZ  INTCLK        GO TO INTERRUPT MODE
0638 1D03          SBO  CLKBIT        TURN ON CLOCK INT
063A 1D02          SBO  INT303        TURN FLOPPY INT
063C 0300          LIMI 3             OPEN UP INTERRUPT WINDOW
063E 0003
0640 C329          MOV  @BAS303(R9),R12 GET CONTROLLER CRU ADDR
0642 0228
0644 C0E7          MOV  @4(R7),R3     GET THE WD WITH INT BIT
0646 0004
0648 0A93          SLA  R3,9          PUT INT BIT INTO CARRY
064A 1701          JNC  COMI21        IF NO INT, JUMP
064C 1D0D          SBO  INTBIT        ELSE ENABLE BIT AT CRU
           *
           *======> PUT OUT COMMAND
           *
064E 3200  COMI21 LDCR R0,8           PUT OUT MAP BITS
0650 1D08          SBO  COMAND        SET COMMAND BIT TO 1
0652 1D0A          SBO  CUE           SET CUE TO 1
0654 1F0B  COMI09 TB   ACCEPT         HAS THE DATA BEEN TAKEN ?
0656 16FE          JNE  COMI09        IF NO, KEEP TESTING
0658 1E0A          SBZ  CUE           TAKE CUE LOW
065A 1F0B  COMI10 TB   ACCEPT         HAS ACCEPT GONE LOW ?
065C 13FE          JEQ  COMI10        IF NO, KEEP CHECKING
           *
           *======> WRITE OUT 2ND BYTE OF THE ADDRESS
           *
```

```
0557 065E 3207          LDCR R7,8          SEND OUT 2ND BYTE
0558 0660 1E08          SBZ  COMAND        TAKE COMMAND LOW
0559 0662 1D0A          SBO  CUE           SET CUE TO 1
0560 0664 1F0B  COMI11  TB   ACCEPT        HAS DATA BEEN TAKEN ?
0561 0666 16FE          JNE  COMI11        IF NO, KEEP TESTING
0562 0668 1E0A          SBZ  CUE           TAKE CUE LOW
0563 066A 1F0B  COMI12  TB   ACCEPT        HAS ACCEPT GONE LOW ?
0564 066C 13FE          JEQ  COMI12        IF NO, KEEP TESTING
0565 066E 06C7          SWPB R7            PUT LAST BYTE IN LEFT BYTE
0566            *
0567            *======> WRITE OUT 3RD BYTE
0568            *
0569 0670 3207          LDCR R7,8          LOAD 3RD BYTE
0570 0672 1D0A          SBO  CUE           SET CUE TO 1
0571 0674 1F0B  COMI13  TB   ACCEPT        HAS THE DATA BEEN TAKEN
0572 0676 16FE          JNE  COMI13        IF NO, KEEP CHECKING
0573 0678 1E0A          SBZ  CUE           TAKE CUE LOW
0574 067A 06C7          SWPB R7            STRAIGHTEN OUT ADDR
0575 067C 020C          LI   R12,CRU01     SET UP CRU BASE OF CLOCK
     067E 0100
0576            *
0577            *======> START CLOCK
0578            *
0579 0680 C0E7  COMI19  MOV  @4(R7),R3     GET THE WD WITH INT BIT
     0682 0004
0580 0684 04E9          CLR  @INTEXP(R9)   CLEAR INT EXPECTED FLAG
     0686 021C
0581 0688 04E9          CLR  @INTOCC(R9)   CLEAR INT OCCURRED FLAG
     068A 0214
0582 068C 0A93          SLA  R3,9          PUT INT BIT INTO CARRY
0583 068E 1702          JNC  COMI14        IF NO INT, JUMP
0584 0690 0729          SETO @INTEXP(R9)   SET THE INT EXPECTED FLAG
     0692 021C
0585 0694 0200  COMI14  LI   R0,CMTOUT     SET UP ADDR OF TIMEOUT ROUTINE
     0696 074C'
0586 0698 A00A          A    R10,R0        ADD IN EPROM BASE
0587 069A C060          MOV  @CLKINT,R1    RE-SET UP CLOCK INT VECTOR
     069C 000E
0588 069E 05C1          INCT R1            POINT TO ADDR
0589 06A0 C440          MOV  R0,*R1        PUT IN NEW ADDR
0590 06A2 0200          LI   R0,TICKCT     SET UP COUNTER FOR 25 SEC
     06A4 0064
0591 06A6 33EA          LDCR @CLKVAL(R10),15 RELOAD THE CLOCK
     06A8 074A'
0592 06AA C157  COMI15  MOV  *R7,R5        IS THE COMMAND DONE ?
0593 06AC 13FE          JEQ  COMI15        IF NO, KEEP CHECKING
0594 06AE 0205          LI   R5,WAIT       SET UP WAIT VALUE
     06B0 0010
0595 06B2 0605  COMI22  DEC  R5            DEC WAIT VALUE
0596 06B4 16FE          JNE  COMI22        IF NOT DONE, CAN BE PULLED
0597 06B6 0300          LIMI 0             SHUT DOWN INTERRUPTS
     06B8 0000
0598 06BA C169          MOV  @INTEXP(R9),R5 WERE WE EXPECTING INTERRUPTS ?
     06BC 021C
0599 06BE 1308          JEQ  COMI16        IF NO, JUMP
0600 06C0 C169          MOV  @INTOCC(R9),R5 DID WE GET AN INTERRUPT ?
     06C2 0214
0601 06C4 1605          JNE  COMI16        IF YES, JUMP
0602 06C6 C169          MOV  @PRNTFL(R9),R5 DO WE PRINT ?
     06C8 0218
```

```
0603 06CA 1302          JEQ  COMI16       IF NO, JUMP
0604 06CC 2FAA          PRINT @NOINT(R10) ELSE PRINT NO INT GENERATED
     06CE 0829'
0605 06D0 C169  COMI16 MOV  @STATFL(R9),R5 DO WE CHECK STATUS ?
     06D2 0222
0606 06D4 130E          JEQ  COMI17       IF NO, JUMP
0607 06D6 C007          MOV  R7,R0        SET UP ADDR OF CMD LIST FOR STATUS
0608 06D8 0205          LI   R5,WP4       SET UP WP ADDR
     06DA 01D0
0609 06DC A149          A    R9,R5        ADJUST RAM
0610 06DE 0206          LI   R6,STATUS    SET UP PC ADDR
     06E0 0458'
0611 06E2 A18A          A    R10,R6       ADJUST EPROM BASE
0612 06E4 0405          BLWP R5           GO CHECK STATUS
0613 06E6 C169          MOV  @ERRFLG(R9),R5 DID WE HAVE AN ERROR ?
     06E8 021A
0614 06EA 1303          JEQ  COMI17       IF NO, JUMP
0615 06EC C169          MOV  @HALTFL(R9),R5 DO WE HALT ON ERRORS ?
     06EE 0216
0616 06F0 1628          JNE  XFERI1       IF YES, JUMP TO EXIT
0617 06F2 C167  COMI17 MOV  @16(R7),R5    DO WE CHAIN ?
     06F4 0010
0618 06F6 1112          JLT  COMI18       IF YES, JUMP
0619 06F8 C169          MOV  @LOOPFL(R9),R5 DO WE LOOP ?
     06FA 021E
0620 06FC 1325          JEQ  XFERI2       IF NO, EXIT
0621 06FE C00C          MOV  R12,R0       SAVE CRU ADDR
0622 0700 020C          LI   R12,>80      GET CRU ADDR OF TMS9902
     0702 0080
0623 0704 1F15          TB   21           HAS A KEY BEEN HIT ?
0624 0706 1607          JNE  COMI23       IF NO, JUMP
0625 0708 04C5          CLR  R5           CLEAR INPUT REG
0626 070A 3605          STCR R5,8         READ THE TMS 9902
0627 070C 0285          CI   R5,ESCAPE    WAS THIS AN ESCAPE CHAR ?
     070E 1B00
0628 0710 1602          JNE  COMI23       IF NO, CONTINUE
0629 0712 046A          B    @SCAN04(R10) GO TO COMMAND SCANNER
     0714 008A'
0630 0716 C300  COMI23 MOV  R0,R12        RESTORE R12
0631 0718 046A          B    @COMI01(R10) YES, BRANCH
     071A 05AC'
0632 071C C1E7  COMI18 MOV  @18(R7),R7    GET THE NEW ADDR
     071E 0012
0633 0720 10AF          JMP  COMI19       GO PROCESS NEXT COMMAND
```

