MXV22M Disk Controller Manual

Micro Technology, Inc.

MXV22M
Disk Controller Manual

810010-202 REV. C

OCTOBER, 1984

MICRO TECHNOLOGY, INCORPORATED

Preface

The purpose of this manual is to provide the user adequate information to configure and operate the MXV22M floppy disk controller. The information provided should clarify the controller connection to any Shugart compatible drive and assist in the selection of associated interface options. Both register definition and command protocol are provided for reference and as an aid in development of user software. Operational procedures outline the use of the controller features as well as explaining operation in an RT-11 software environment.

Contents

PAGE

SECTION 1 GENERAL INFORMATION 1

INTRODUCTION 1
COMPONENTS 2
COMPATIBILITY 2
5 1/4 Inch Logical Track Format 2
Sector Header Field 4
Data Field 4
Recording Scheme 5
Double Frequency (FM) 6
DEC Modified MFM 6
Cyclic Redundancy Check 7
SPECIFICATIONS 8

SECTION 2 INSTALLATION 9

GENERAL 9
CONFIGURATION 9
Address Vector Selection 9
Device Interrupt Priority 10
Bootstrap 12
Write Precompensation 12
Step Rate Control 13
22 Bit Addressing 14
Miscellaneous Options 14
DRIVE CONFIGURATION 15
CABLING 15
CONTROLLER INSTALLATION 17
INITIAL OPERATION AND CHECKOUT 18

SECTION 3 FUNCTIONAL DESCRIPTION 21

GENERAL 21
REGISTER DEFINITIONS 21
MXVCS - Command and Status Register (177170(8)) 21
MXVDB - Data Buffer (177172(8)) 23
Extended Status Register 27
COMMAND PROTOCOL 28
Fill Buffer (000) 28
Empty Buffer (001) 29

Write Sector (010) 30
Read Sector (011) 31
Set Media Density (100) 32
Read Status (101) 33
Write Deleted Data Sector (110) 33
Read Error Code (111) 33

SECTION 4 CONTROLLER OPERATIONS 35

GENERAL 35
BOOTSTRAPPING THE CONTROLLER 35
Bootstrap Operation 36
FILL/WRITE OPERATIONS 38
READ/EMPTY OPERATIONS 39
WRITE CURRENT CONTROL 40
POWER FAIL PROTECTION 40

SECTION 5 SOFTWARE CONSIDERATIONS 41

GENERAL 41
OPERATION USING RT-11 41
Modifying RX02 Driver for RT11 42
OPERATION WITH LAYERED PRODUCTS 42
OPERATION WITH RSX11M 42
Modifying RX02 Driver RSX11M 43
OPERATION WITH OTHER DEC SOFTWARE 43
OPERATION WITH DEC DIAGNOSTIC 43
Exceptions 43

Appendix

APPENDIX A 45

APPENDIX B 47

APPENDIX C 49

APPENDIX F 51

Tables

Table 1-1: Address Marks 5
Table 1-2: Standard MFM Encoding 7
Table 1-3: Modifying Algorithm 7
Table 2-1: Address/Vector Option Configuration 10
Table 2-2: Priority Level Configuration. 10
Table 2-3: Bootstrap Option. 12
Table 2-4: Write Precompensation 13
Table 2-5: Step Rate 13
Table 2-6: 22 Bit Addressing 14
Table 2-7: Miscellaneous Options 14
Table 2-8: Tandon TM 100-4 Drive Configuration 15
Table 4-1: Command Word Formats. 37

Figures

Figure 1-1: 5 1/4-Inch Logical Track Format. 3
Figure 1-2: FM Recording Characteristics 6
Figure 1-3: MFM Recording Characteristics. 6
Figure 2-1: Configuration Jumper Locations 11
Figure 2-2: Drive/Controller Cabling 16
Figure 2-3: Connector Pin Definitions. 17
Figure 4-2: Write Data Subroutines 38
Figure 4-3: Read Data Subroutines. 39

Section 1

General Information

1. INTRODUCTION

The MXV22M is a dual density controller compatible with the DEC* RX02 floppy disk system. Configured with any Shugart compatible drive, it is a direct replacement for the RXV21 subsystem. The controller provides either single density encoding or double density encoding, providing 512K bytes of storage on a single diskette. When configured with two drive, each drive may operate at a different density.