```
                 ******************************************************
                 *              INTERRUPT SERVICE ROUTINES            *
                 ******************************************************
                 *
                 *======> ISR FOR CRU TRANSFER TIMEOUT
                 *
0722 0300        XFERIN LIMI 0            SHUT DOWN INTERRUPTS
     0724 0000
0726 C26D              MOV  @18(R13),R9   GET RAM BASE
     0728 0012
072A 020C              LI   R12,CRU01     GET CLOCK INT
     072C 0100
072E 1E00              SBZ  INTCLK        GO TO INT MODE
0730 1E03              SBZ  CLKBIT        CLEAR THE INT
0732 1D03              SBO  CLKBIT        SET UP FOR NEXT INT
0734 C069              MOV  @PRNTFL(R9),R1 DO WE PRINT ?
     0736 0218
0738 1304              JEQ  XFERI1        IF NO, JUMP
073A C069              MOV  @ROMBAS(R9),R1 GET THE EPROM BASE
     073C 0224
073E 2FA1              PRINT @XFRMSG(R1) PRINT THE MESSAGE
     0740 07D1'
0742 020E        XFERI1 LI   R14,XFERI2    CHANGE RETURN ADDRESS
     0744 0748'
0746 A381              A    R1,R14        ADJUST FOR RAM
0748 0380        XFERI2 RTWP               RETURN
074A 5B8D        CLKVAL DATA >5B8D         250 MS TICK VALUE
                 *
                 *======> COMMAND EXECUTION TIME OUT ISR
                 *
074C 0300        CMTOUT LIMI 0            SHUT DOWN INTERRUPTS
     074E 0000
0750 C26D              MOV  @18(R13),R9   GET RAM BASE
     0752 0012
0754 020C              LI   R12,CRU01     GET 9901 CRU ADDR
     0756 0100
0758 1E00              SBZ  INTCLK        GO TO INT MODE
075A 1E03              SBZ  CLKBIT        CLEAR THE INTERRUPT
075C 1D03              SBO  CLKBIT        SET UP THE INTERRUPT
075E 0629              DEC  @WP3(R9)      DEC THE TIME OUT CTR
     0760 01B0
0762 16F2              JNE  XFERI2        IF NOT 0, EXIT
0764 C069              MOV  @PRNTFL(R9),R1 DO WE PRINT ?
     0766 0218
0768 13EC              JEQ  XFERI1        IF NO, JUMP
076A C069              MOV  @ROMBAS(R9),R1 GET THE EPROM BASE
     076C 0224
076E 2FA1              PRINT @TOUTMG(R1) PRINT THE TIME OUT MESSAGE
     0770 07FE'
0772 10E7              JMP  XFERI1        GO TO ERROR EXIT
                 *
                 *======> CONTROLLER ISR
                 *
0774 0300        CONINT LIMI 0            SHUT DOWN INTERRUPTS
     0776 0000
0778 C26D              MOV  @18(R13),R9   GET RAM BASE
     077A 0012
077C C329              MOV  @BAS303(R9),R12 GET /303 CRU ADDR
     077E 0228
0780 1E0D              SBZ  INTBIT        CLEAR INTERRUPT
```

```
0679 0782 1D0D        SBO  INTBIT       SET UP INTERRUPT AGAIN
0680 0784 020C        LI   R12,CRU01    GET CRU BASE OF 9901
     0786 0100
0681 0788 1E00        SBZ  INTCLK       GO TO INTERRUPT MODE
0682 078A 1E02        SBZ  INT303       CLEAR INTERRUPT
0683 078C 1D02        SBO  INT303       SET UP INTERRUPT
0684 078E 0729        SETO @INTOCC(R9)  SET INTERRUPT OCCURRED FLAG
     0790 0214
0685 0792 C029        MOV  @INTEXP(R9),R0 DID WE EXPECT AN INTERRUPT ?
     0794 021C
0686 0796 16D8        JNE  XFERI2       IF YES, GO TO EXIT
0687 0798 C069        MOV  @PRNTFL(R9),R1 DO WE PRINT ?
     079A 0218
0688 079C 13D2        JEQ  XFERI1       IF NO, JUMP
0689 079E C069        MOV  @ROMBAS(R9),R1 GET EPROM BASE
     07A0 0224
0690 07A2 2FA1        PRINT @NOTEXP(R1) PRINT 'UNEXP INT' MSG
     07A4 0857'
0691 07A6 10CD        JMP  XFERI1       GO TO ERROR EXIT
0692             ***********************************************************
0693             *                COMMAND ISSUER MESSAGES                  *
0694             ***********************************************************
0695 07A8   0D   BSYMSG BYTE CR,LF
     07A9   0A
0696 07AA   43        TEXT 'CONTROLLER BUSY - COMMAND NOT ISSUED'
0697 07CE   0D        BYTE CR,LF,0
     07CF   0A
     07D0   00
0698 07D1   0D   XFRMSG BYTE CR,LF
     07D2   0A
0699 07D3   43        TEXT 'CONTROLLER WOULD NOT ACCEPT LIST ADDRESS'
0700 07FB   0D        BYTE CR,LF,0
     07FC   0A
     07FD   00
0701 07FE   0D   TOUTMG BYTE CR,LF
     07FF   0A
0702 0800   43        TEXT 'COMMAND DID NOT COMPLETE IN 25 SECONDS'
0703 0826   0D        BYTE CR,LF,0
     0827   0A
     0828   00
0704 0829   0D   NOINT  BYTE CR,LF
     082A   0A
0705 082B   43        TEXT 'COMMAND DID NOT INTERRUPT UPON COMPLETION'
0706 0854   0D        BYTE CR,LF,0
     0855   0A
     0856   00
0707 0857   0D   NOTEXP BYTE CR,LF
     0858   0A
0708 0859   55        TEXT 'UNEXPECTED INTERRUPT FROM CONTROLLER'
0709 087D   0D        BYTE CR,LF,0
     087E   0A
     087F   00
```

```
                 ***********************************************************
                 *                 ISSUE COMMAND SUBROUTINE                *
                 *                                                         *
                 *  The 'ISSUE COMMAND' subroutine allows the user to      *
                 *  interactively issue a maximum of 10 command lists to   *
                 *  the floppy disk controller. The operator enters this   *
                 *  subroutine by entering a 'C' to a '??' prompt. The     *
                 *  operator is then asked a series of questions which     *
                 *  allow him to build command lists which can later be    *
                 *  issued to the controller.                              *
                 *                                                         *
                 ***********************************************************
0880                    EVEN
0880 C26D        ISSCMD MOV  @18(R13),R9  GET RAM BASE FROM OLD WP R9
0882 0012
0884 C2A9               MOV  @ROMBAS(R9),R10  GET EPROM BASE
0886 0224
0888 8A6A               C    @INTVAL(R10),@INITFL(R9) DO WE NEED TO INIT ?
088A 006E'
088C 0210
088E 1307               JEQ  ISSC00       IF NO, JUMP
0890 0201               LI   R1,WP3       SET UP FOR BLWP
0892 01B0
0894 A049               A    R9,R1        ADJUST RAM BASE
0896 0202               LI   R2,INIT      SET UP PC ADDR
0898 02CE'
089A A08A               A    R10,R2       ADJUST ROM BASE
089C 0401               BLWP R1           GO DO INITIALIZATION
089E 04E9        ISSC00 CLR  @UNITFL(R9)  CLEAR UNIT FLAG, WE WILL INSERT
08A0 0220
                 *                        OUR OWN
08A2 0207               LI   R7,BUF       GET ADDR OF BUFFER
08A4 00C8
08A6 A1C9               A    R9,R7        ADJUST RAM BASE
08A8 0208               LI   R8,CHNCTR    SET UP CTR
08AA 000A
08AC 04F7        ISSC01 CLR  *R7+         ZERO WORD 0 (PRIMARY STATUS)
08AE 04F7               CLR  *R7+         ZERO WORD 1 (SECONDARY STATUS)
08B0 04C1        ISSC02 CLR  R1           CLEAR INPUT REGISTER
08B2 2FAA               PRINT @COMMSG(R10) ASK FOR COMMAND
08B4 0A66'
08B6 06AA               BL   @INSUB(R10)  MOVE CODE & EXECUTE
08B8 0D64'
08BA 2E41               HEXIN R1          GET A HEX VALUE
08BC 09D8'              DATA ISSC21       IF 'CR' GO CALL COMMAND ISSUER
08BE 08B0'              DATA ISSC02       IF NOT HEX, ASK AGAIN
08C0 0281        ISSC03 CI   R1,MAXCMD    IS THIS TOO LARGE ?
08C2 0010
08C4 15F5               JGT  ISSC02       IF YES, ASK AGAIN
08C6 06C1               SWPB R1           LEFT JUSTIFY FOR MOVE
08C8 04C4        ISSC04 CLR  R4           SET DEFAULT TO 0
08CA 2FAA               PRINT @UNITMG(R10) ASK FOR UNIT
08CC 0A7F'
08CE 06AA               BL   @INSUB(R10)  MOVE CODE & EXECUTE
08D0 0D64'
08D2 2E44               HEXIN R4          GET A HEX VALUE
08D4 08D8'              DATA ISSC05       GO PROCESS THE DEFAULT
08D6 08C8'              DATA ISSC04       IF NOT HEX, ASK AGAIN
08D8 0284        ISSC05 CI   R4,MAXUNT    IS THE UNIT TOO LARGE ?
08DA 0003
```

```
0757 08DC 15F5          JGT  ISSC04       IF YES, ASK AGAIN
0758 08DE A044          A    R4,R1        ADD UNIT TO COMMAND
0759 08E0 2FAA  ISSC06 PRINT @VERMSG(R10) ASK VERIFY MESSAGE
     08E2 0A9B'
0760 08E4 2EC5          INCHAR R5         GET THE CHARACTER
0761 08E6 04C4          CLR  R4           SET BIT TO 0
0762 08E8 0285          CI   R5,SPACE     SPACE TERMINATOR ?
     08EA 2000
0763 08EC 130B          JEQ  ISSC07       IF YES, JUMP
0764 08EE 0285          CI   R5,CARRET    CARRIAGE RETURN TERMINATOR ?
     08F0 0D00
0765 08F2 1308          JEQ  ISSC07       IF YES, JUMP
0766 08F4 0285          CI   R5,NO        IS ANSWER A 'N' ?
     08F6 4E00
0767 08F8 1305          JEQ  ISSC07       IF YES, JUMP
0768 08FA 0285          CI   R5,YES       IS ANSWER A 'Y' ?
     08FC 5900
0769 08FE 16F0          JNE  ISSC06       IF NOT 'Y' or 'N', ASK AGAIN
0770 0900 0204          LI   R4,VFYBIT    TURN ON VERIFY BIT
     0902 0040
0771 0904 A044  ISSC07 A    R4,R1         COMPLETE WORD 2
0772 0906 CDC1          MOV  R1,*R7+      PUT WORD 2 IN LIST
0773 0908 C069  ISSC08 MOV  @STORFL(R9),R1 MASS STORAGE MODE ?
     090A 022C
0774 090C 1316          JEQ  ISSC25       IF NO, JUMP
0775 090E 04C1          CLR  R1           CLEAR INPUT REGISTER
0776 0910 2FAA          PRINT @STORE1(R10) ASK FOR MSW OF STORAGE ADDR
     0912 0B1B'
0777 0914 06AA          BL   @INSUB(R10)  MOVE CODE & EXECUTE
     0916 0D64'
0778 0918 2E41          HEXIN R1          GET THE MSW
0779 091A 091E'         DATA ISSC09       PROCESS THE 0
0780 091C 0908'         DATA ISSC08       IF NOT HEX, AS AGAIN
0781 091E 0A11  ISSC09 SLA  R1,1          STRIP OFF BIT 0
0782 0920 0911          SRL  R1,1
0783 0922 CDC1          MOV  R1,*R7+      PUT WORD 3 AWAY
0784 0924 04C1  ISSC10 CLR  R1            CLEAR INPUT REGISTER
0785 0926 2FAA          PRINT @STORE2(R10) ASK FOR LSW OF STORAGE ADR
     0928 0B44'
0786 092A 06AA          BL   @INSUB(R10)  MOVE CODE & EXECUTE
     092C 0D64'
0787 092E 2E41          HEXIN R1          GET THE LSW
0788 0930 0934'         DATA ISSC11       PROCESS THE 0
0789 0932 0924'         DATA ISSC10       IF NOT HEX, ASK AGAIN
0790 0934 0911  ISSC11 SRL  R1,1          STRIP OFF BIT 15
0791 0936 0A11          SLA  R1,1
0792 0938 1021          JMP  ISSC28       GO GET THE LENGTH
0793 093A 04C1  ISSC25 CLR  R1            CLEAR INPUT REGISTER
0794 093C 2FAA          PRINT @PHY1(R10)  ASK FOR TRACK #
     093E 0B6D'
0795 0940 06AA          BL   @INSUB(R10)  MOVE CODE & EXECUTE
     0942 0D64'
0796 0944 2E41          HEXIN R1          GET THE TRACK #
0797 0946 094A'         DATA ISSC26       PROCESS 0
0798 0948 093A'         DATA ISSC25       IF NOT HEX, ASK AGAIN
0799 094A 0261  ISSC26 ORI  R1,>8000      TURN ON PHYSICAL BIT
     094C 8000
0800 094E CDC1          MOV  R1,*R7+      PUT WORD 3 IN LIST
0801 0950 04C0  ISSC27 CLR  R0            INITIALIZE SURFACE TO 0
0802 0952 2FAA          PRINT @PHY2(R10)  ASK FOR SURFACE
```