All electronics are contained on one dual-wide board which plugs directly into any standard LSI-11 backplane and interfaces through a 34 conductor ribbon cable to any Shugart compatible drive (or a 50 conductor cable, see section 2.3). All controllers are 100% tested and ready for plug in and operation. The controller is configured for the standard device address 177170(8) and interrupt vector 264(8). The interrupt level is factory set to level four. Features include:

22 bit addressing

Transparent firmware bootstrap automatically loads either single or double density diskettes.

Formatting capability permits writing sector headers, checking the written headers, and writing the data fields in the user selected density.

Jumpers allow user selection of both the alternate address and vector.

Jumper selectable four-level drive interrupt priority compatible with the LS-11/23.

Provides power fail protection for data integrity.

Write current control signal for tracks greater than forty-three.

Write precompensation for reduced error rates.

*DEC, PDP-11, LSI-11, and QBUS are registered trademark of Digital Equipment Corporation.

1.1. COMPONENTS

The controller is provided with the following components:

Floppy disk controller
Manual

1.2. COMPATIBILITY

This section discusses the aspects of hardware, software and media compatibility with Digital Equipment's RX02 system. The information will aid the user in data interchanging with foreign systems.

Hardware

The controller is compatible with the LSI-11, LSI-11/2 and LSI-11/23 processors. All circuitry is contained on one dual-wide board that plugs directly into any standard LSI-11 backplane. Alternate address selection and a four-level device interrupt priority scheme provide the user added flexibility for expanded system configurations. Shugart compatible drive logic is interfaced through a 34-pin ribbon connector. The connector pins are compatible with both the Shugart SA460 and Tandon TM 100-4.

Software

The MXV22M is completely compatible with RXV21 register definition and command protocol. All DEC-supplied software designed to operate with the RX02 system will operate with the controller without modification.

Drives

A double sided 96 TPI 5 1/4" mini-floppy drive must be used to attain 512K bytes of storage capacity. Two drives which are available are the Shugart SA460 and the Tandon T100-4.

Media

Either preformatted or blank soft sectored diskettes may be used with the controller. The following list summarizes the suggested media.

VERBATIM	MD 550-01-18188
3M	DC 051111

1.2.1. 5 1/4 Inch Logical Track Format

The diskette surface is divided into 77 concentric tracks numbered 0-76. Each cylinder consists of two tracks. The track on side 0 contains sectors 1-13 while the track on side 1 contains sectors 14-26. The track begins and ends at the index mark. The track is formatted in such a way that this "soft" index is preceded by the leading edge of the physical index hole in the diskette. Following the physical index are 40 bytes of "FF" data, 56 bytes of "0" data, and the index address mark indicating the beginning of the track. Following the index address mark is the post index gap consisting of 26 bytes of "FF" data and 6 bytes of "0" data. The next field is the sector header for the first sector. On side 0 this is 1. On side 1 this sector is sector 14. Following the sector header is the ID gap consisting of 11 bytes of "FF" data and 6 bytes of "0" data. The next field is the data record for the sector. Following the data field is the data gap consisting of 28 bytes of "FF" data and 6 bytes of "0" data. This field leads to the next sector header. Following the 13th data record on side 0 and the 26th data record on side 1 is the pre-index gap consisting of approximately 620 bytes of "FF" data.