```
     0954 0B95'
0803 0956 06AA          BL   @INSUB(R10)  MOVE CODE & EXECUTE
     0958 0D64'
0804 095A 2E40          HEXIN R0          GET SURFACE/SECTOR
0805 095C 0960'         DATA ISSC29       PROCESS 0
0806 095E 0950'         DATA ISSC27       IF NOT HEX, ASK AGAIN
0807 0960 0240  ISSC29 ANDI R0,1          STRIP OFF BIT
     0962 0001
0808 0964 0A80          SLA  R0,8         LEFT JUSTIFY
0809 0966 04C1  ISSC31 CLR  R1            CLEAR INPUT REGISTER
0810 0968 2FAA          PRINT @PHY3(R10)  ASK FOR SECTOR
     096A 0BB1'
0811 096C 06AA          BL   @INSUB(R10)  MOVE CODE & EXECUTE
     096E 0D64'
0812 0970 2E41          HEXIN R1          GET INPUT
0813 0972 0976'         DATA ISSC32       PROCESS 0
0814 0974 0966'         DATA ISSC31       IF NOT HEX, ASK AGAIN
0815 0976 0241  ISSC32 ANDI R1,>1F        STRIP OFF BITS
     0978 001F
0816 097A D040          MOVB R0,R1        PUT IN SURFACE
0817 097C CDC1  ISSC28 MOV  R1,*R7+       PUT WORD 4 IN LIST
0818 097E 04C1  ISSC12 CLR  R1            ZERO THE LENGTH
0819 0980 2FAA          PRINT @BYTCNT(R10) ASK FOR LENGTH
     0982 0AC6'
0820 0984 06AA          BL   @INSUB(R10)  MOVE CODE & EXECUTE
     0986 0D64'
0821 0988 2E41          HEXIN R1          GET THE LENGTH
0822 098A 098E'         DATA ISSC13       PROCESS THE 0
0823 098C 097E'         DATA ISSC12       IF NOT HEX, ASK AGAIN
0824 098E 0241  ISSC13 ANDI R1,>FFFE      STRIP OFF BIT 15
     0990 FFFE
0825 0992 CDC1          MOV  R1,*R7+      PUT LENGTH IN LIST
0826 0994 0201  ISSC14 LI   R1,>F         INITIALIZE MAP BITS
     0996 000F
0827 0998 2FAA          PRINT @MAPMSG(R10) ASK FOR MAP ADDR
     099A 0AD6'
0828 099C 06AA          BL   @INSUB(R10)  MOVE CODE & EXECUTE
     099E 0D64'
0829 09A0 2E41          HEXIN R1          GET THE MAP
0830 09A2 09A6'         DATA ISSC15       PROCESS THE 0
0831 09A4 0994'         DATA ISSC14       IF NOT HEX, ASK AGAIN
0832 09A6 0281  ISSC15 CI   R1,MAXMAP     IS THE MAP TOO BIG ?
     09A8 000F
0833 09AA 15F4          JGT  ISSC14       IF YES, ASK AGAIN.
0834 09AC CDC1          MOV  R1,*R7+      PUT THE MAP IN THE LIST
0835 09AE 04C1  ISSC16 CLR  R1            CLEAR THE ADDRESS
0836 09B0 2FAA          PRINT @MEMMSG(R10) ASK FOR MEMORY ADDRESS
     09B2 0AFC'
0837 09B4 06AA          BL   @INSUB(R10)  MOVE CODE & EXECUTE
     09B6 0D64'
0838 09B8 2E41          HEXIN R1          GET THE ADDRESS
0839 09BA 09BE'         DATA ISSC17       PROCESS THE 0
0840 09BC 09AE'         DATA ISSC16       IF NOT HEX, ASK AGAIN
0841 09BE 0241  ISSC17 ANDI R1,>FFFE      STRIP OFF BIT 15
     09C0 FFFE
0842 09C2 CDC1          MOV  R1,*R7+      PUT MEMORY ADDR IN LIST
0843 09C4 0608          DEC  R8           DEC LIST CTR
0844 09C6 130D          JEQ  ISSC20       IF LAST LIST, JUMP
0845 09C8 0201          LI   R1,CHNBIT    TURN ON CHAIN BIT AND MAP >F
     09CA 800F
```

```
0846 09CC CDC1          MOV  R1,*R7+       PUT INTO COMMAND LIST
0847 09CE 05C7          INCT R7            R7 POINTS TO NEXT LIST
0848 09D0 C9C7          MOV  R7,-2(R7)     SET UP CHAIN ADDR
     09D2 FFFE
0849 09D4 046A          B    @ISSC01(R10)  GO ASK QUESTIONS AGAIN
     09D6 08AC'
0850 09D8 0288  ISSC21  CI   R8,CHNCTR     DID THEY STOP AT 1ST LIST ?
     09DA 000A
0851 09DC 1343          JEQ  ISSC30        IF YES, EXIT
0852 09DE 0227          AI   R7,MINUS8     POINT TO LAST CHAIN ADDR
     09E0 FFF8
0853 09E2 04F7  ISSC20  CLR  *R7+          CLEAR OUT CHAIN POINTER
0854 09E4 04D7          CLR  *R7
0855 09E6 0200          LI   R0,BUF        SET UP ADDR OF COMMAND LISTS
     09E8 00C8
0856 09EA A009          A    R9,R0         ADJUST RAM BASE
0857 09EC 2FAA  ISSC18  PRINT @STATMG(R10) ASK CHECK STATUS COMMAND
     09EE 0BD2'
0858 09F0 2EC1          INCHAR R1          GET ANSWER
0859 09F2 0705          SETO R5            SET UP DEFAULT
0860 09F4 0281          CI   R1,SPACE      SPACE TERMINATOR ?
     09F6 2000
0861 09F8 130A          JEQ  ISSC33        IF YES, JUMP
0862 09FA 0281          CI   R1,CARRET     CARRIAGE RETURN TERMINATOR ?
     09FC 0D00
0863 09FE 1307          JEQ  ISSC33        IF YES, JUMP
0864 0A00 0281          CI   R1,YES        IS ANSWER = 'Y' ?
     0A02 5900
0865 0A04 1304          JEQ  ISSC33        IF YES, JUMP
0866 0A06 0281          CI   R1,NO         IS ANSWER = 'N' ?
     0A08 4E00
0867 0A0A 16F0          JNE  ISSC18        IF NOT 'Y' or 'N', JUMP
0868 0A0C 04C5          CLR  R5            SET FOR NO STATUS
0869 0A0E CA45  ISSC33  MOV  R5,@STATFL(R9) STORE FLAG VALUE
     0A10 0222
0870 0A12 2FAA  ISSC23  PRINT @LOOPMG(R10) ASK LOOP QUESTION
     0A14 0BF0'
0871 0A16 04C1          CLR  R1            CLEAR THE LOOP FLAG
0872 0A18 2EC5          INCHAR R5          GET THE ANSWER
0873 0A1A 0285          CI   R5,SPACE      SPACE TERMINATOR ?
     0A1C 2000
0874 0A1E 130A          JEQ  ISSC34        IF YES, JUMP
0875 0A20 0285          CI   R5,CARRET     CARRIAGE RETURN TERMINATOR ?
     0A22 0D00
0876 0A24 1307          JEQ  ISSC34        IF YES, JUMP
0877 0A26 0285          CI   R5,NO         IS THE ANSWER NO ?
     0A28 4E00
0878 0A2A 1304          JEQ  ISSC34        IF YES, JUMP
0879 0A2C 0285          CI   R5,YES        IS THE ANSWER YES ?
     0A2E 5900
0880 0A30 16F0          JNE  ISSC23        IF NO, ASK AGAIN
0881 0A32 0701          SETO R1            SET THE LOOP FLAG
0882 0A34 CA41  ISSC34  MOV  R1,@LOOPFL(R9) STORE FLAG VALUE
     0A36 021E
0883 0A38 2FAA  ISSC24  PRINT @EXMSG(R10)  ASK EXECUTE QUESTION
     0A3A 0C19'
0884 0A3C 2EC5          INCHAR R5          GET THE ANSWER
0885 0A3E 0285          CI   R5,NO         IS THE ANSWER NO ?
     0A40 4E00
0886 0A42 1310          JEQ  ISSC30        IF YES, EXIT ROUTINE
```

```
0887 0A44 0285          CI   R5,SPACE     SPACE TERMINATOR ?
     0A46 2000
0888 0A48 1306          JEQ  ISSC35       IF YES, JUMP
0889 0A4A 0285          CI   R5,CARRET    CARRIAGE RETURN TERMINATOR ?
     0A4C 0D00
0890 0A4E 1303          JEQ  ISSC35       IF YES, JUMP
0891 0A50 0285          CI   R5,YES       IS THE ANSWER YES ?
     0A52 5900
0892 0A54 16F1          JNE  ISSC24       IF NO, ASK AGAIN
0893 0A56 0204  ISSC35  LI   R4,WP3       SET UP WP ADDR
     0A58 01B0
0894 0A5A A109          A    R9,R4        ADJUST FOR RAM
0895 0A5C 0205          LI   R5,COMISR    SET UP PC ADDR
     0A5E 05A4
0896 0A60 A14A          A    R10,R5       ADJUST FOR ROM
0897 0A62 0404          BLWP R4           ISSUE COMMAND
0898 0A64 0380  ISSC30  RTWP              RETURN
```