Each track is formatted in the above manner. Refer to Figure 1-1. The sector header field of each sector contains information describing both the sector and the track number. All the above fields are recorded in FM except as noted in the following sections.

```
             ---------
  INDEX   |          |
   -------          ---------

 ---------------------------------------------------------//----------
|DATA|GAP4|I|GAP1|ID  |GAP2|DATA|GAP3|ID  |    |DATA|    |  |ID  |    |
|REC.|666 |A|32  |REC.|17  |REC.|34  |REC.|GAP2|REC.|GAP3|  |REC.|GAP2|
|13  |BYTE|M|BYTE|1   |BYTE|1   |BYTE|2   |    |2   |    |  |13  |    |
|    |FM  | |FM  |    |FM  |    |FM  |    |    |    |    |  |    |    |
 ---------------------------------------------------------//----------
      |        |    |   |        |         |
      |  ----------  |  ----------   --------------  |
      |  |26BYT|6BYT||||11BYT|6BYT||1BYT|27BYT|6BYT|||
      |  | FF  | 00 ||| FF  | 00 || FF | FF  | 00 |||
      |  ----------  |  ----------   --------------  |
      |              |      |--WRITE---|              |
      |              |         SPLICE                 |
 -----------   ------------------------------    -----------------
|660BYT|6BYT|  |IDAM|TRK|1BYT|SECT|1BYT|CRC|CRC|  |DAM| USER |CRC|CRC|
|  FF  | 00 |  |    |   | 00 |    | 00 | 1 | 2 |  |   | DATA | 1 | 2 |
 -----------   ------------------------------    ---|------|--------
                                               FM-->|128BYT|<--
                                               MFM->|256BYT|<--
```

Figure 1-1: 5 1/4-Inch Logical Track Format

1.2.2. Sector Header Field

The header field consists of 7 bytes of information. Preceding the header is a field of 6 bytes of "zero" data for synchronization. The header and this preamble are always recorded in FM.

1. Byte 1. ID Address Mark - A unique mark consisting of 1 byte of FE (hex) data with three missing clock-transitions using a C7 (hex) clock pattern. This mark is decoded by the controller and indicates the start of the sector header.
2. Byte 2. Track Address - This byte indicates the absolute (0-114(8)) track address. Each sector contains this track information to locate its position on one of the 77 tracks.
3. Byte 3. "Zero"
4. Byte 4. Sector Address - This byte indicates the absolute (1-32(8)) sector address. Each sector contains this information to identify its position on the track.
5. Byte 5. "Zero"
6. Byte 6, 7. CRC - This is the 16 bit cyclic redundancy character and is calculated for each header from the first 5 bytes of information, using the IBM 3740 polynomial. (Refer to Cyclic Redundancy Check, Section 1.2.7.).

1.2.3. Data Field

The data field consists of either 131(10) or 259(10) bytes of information depending upon the recording method. Preceding the data field is a field of 6 bytes of "zero" data for synchronization.

The preamble and data address mark are always written in FM. The user data and CRC character are either written in FM or MFM modified, depending upon the formatted diskette density.

1. Byte 1. Data Address Mark - A unique mark consisting of a data byte (see Table 1-1) with three missing clock transitions using a C7 (hex) clock pattern. This byte is always written in FM and is decoded by the controller to indicate the start of the data field, its recording method (FM vs MFM), and if the field is a deleted data field.

ADDRESS MARK	INDICATED DENSITY	DATA	CLOCK
INDEX	NA	FC	D7
ID	NA	FE	C7
DATA	FM	FB	C7
	Modified	FD	C7
DELETED DATA	FM	F8	C7
	Modified	F9	C7

Table 1-1: Address Marks

2. Bytes 2-129 (FM) or Bytes 2-257 (MFM Modified). User Data. This field is recorded in either FM or MFM modified. Depending upon the encoding scheme, either 128 or 256 bytes of information can be stored.

3. Bytes 130-131 or 258-259. CRC - This is the 16 bit cyclic redundancy character and is calculated for each data field from the first 129 or 257 bytes of information using the IBM 3740 polynomial. (Refer to Cyclic Redundancy Check, Section 1.2.7). These bytes are recorded with the same encoding scheme as the data field.

1.2.4. Recording Scheme

Two recording schemes are used by the MX22M: double frequency (FM) and DEC modified Miller code (MFM). FM is used for single density recording.

1.2.5. Double Frequency (FM)

FM recording is characterized by a flux transition at the beginning of each bit cell which is commonly termed a clock pulse or transition as shown in Figure 1-2. A logic "one" is represented by a flux transition within the bit cell; a logic "zero" is represented by the lack of any flux transition within a bit cell. In FM the bit cell time is 8us.

```
       "0"      "0"       "1"       "0"       "1"       "0"
   _        _        _    _    _        _    _    _        _
  | |      | |      | |  | |  | |      | |  | |  | |      | |
--   -------  -------  --   --  -------  --   --  -------  --
   C        C        C    D    C        C    D    C        C
   |  bit   |
   |  cell  |
-->|  8us   |<--
```