```
0900              ***********************************************************
0901              *                ISSUE COMMAND MESSAGES                   *
0902              ***********************************************************
0903 0A66   0D  COMMSG BYTE CR,LF
     0A67   0A
0904 0A68   43         TEXT 'COMMAND # (0 - >10) ? '
0905 0A7E   00         BYTE 0
0906 0A7F   0D  UNITMG BYTE CR,LF
     0A80   0A
0907 0A81   55         TEXT 'UNIT # (0/1/2/3,DEF=0) ? '
0908 0A9A   00         BYTE 0
0909 0A9B   0D  VERMSG BYTE CR,LF
     0A9C   0A
0910 0A9D   44         TEXT 'DATA VERIFY ON READ/WRITE (Y/N,DEF=N) ? '
0911 0AC5   00         BYTE 0
0912 0AC6   0D  BYTCNT BYTE CR,LF
     0AC7   0A
0913 0AC8   42         TEXT 'BYTE COUNT ? '
0914 0AD5   00         BYTE 0
0915 0AD6   0D  MAPMSG BYTE CR,LF
     0AD7   0A
0916 0AD8   4D         TEXT 'MEMORY MAP ADDRESS (0- >F,DEF=F) ? '
0917 0AFB   00         BYTE 0
0918 0AFC   0D  MEMMSG BYTE CR,LF
     0AFD   0A
0919 0AFE   4D         TEXT 'MEMORY ADDRESS (0- >FFFE) ? '
0920 0B1A   00         BYTE 0
0921 0B1B   0D  STORE1 BYTE CR,LF
     0B1C   0A
0922 0B1D   4D         TEXT 'MSW DISK STORAGE ADDRESS (0- >7FFE) ? '
0923 0B43   00         BYTE 0
0924 0B44   0D  STORE2 BYTE CR,LF
     0B45   0A
0925 0B46   4C         TEXT 'LSW DISK STORAGE ADDRESS (0- >FFFE) ? '
0926 0B6C   00         BYTE 0
0927 0B6D   0D  PHY1   BYTE CR,LF
     0B6E   0A
0928 0B6F   54         TEXT 'TRACK NUMBER (0 - >4C OR 0 -  >22) ? '
0929 0B94   00         BYTE 0
0930 0B95   0D  PHY2   BYTE CR,LF
     0B96   0A
0931 0B97   53         TEXT 'SURFACE (0 OR 1,DEF=0) ? '
0932 0BB0   00         BYTE 0
0933 0BB1   0D  PHY3   BYTE CR,LF
     0BB2   0A
0934 0BB3   53         TEXT 'SECTOR (1 - >1A OR 1 - >10) ? '
0935 0BD1   00         BYTE 0
0936 0BD2   0D  STATMG BYTE CR,LF
     0BD3   0A
0937 0BD4   43         TEXT 'CHECK STATUS (Y/N,DEF=Y) ? '
0938 0BEF   00         BYTE 0
0939 0BF0   0D  LOOPMG BYTE CR,LF
     0BF1   0A
0940 0BF2   4C         TEXT 'LOOP THRU COMMAND CHAIN (Y/N,DEF=N) ? '
0941 0C18   00         BYTE 0
0942 0C19   0D  EXMSG  BYTE CR,LF
     0C1A   0A
0943 0C1B   45         TEXT 'EXECUTE (Y/N,DEF=Y) ? '
0944 0C31   00         BYTE 0
```

```
                 ******************************************************
                 *            BLOCK TRANSFER ROUTINE                  *
                 *                                                    *
                 * This routine is entered from the /425 demonstration *
                 * software command scanner by entering the character 'X'. *
                 * This routine is used to move a block of data from one *
                 * memory loacation to another. The user will be prompted *
                 * for the source address, destination address, and the *
                 * number of bytes to be transferred. All entries must be *
                 * an even number i.e. word boundry.                  *
                 *                                                    *
                 * PROMPT:  ??X                                       *
                 * '        SOURCE = XXXX  DEST = YYYY LENGTH = LLLL  *
                 *                                                    *
                 ******************************************************
0C32                    EVEN
0C32 C26D        XFER00 MOV  @18(R13),R9  GET THE RAM BASE
     0C34 0012
0C36 C2A9               MOV  @ROMBAS(R9),R10 GET EPROM BASE
     0C38 0224
0C3A 0204               LI   R4,XFERM1    GET ADDR OF MESG 1
     0C3C 0C60'
0C3E 06AA               BL   @GETDAT(R10) GO GET THE DATA
     0C40 0D4C'
0C42 C141               MOV  R1,R5        PUT 1st ADDR IN R5
0C44 0204               LI   R4,XFERM2    GET ADDR OF MESG 2
     0C46 0C6C'
0C48 06AA               BL   @GETDAT(R10) GO GET THE DATA
     0C4A 0D4C'
0C4C C1C1               MOV  R1,R7        PUT 2nd ADDR IN R7
0C4E 0203               LI   R3,MOV1      SET UP EXECUTE CODE
     0C50 CDF5
0C52 0204        XFER02 LI   R4,XFERM3    GET ADDR OF MESG 3
     0C54 0C76'
0C56 06AA               BL   @GETDAT(R10) GO GET THE LENGTH
     0C58 0D4C'
0C5A C181               MOV  R1,R6        PUT THE LENGTH IN R6
0C5C 13FA               JEQ  XFER02       IF 0, ASK AGAIN
0C5E 1030               JMP  INIT15       GO TO EXECUTE CODE
                 *
                 *=====> TRANSFER & INITIALIZE MESSAGES
                 *
0C60   0D        XFERM1 BYTE CR,LF
     0C61   0A
0C62   53               TEXT 'SOURCE = '
0C6B   00               BYTE 0
0C6C   20        XFERM2 TEXT '  DEST = '
0C75   00               BYTE 0
0C76   20        XFERM3 TEXT '  LENGTH = '
0C81   00               BYTE 0
0C82   20        INITM1 TEXT '  DATA = '
0C8B   00               BYTE 0
```

```
                 ***********************************************************
                 *          MEMORY INITIALIZE ROUTINE                      *
                 *                                                         *
                 * This routine is entered form the /425 demonstration     *
                 * software command scanner by entering the character 'P'. *
                 * This routine allows the user to initialize memory with  *
                 * a data pattern. The user will be prompted for a memory  *
                 * start address, byte count, and data pattern. All        *
                 * entries are in hexadecimal. Shown below is the prompting *
                 * for this routine:                                       *
                 *                                                         *
                 * PROMPT: ??P                                             *
                 *         ADDR1 = XXXX  LENGTH = LLLL  DATA = PPPP        *
                 *                                                         *
                 ***********************************************************
0C8C                     EVEN
0C8C C26D        INIT10 MOV  @18(R13),R9  GET THE RAM BASE
0C8E 0012
0C90 C2A9               MOV  @ROMBAS(R9),R10 GET THE EPROM BASE
0C92 0224
0C94 0204               LI   R4,VFYM1     GET ADDR OF MESG 1
0C96 0D2C'
0C98 06AA               BL   @GETDAT(R10) GO GET THE DATA
0C9A 0D4C'
0C9C C141               MOV  R1,R5        PUT START ADDR IN R5
0C9E 0203               LI   R3,MOV2      SET UP EXECUTE CODE
0CA0 CD47
0CA2 0204        INIT14 LI   R4,XFERM3    GET ADDR OF MESG 2
0CA4 0C76'
0CA6 06AA               BL   @GETDAT(R10) GO GET THE DATA
0CA8 0D4C'
0CAA C181               MOV  R1,R6        PUT LENGTH IN R6
0CAC 13FA               JEQ  INIT14       IF 0, ASK AGAIN
0CAE 04C1        INIT12 CLR  R1           CLEAR THE INPUT REG
0CB0 2FAA               PRINT @INITM1(R10) ASK FOR DATA PATTERN
0CB2 0C82'
0CB4 06AA               BL   @INSUB(R10)  MOVE CODE & EXECUTE
0CB6 0D64'
0CB8 2E41               HEXIN R1
0CBA 0CBE'              DATA INIT13       PROCESS THE 0
0CBC 0CAE'              DATA INIT12       IF NOT HEX, ASK AGAIN
0CBE C1C1        INIT13 MOV  R1,R7        PUT PATTERN IN R7
0CC0 2FAA        INIT15 PRINT @CRLF(R10)  DO A CARRIAGE RETURN
0CC2 0270'
0CC4 0483        INIT11 X    R3           EXECUTE A MOVE COMMAND
0CC6 0646               DECT R6           DECREMENT THE LENGTH
0CC8 16FD               JNE  INIT11       IF NOT DONE, JUMP BACK
0CCA 0380               RTWP              RETURN TO SCANNER
```