Figure 1-2: FM Recording Characteristics

1.2.6. DEC Modified MFM

MFM recording consists of flux transitions for a logic "one" and no flux transitions for a logic "zero". A clock transition only occurs between two consecutive logic "zeros" as shown in Figure 1-3 below. The MFM bit cell time is 4us.

```
  "1"  "1"  "0"    "0"  "1"   "0"   "1"   "1"
   _    _        _      _           _     _
  | |  | |      | |    | |         | |   | |
---  ---  ------  ------  ---------  ---   ---
   D    D        C      D           D     D
   | bit |
   | cell|
-->| 4us |<--
```

Figure 1-3: MFM Recording Characteristics

Table 1-2 summarizes the standard MFM encoding algorithm.

DATA		ENCODED DATA		
DN-1	DN	DN-1	CN	DN
0	0	0	1	0
1	0	1	0	0
0	1	0	0	1
1	1	1	0	1

Table 1-2: Standard MFM Encoding

Because single density headers are used for both FM and MFM recording formats, and since certain MFM patterns map into single density address marks, a modified algorithm is used. The mapping occurs when a data pattern of exactly four consecutive "ones" is encoded. Whenever this pattern is encoded a special algorithm is applied. Table 1-3 defines the encoding algorithm for this special case.

DATA											
DN-5		DN-4		DN-3		DN-2		DN-1		DN	
0		1		1		1		1		0	
X	0	1	0	0	0	1	0	0	0	1	0
CN-5	DN-5	CN-4	DN-4	CN-3	DN-3	CN-2	DN-2	CN-1	DN-1	CN	DN
ENCODED DATA											

Table 1-3: Modifying Algorithm (Special Case)

When reading double density data fields the controller checks for a missing clock bit between two zero data cells, and if found, substitutes ones for the two zero data bits (generated by the special encoding algorithm).

1.2.7. Cyclic Redundancy Check

Each sector header field and data field has a two byte CRC character appended. This 16 bit character is the remainder that results when dividing the data bits [represented as a polynomial M(x)] by a generator polynomial G(x). The polynomial used for IBM 3740 is G(x) = X^16 + X^12 + X^5 + 1. For the sector header the data bits include byte 1 thru 5. For an FM data field the data bits include byte 1 thru byte 129. For an MFM data field the data bits include byte 1 thru byte 257.

1.3. SPECIFICATIONS

RECORDING TECHNIQUE:

Single Density	Modified IBM 3740 FM
Double Density	DEC Modified MFM

POWER REQUIREMENTS:

Voltage	Single 5V supply (from LSI-11 backplane)
Current	2.5A Typical

ENVIRONMENTAL

Temperature	0 degree - 45 degrees C
Humidity	10% - 95% non-condensing

Section 2 Installation

2. GENERAL

The controller is shipped with standard options configured. The standard address 177170(8) and vector 264(8) are set. The device interrupt priority is set to level four. The firmware bootstrap is disabled.

Most options are factory foil-etched to the most often used configuration. The foil jumpers must first be cut before the alternate jumpers are inserted. Refer to Tables 2-1, 2-2, and 2-3 for alternate options and Figure 2-1 for jumper location. Several of the options are selectable by using AMP 530153-2 pin jumpers. If these pin jumpers are not available use #30 wire wrap.

2.1. CONFIGURATION

2.1.1. Address Vector Selection

The controller is shipped with the DEC standard device address and vector assignments preset to 177170(8) and 264(8), respectively. Any change in these assignments would necessitate a change in system software. However, an alternate address and vector option is selectable and is defined as 177174(8) and 270(8), respectively. To select the alternate address/vector, first cut the foil between W15 and W16. Jumper W16 to W17 and jumper W8 to W9 as shown in Table 2-1.

OPTION	JUMPERS		
	15-16	16-17	8-9
Standard Address/Vector* 177170/264	IN	OUT	OUT
Alternate Address/Vector 177174/270	OUT	IN	IN
*Factory Preset			

Table 2-1: Address/Vector Option Configuration

2.1.2. Device Interrupt Priority

The MXV22M supports the four-level device interrupt priority scheme compatible with the LSI-11/23. The controller asserts interrupt requests and monitors higher level request lines during interrupt arbitration as described in Table 2-2. The level four request is always asserted by the controller, regardless of its priority, to maintain compatibility with the LSI-11 and LSI-11/2 processors.
The interrupt priority level is configured to level four at the factory. If a different interrupt level is desired the following foil-etched jumpers must be cut. Refer to Table 2-2 for the proper jumpers to insert for the desired priority level.