```
                 ***********************************************************
                 *          BLOCK TO BLOCK COMPARE                         *
                 *                                                         *
                 * This routine is entered from the /425 demonstration     *
                 * software command scanner by entering the character 'V'. *
                 * This routine allows the user to verify the contents of  *
                 * two blocks of memory by comparing the contents. If the  *
                 * corresponding memory words are not equal an error       *
                 * message is printed. The user will be prompted for the   *
                 * starting addresses of the two blocks of data and the    *
                 * number of bytes to be compared. All entries are in      *
                 * are in hexadecimal and the data is compared on word     *
                 * boundries. Shown below is the prompting and error       *
                 * print out;                                              *
                 *                                                         *
                 * PROMPT: ??V                                             *
                 *         ADDR1 = XXXX  ADDR2 = YYYY  LENGTH = LLLL       *
                 *                                                         *
                 * ** ERROR                                                *
                 * ADDR1 = XX42  DATA = ABCD  ADDR2 = YY42  DATA = ABCC    *
                 *                                                         *
                 ***********************************************************
0CCC                     EVEN
0CCC C26D        VEFY00 MOV  @18(R13),R9  GET THE RAM BASE
0CCE 0012
0CD0 C2A9               MOV  @ROMBAS(R9),R10 GET THE ROM BASE
0CD2 0224
0CD4 0204               LI   R4,VFYM1     GET ADDR OF MESG 1
0CD6 0D2C'
0CD8 06AA               BL   @GETDAT(R10) GET 1st ADDRESS
0CDA 0D4C'
0CDC C141               MOV  R1,R5        PUT 1st ADDR IN R5
0CDE 0204               LI   R4,VFYM2     GET ADDR OF MESG 2
0CE0 0D38'
0CE2 06AA               BL   @GETDAT(R10) GET 2nd ADDRESS
0CE4 0D4C'
0CE6 C181               MOV  R1,R6        PUT 2nd ADDR IN R6
0CE8 0204        VEFY03 LI   R4,XFERM3    GET ADDR OF MESG 3
0CEA 0C76'
0CEC 06AA               BL   @GETDAT(R10) GET THE LENGTH
0CEE 0D4C'
0CF0 C1C1               MOV  R1,R7        PUT LENGTH IN R7
0CF2 13FA               JEQ  VEFY03       IF 0, ASK AGAIN
0CF4 2FAA               PRINT @CRLF(R10)  DO A CARRIAGE RETURN
0CF6 0270'
0CF8 8DB5        VEFY01 C    *R5+,*R6+    DOES THIS WORD COMPARE ?
0CFA 1603               JNE  VEFY02       IF NO, JUMP
0CFC 0647               DECT R7           DECREMENT THE LENGTH
0CFE 16FC               JNE  VEFY01       IF NOT DONE, KEEP CHECKING
0D00 0380               RTWP              RETURN TO SCANNER
0D02 2FAA        VEFY02 PRINT @VFYM3(R10) PRINT OUT ERROR
0D04 0D41'
0D06 0645               DECT R5           BACK UP POINTERS
0D08 0646               DECT R6
0D0A 2FAA               PRINT @VFYM1(R10) PUT OUT 1st ADDR
0D0C 0D2C'
0D0E 2E85               HEXOUT R5
0D10 2FAA               PRINT @INITM1(R10) PUT OUT 1st DATA
0D12 0C82'
0D14 2E85               HEXOUT *R5+
```

```
0D16 2FAA          PRINT @VFYM2A(R10) PUT OUT 2nd ADDR
0D18 0D35'
0D1A 2E86          HEXOUT R6
0D1C 2FAA          PRINT @INITM1(R10) PUT OUT 2nd DATA
0D1E 0C82'
0D20 2EB6          HEXOUT *R6+
0D22 2FAA          PRINT @CRLF(R10)  DO A CARRIAGE RETURN
0D24 0270'
0D26 0647          DECT R7           DECREMENT THE LENGTH
0D28 16E7          JNE  VEFY01       IF NOT DONE, KEEP CHECKING
0D2A 0380          RTWP              RETURN TO SCANNER
                 ********************************************************
                 *            VERIFY MESSAGES                           *
                 ********************************************************
0D2C   41  VFYM1   TEXT 'ADDR1 = '
0D34   00          BYTE 0
0D35   20  VFYM2A  BYTE >20,>20,>20
0D36   20
0D37   20
0D38   41  VFYM2   TEXT 'ADDR2 = '
0D40   00          BYTE 0
0D41   2A  VFYM3   TEXT '** ERROR'
0D49   0D          BYTE CR,LF,0
0D4A   0A
0D4B   00
                 ********************************************************
                 *           GET DATA SUBROUTINE                        *
                 ********************************************************
0D4C               EVEN
0D4C A10A  GETDAT  A    R10,R4       ADJUST EPROM BASE
0D4E C00B          MOV  R11,R0       SAVE RETURN
0D50 04C1  GETDA1  CLR  R1           CLEAR INPUT REGISTER
0D52 2F94          PRINT *R4         PRINT THE PROMPT
0D54 06AA          BL   @INSUB(R10)  MOVE CODE & EXECUTE
0D56 0D64'
0D58 2E41          HEXIN R1          INPUT THE HEX VALUE
0D5A 0D5E'         DATA GETDA2       PROCESS 0
0D5C 0D50'         DATA GETDA1       IF NOT HEX, ASK AGAIN
0D5E 0241  GETDA2  ANDI R1,>FFFE     STRIP OFF BYTE BIT
0D60 FFFE
0D62 0450          B    *R0          RETURN TO UTILITY
0D64 020C  INSUB   LI   R12,EXAREA   SET UP EXECUTE ADDR
0D66 FD20
0D68 CF3B          MOV  *R11+,*R12+  MOVE 'HEXIN' OPCODE
0D6A C73B          MOV  *R11+,*R12   MOVE ADDR1
0D6C AF0A          A    R10,*R12+    ADD IN ROM BASE
0D6E C73B          MOV  *R11+,*R12   MOVE ADDR2
0D70 AF0A          A    R10,*R12+    ADD IN ROM BASE
0D72 C72A          MOV  @RTCMD(R10),*R12 SET UP 'RT'
0D74 0D7A'
0D76 0460          B    @EXAREA      GO EXECUTE CODE
0D78 FD20
0D7A 045B  RTCMD   RT                NEEDED OPCODE
```

```
                 ************************************************************
                 *                    FORMAT DISKETTE COMMAND               *
                 *                                                          *
                 * This command formats a diskette. The user is asked for   *
                 * the size of diskette and the format to be used. Shown    *
                 * below is the prompting:                                  *
                 *                                                          *
                 *   ??F                                                    *
                 *   NUMBER OF SIDES (1 OR 2,DEF=1) ?                       *
                 *   SIZE OF DISKETTE (8 OR 5 IN,DEF=8 ) ?                  *
                 *   FORMAT 0 = IBM SINGLE, 1 = IBM DOUBLE, 2 = TI DOUBLE   *
                 *              (DEF=0) ?                                   *
                 ************************************************************
0D7C                   EVEN
0D7C C26D        FORMAT MOV  @18(R13),R9  GET THE RAM BASE
0D7E 0012
0D80 C2A9              MOV  @ROMBAS(R9),R10 GET THE EPROM BASE
0D82 0224
0D84 8A6A              C    @INTVAL(R10),@INITFL(R9) HAVE WE INITED ?
0D86 006E'
0D88 0210
0D8A 1307              JEQ  FORM10       IF YES, GO ON
0D8C 0201              LI   R1,WP3       SET UP WP
0D8E 01B0
0D90 A049              A    R9,R1        ADJUST FOR RAM
0D92 0202              LI   R2,INIT      GET PC ADDR
0D94 02CE'
0D96 A08A              A    R10,R2       ADJUST ROM BASE
0D98 0401              BLWP R1           INITIALIZE SYSTEM
0D9A 0208        FORM10 LI   R8,MINMAX   SET UP MAX TRK OF MINI
0D9C 0022
0D9E 04E9              CLR  @LOOPFL(R9)  TURN OFF LOOP FLAG
0DA0 021E
0DA2 0729              SETO @STATFL(R9)  SET STATUS CHECKER FLAG ON
0DA4 0222
0DA6 0206        FORM11 LI   R6,1         SET DEFAULT SIDE TO 1
0DA8 0001
0DAA 2FAA              PRINT @SIDMSG(R10) PRINT SIDE MESSAGE
0DAC 0FD5'
0DAE 06AA              BL   @INSUB(R10)  MOVE CODE & EXECUTE
0DB0 0D64'
0DB2 2E46              HEXIN R6          GET # OF SIDES
0DB4 0DC4'             DATA FORM20       IF 0, ASK AGAIN
0DB6 0DA6'             DATA FORM11       IF NOT HEX, ASK AGAIN
0DB8 0286              CI   R6,1         IS IT A 1 ?
0DBA 0001
0DBC 1303              JEQ  FORM20       IF YES, JUMP
0DBE 0286              CI   R6,2         IS IT A 2 ?
0DC0 0002
0DC2 16F1              JNE  FORM11       IF NO, JUMP BACK
0DC4 C046        FORM20 MOV  R6,R1        SAVE # SIDES
0DC6 0206        FORMA1 LI   R6,8         SET DEFAULT SIZE TO 8
0DC8 0008
0DCA 2FAA              PRINT @SIZMSG(R10) PRINT SIZE PROMPT
0DCC 0F6E'
0DCE 06AA              BL   @INSUB(R10)  MOVE CODE & EXECUTE
0DD0 0D64'
0DD2 2E46              HEXIN R6          GET HEX INPUT
0DD4 0DE8'             DATA FORM19       IF 0, ASK AGAIN
0DD6 0DC6'             DATA FORMA1       IF NOT HEX, ASK AGAIN
```

```
1163 0DD8 2FAA          PRINT @CRLF(R10)  DO A CARRIAGE RETURN
     0DDA 0270'
1164 0DDC 0286          CI   R6,5         IS IT 5 ?
     0DDE 0005
1165 0DE0 1305          JEQ  FORM12       IF YES, JUMP
1166 0DE2 0286          CI   R6,8         IS IT 8 ?
     0DE4 0008
1167 0DE6 16EF          JNE  FORMA1       IF NO, ASK AGAIN
1168 0DE8 0208  FORM19  LI   R8,STDMAX    SET UP MAX SIZE FOR STD
     0DEA 004C
1169 0DEC 0286  FORM12  CI   R6,5         MINI ?
     0DEE 0005
1170 0DF0 1603          JNE  FORMA2       IF NO, JUMP
1171 0DF2 0281          CI   R1,2         2 SIDES ?
     0DF4 0002
1172 0DF6 13D7          JEQ  FORM11       JUMP IF ILLEGAL COMBO
1173 0DF8 04C4  FORMA2  CLR  R4           SET DEFAULT TO IBM SINGLE
1174 0DFA 2FAA          PRINT @FORMG1(R10) ASK FOR FORMAT TYPE
     0DFC 0F92'
1175 0DFE 06AA          BL   @INSUB(R10)  MOVE CODE & EXECUTE
     0E00 0D64'
1176 0E02 2E44          HEXIN R4          GET THE VALUE
1177 0E04 0E08'         DATA FORMA3       PROCESS THE 0
1178 0E06 0DF8'         DATA FORMA2       IF NOT HEX, ASK AGAIN
1179 0E08 C301  FORMA3  MOV  R1,R12       MOVE SIDE
1180 0E0A 020B          LI   R11,1        SET UP SECTOR #
     0E0C 0001
1181 0E0E 2FAA          PRINT @CRLF(R10)  DO A CARRIAGE RETURN
     0E10 0270'
1182 0E12 0284          CI   R4,2         IS THE # VALID ?
     0E14 0002
1183 0E16 15F0          JGT  FORMA2       IF NO, JUMP BACK
1184 0E18 0207          LI   R7,MINTA1    GET START OF MINI TABLE
     0E1A 0F1E'
1185 0E1C A1CA          A    R10,R7       ADD IN ROM BASE
1186 0E1E 0288          CI   R8,STDMAX    IS THIS STD ?
     0E20 004C
1187 0E22 1306          JEQ  FORMA4       IF YES, JUMP
1188 0E24 0284          CI   R4,2         IS THIS TI ?
     0E26 0002
1189 0E28 13E7          JEQ  FORMA2       IF YES, ILLEGAL CMD, ASK AGAIN
1190 0E2A C104          MOV  R4,R4        IS THIS IBM SINGLE ?
1191 0E2C 130D          JEQ  FORMA6       IF YES, JUMP
1192 0E2E 100A          JMP  FORMA5       GO ADJUST ADDR
1193 0E30 0227  FORMA4  AI   R7,32        BUMP DOWN 2 TABLES
     0E32 0020
1194 0E34 C104          MOV  R4,R4        IBM SINGLE ?
1195 0E36 1308          JEQ  FORMA6       IF YES, JUMP
1196 0E38 0227          AI   R7,16        BUMP DOWN TO IBM DOUBLE
     0E3A 0010
1197 0E3C 0284          CI   R4,1         IBM DOUBLE ?
     0E3E 0001
1198 0E40 1303          JEQ  FORMA6       IF YES, JUMP
1199 0E42 04CB          CLR  R11          CLEAR SECTOR, TI DOUBLE
1200 0E44 0227  FORMA5  AI   R7,16        ADJUST TABLE ADDR
     0E46 0010
1201 0E48 0201  FORMA6  LI   R1,BUF       GET ADDR OF BUFFER
     0E4A 00C8
1202 0E4C 0202          LI   R2,DEFCMD    GET ADDR OF DEFINE DRIVE CMD
     0E4E 0EF6'
```