W19 - W20	W28 - W29
W22 - W23	W30 - W31
W24 - W25	

PTY LEV	ASRT	MON	JUMPERS									
			18-19	19-20	21-22	22-23	24-25	25-26	27-28	28-29	30-31	31-32
4*	4	5,6	Out	In	Out	In	In	Out	Out	In	In	Out
5	4,5	6	Out	In	In	Out	In	Out	Out	In	Out	In
6	4,6	7	In	Out	Out	In	Out	In	In	Out	In	Out
7	4,6,7	None	In	Out	In	Out	Out	In	In	Out	Out	In
*Factory Preset												

Table 2-2: Priority Level Configuration

```
           ---------
         \         /
          \       /    Factory Test
         ----------------------------+++++++++++++++++++++++++---
        |                   |       ||||||||||||||||||||||||   |
        |                   |                           W35<---+-Factory
        |                   +----->W1 W2 W3              *     | Test
        |                          *  *  *     *  *  *  *      |
        |                                     W44 45 33 34     |
        |                                                      |
        |                                                      |
        |                                               W6     |
        |                                               * <----+-Factory
        |                                               W5     | Test
        |                                               *      |
        |                                               W4     |
        |                                               *      |
        |                                                      |
        |                   W7                                 |
        |                   *                                  |
22 Bit--+----------->W47 46 W8                                 |
Address |             *  *  *                                  |
        |                   *                                  |
        |                   W9                                 |
        |                                                      |
Step----+----------->W36 39                                    |
Rate    |             *  *                                     |
        |             *  *                                     |
        |            W37 38                                    |
        |                                                      |
        |                                                      |  Address
Boot-   |            W41        W15<---------------------------+  Selection
Strap---+-----------> *          *                 *  *  *     |
        |              *  *  *   *                W14 13 12<---+- Write
        |             W42 43 17 16                             | Precom-
        |                                                      | pensation
        |                                   W20 19 18          |
        |                 W10                *  *  * <---------+- Interrupt
TBS7----+----------------> *                 *  *  *         / |  Priority
        |                  *                W23 22 21       /  |
        |                 W11                              /   |
        |                                 W24 30 29       /    |
        |                                  *  *  * <------     |
        |                             _   W25 31 28            |
        |_                        __| |    *  *  *            _|
          |                      |    |    *  *  *           |
          |                      |    |   W26 32 27          |
           ----------------------     ----------------------
```

Figure 2-1: Configuration Jumper Locations

2.1.3. Bootstrap

The controller board incorporates a transparent firmware bootstrap. The bootstrap is initiated whenever program execution is started at location 173000(8) homing all drives to track 0. Next, track 1, sector 1, of unit 0 is read and diskette density is determined. If the diskette is single density, sectors 1, 3, 5, and 7 are loaded into memory starting at location 0. If the diskette is double density, sectors 1 and 3 are loaded. Program execution is then transferred to location 0. Controllers are shipped with this feature disabled. To enable the bootstrap remove the jumper between W42 and W43 and insert the jumper between W41 and W42 as shown in Table 2-3.

NOTE

Only one bootstrap should be enabled in a system for proper operation. If another bootstrap exists in the system, it must be disabled before enabling the controller bootstrap.

BOOTSTRAP	JUMPERS	
	41-42	42-43
ENABLED	IN	OUT
DISABLED *	OUT	IN
*Factory Preset		

Table 2-3: Bootstrap Option

2.1.4. Write Precompensation

The MXV22M controller provides hardware write precompensation to reduce the bit shift exhibited by all drives as the recorded flux density increases. The controller recognizes the patterns which produce bit shift and precompensates the written pattern. This unique feature allows the controller to perform reliably with any Shugart compatible drive.

Controllers are shipped with this feature enabled and it is recommended that for more reliable operation the feature not be disabled. However, if so desired, the feature can be defeated by cutting the foil-etched jumper between W12 and W13 and inserting a jumper between W13 and W14 as shown in Table 2-4.