```
0E50 A049          A    R9,R1          ADJUST RAM BASE
0E52 A08A          A    R10,R2         ADJUST ROM BASE
0E54 0200          LI   R0,20          SET UP CTR
0E56 0014
0E58 CC72  FORMA7  MOV  *R2+,*R1+      MOVE CMD LIST
0E5A 0640          DECT R0             DEC LENGTH
0E5C 16FD          JNE  FORMA7         IF NOT DONE, KEEP MOVING
0E5E C841          MOV  R1,@-6(R1)     PUT ADDR IN LIST
0E60 FFFA
0E62 C081          MOV  R1,R2          SAVE ADDR OF DATA
0E64 0200          LI   R0,8           SET UP LOOP CTR
0E66 0008
0E68 CC77  FORM17  MOV  *R7+,*R1+      MOVE THE DATA INTO RAM
0E6A 0600          DEC  R0             DEC THE CTR
0E6C 16FD          JNE  FORM17         IF NOT DONE, JUMP BACK
0E6E 0729          SETO @UNITFL(R9)    INSERT THE UNIT
0E70 0220
0E72 0203          LI   R3,WP3         SET UP BLWP VECTOR
0E74 01B0
0E76 0204          LI   R4,COMISR
0E78 05A4'
0E7A A0C9          A    R9,R3          ADJUST FOR RAM
0E7C A10A          A    R10,R4         ADJUST ROM BASE
0E7E 0200          LI   R0,BUF         SET UP PARAMETER
0E80 00C8
0E82 A009          A    R9,R0
0E84 028C          CI   R12,2          TWO SIDED ?
0E86 0002
0E88 1601          JNE  FORM13         IF NO, JUMP
0E8A 0592          INC  *R2            CHANGE TABLE
0E8C 0403  FORM13  BLWP R3             ISSUE CMD
0E8E C1A9          MOV  @ERRFLG(R9),R6 DID WE GET AN ERROR ?
0E90 021A
0E92 1630          JNE  FORM18         IF YES, EXIT
0E94 04C6          CLR  R6             SET UP COUNTER
0E96 04C7          CLR  R7             SET UP SIDE
0E98 028C          CI   R12,2          2 SIDES ?
0E9A 0002
0E9C 1602          JNE  FORMA8         IF NO, JUMP
0E9E 0207          LI   R7,>100        INIT TO SIDE 2
0EA0 0100
0EA2 0200  FORMA8  LI   R0,20          SET UP LOOP CTR
0EA4 0014
0EA6 0201          LI   R1,BUF         GET ADDR OF BUFFER
0EA8 00C8
0EAA 0202          LI   R2,FORCMD      GET ADDR OF FORMAT CMD
0EAC 0F0A'
0EAE A049          A    R9,R1          ADJUST FOR RAM
0EB0 A08A          A    R10,R2         ADJUST FOR ROM
0EB2 CC72  FORMA9  MOV  *R2+,*R1+      MOVE DATA
0EB4 0640          DECT R0             DEC THE CTR
0EB6 16FD          JNE  FORMA9         IF NOT DONE, MOVE MORE
0EB8 E846          SOC  R6,@-14(R1)    PUT TRK #
0EBA FFF2
0EBC C84B          MOV  R11,@-12(R1)   PUT IN SECTOR #
0EBE FFF4
0EC0 028C          CI   R12,2          DOUBLE SIDED ?
0EC2 0002
0EC4 1606          JNE  FORM14         IF NO, JUMP
0EC6 0607          DEC  R7             WAS IT 0 ?
```

```
0EC8 1102          JLT  FORM15        IF YES, JUMP
0ECA 04C7          CLR  R7            SET TO SIDE 0
0ECC 1002          JMP  FORM14        GO PUT IN SIDE
0ECE 0207  FORM15  LI   R7,>100       SET UP SIDE 2
0ED0 0100
0ED2 D847  FORM14  MOVB R7,@-12(R1)   PUT SIDE IM CMD
0ED4 FFF4
0ED6 C001          MOV  R1,R0         SET UP PARAMETER
0ED8 0220          AI   R0,-20        POINT TO WORD0
0EDA FFEC
0EDC 0403          BLWP R3            ISSUE FORMAT CMD
0EDE C029          MOV  @ERRFLG(R9),R0  DID WE GET AN ERROR ?
0EE0 021A
0EE2 1608          JNE  FORM18        IF YES, EXIT
0EE4 028C          CI   R12,2         2 SIDES ?
0EE6 0002
0EE8 1602          JNE  FORM16        IF NO, JUMP
0EEA C1C7          MOV  R7,R7         HAVE WE DONE BOTH SIDES ?
0EEC 13DA          JEQ  FORMA8        IF NO, GO BACK
0EEE 0586  FORM16  INC  R6            INC THE TRK #
0EF0 8206          C    R6,R8         ARE WE DONE
0EF2 12D7          JLE  FORMA8        IF NO, JUMP BACK
0EF4 0380  FORM18  RTWP               IF YES, EXIT
```

```
1265                *
1266                *======> COMMANDS
1267                *
1268 0EF6 0000  DEFCMD DATA 0,0,>1000,0,0,0
     0EF8 0000
     0EFA 1000
     0EFC 0000
     0EFE 0000
     0F00 0000
1269 0F02 000F         DATA >F,0,0,0
     0F04 0000
     0F06 0000
     0F08 0000
1270 0F0A 0000  FORCMD DATA 0,0,>900,>8000
     0F0C 0000
     0F0E 0900
     0F10 8000
1271 0F12 0000         DATA 0,0
     0F14 0000
1272 0F16 000F         DATA >F,0,0,0
     0F18 0000
     0F1A 0000
     0F1C 0000
1273 0F1E 0001  MINTA1 DATA 1,35,>FA0,>3E8
     0F20 0023
     0F22 0FA0
     0F24 03E8
1274 0F26 1D4C         DATA >1D4C,>3E8,0,>4080
     0F28 03E8
     0F2A 0000
     0F2C 4080
1275 0F2E 0001  MINTA2 DATA 1,35,>FA0,>3E8
     0F30 0023
     0F32 0FA0
     0F34 03E8
1276 0F36 1D4C         DATA >1D4C,>3E8,>1000,>4100
     0F38 03E8
     0F3A 1000
     0F3C 4100
1277 0F3E 0101  STDTA1 DATA >101,77,1000,1500
     0F40 004D
     0F42 03E8
     0F44 05DC
1278 0F46 0DAC         DATA 3500,1000,0,>6880
     0F48 03E8
     0F4A 0000
     0F4C 6880
1279 0F4E 0101  STDTA2 DATA >101,77,1000,1500
     0F50 004D
     0F52 03E8
     0F54 05DC
1280 0F56 0DAC         DATA 3500,1000,>1000,>6900
     0F58 03E8
     0F5A 1000
     0F5C 6900
1281 0F5E 0101  STDTA3 DATA >101,77,1000,1500
     0F60 004D
     0F62 03E8
     0F64 05DC
1282 0F66 0DAC         DATA 3500,1000,>1100,>6920
```

```
     0F68 03E8
     0F6A 1100
     0F6C 6920
 1283            ********************************************************
 1284            *          FORMAT MESSAGES                             *
 1285            ********************************************************
 1286 0F6E   0D  SIZMSG BYTE CR,LF
      0F6F   0A
 1287 0F70   53         TEXT 'SIZE OF DISK (8 OR 5 IN,DEF=8) ? '
 1288 0F91   00         BYTE 0
 1289 0F92   0D  FORMG1 BYTE CR,LF
      0F93   0A
 1290 0F94   46         TEXT 'FORMAT: 0 = IBM SINGLE, 1 = IBM DOUBLE'
 1291 0FBA   2C         TEXT ', 2 = TI DOUBLE (DEF=0) ? '
 1292 0FD4   00         BYTE 0
 1293 0FD5   0D  SIDMSG BYTE CR,LF
      0FD6   0A
 1294 0FD7   4E         TEXT 'NUM OF SIDES (1 OR 2,DEF=1) ? '
 1295 0FF5   00         BYTE 0
 1296 0FF6 2401  CHKSUM DATA >2401
 1297 0FF8       ENDEMO
 1298                   END
NO ERRORS,     NO WARNINGS
```

INDEX

In the page number list of this alphabetical index, subject matter is covered by a table if a T precedes the page number, or is covered by a figure if an F precedes the page number.