WRITE PRECOMPENSATION	JUMPERS	
	12-13	13-14
ENABLED*	IN	OUT
DISABLED	OUT	IN
*Factory Preset		

Table 2-4: Write Precompensation

2.1.5. Step Rate Control

The controller is shipped configured with a 3ms step rate. Refer to Table 2-5 for the step rate jumper options.

STEP RATE	JUMPERS
Rate	36-37
3ms *	IN
6MP	OUT
*FACTORY PRESET	

Table 2-5: Step Rate

2.1.6. 22 Bit Addressing

The controller is shipped with 22 bit addressing disabled. Enabling this option provides extended address control during DMA transactions allowing the controller to transfer information throughout 22 bit address space. The additional four bits of address (A(18)-A(21)) are communicated to the controller as described in section 3.1.2. Before enabling this option it is necessary to modify the corresponding software drivers in order to maintain proper register communication as described in section 5. To enable 22-bit addressing jumper W46 and W47 as shown in Table 2-6.

22 BIT ADDRESSING	JUMPERS
	46-47
ENABLED	IN
DISABLED *	OUT
*Factory Preset	

Table 2-6: 22 Bit Addressing

2.1.7. Miscellaneous Options

There are several options related to factory configuration of the controller. These options must be configured as shown for proper operation of the controller. Refer to Table 2-7 for these options. During DMA operations if the bus address established extends into the peripheral address page the controller asserts bank select 7 (BS7) as required by normal bus protocol. If the application requires extended memory, overlapping the peripheral address page, this option can be disabled as indicated in Table 2-7.

OPTION	JUMPERS						
	1-2	2-3	4-5	5-6	33-34	34-35	10-11
FACTORY*	OUT	IN	OUT	IN	OUT	IN	-
BS7 ENABLED	-	-	-	-	-	-	IN
BS7 DISABLED	-	-	-	-	-	-	OUT
* Factory Preset							

Table 2-7: Miscellaneous Options

2.2. Drive Configuration

For proper operation, the mini-floppy drives must be configured with attention to several options. The controller uses radial drive selection. Thus the drive(s) should be correspondingly configured. When two drives are used, the first should be 0 and the second drive 1. For details concerning these and other option refer to Table 2-8.

OPTION	DESCRIPTION	DUAL DRIVE 0	DUAL DRIVE 1	SINGLE DRIVE 0
DS1	Drive select 0	IN	OUT	IN
DS2	Drive select 1	OUT	IN	OUT
DS3	Drive select 2	OUT	OUT	OUT
DS4	Drive select 3	OUT	OUT	OUT
HS		IN	IN	IN
HM		OUT	OUT	OUT
MUX		OUT	OUT	OUT

Table 2-8: Tandon TM 100-4 Drive Configuration

2.3. Cabling

A 34-conductor ribbon cable connects the controller to any Shugart compatible drive(s). If the optional cable is purchased with the controller, connect the socket connector to the 50-pin header located at the edge of the controller board. Observe the alignment of pin 1 of the socket connector and header as indicated by the arrows shown in Figure 2-2. The two 34-pin connectors should be connected to the corresponding drives, again observing the location of pin 1.

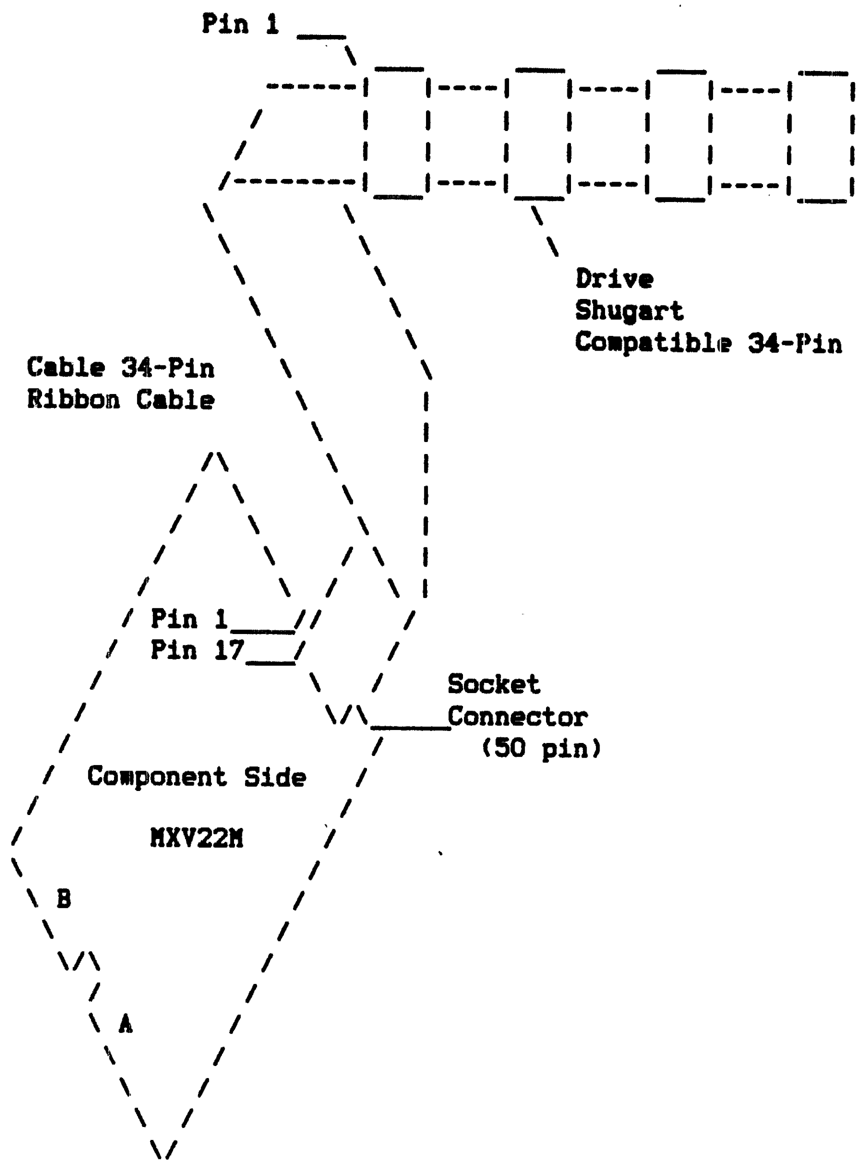

Figure 2-2: Drive/Controller Cabling

The connector pins illustrated in Figure 2-3 are compatible with both the Shugart SA 460 and Tandon 100-4. Any drive that has both a Shugart compatible interface and connector should function properly with the controller.

```
          -------
 |---| 1 | 2 |
 |---| 3 | 4 |
 |---| 5 | 6 |
 |---| 7 | 8 |
 |---| 9 |10 |
 |---|11 |12 |
 |---|13 |14 |
 |---|15 |16 |
 |---|17 |18 |                                                    |2 |
 |---|19 |20 |                                                    |4 |
 |---|21 |22 |-----      DRIVE SELECT 4      ---------------->|6 |
 |---|23 |24 |<----      INDEX               -----------------|8 |
 |---|25 |26 |-----      DRIVE SELECT 1      ---------------->|10|
 |---|27 |28 |-----      DRIVE SELECT 2      ---------------->|12|
 |---|29 |30 |-----      DRIVE SELECT 3      ---------------->|14|
 |---|31 |32 |-----      MOTOR ON            ---------------->|16|
 |---|33 |34 |-----      DIRECTION           ---------------->|18|
 |---|35 |36 |-----      STEP                ---------------->|20|
 |---|37 |38 |-----      WRITE DATA          ---------------->|22|
 |---|39 |40 |-----      WRITE GATE          ---------------->|24|
 |---|41 |42 |<----      TRACK 00            -----------------|26|
 |---|43 |44 |<----      WRITE PROTECT       -----------------|28|
 |---|45 |46 |<----      READ DATA           -----------------|30|
 |---|47 |48 |-----      SIDE SELECT         ---------------->|32|
 |---|49 |50 |                                                    |34|
  |       -------                                                 Drive
 ---                                                            Connector
  --
Controller
  Connector
```

Figure 2-3: Connector Pin Definitions

2.4. CONTROLLER INSTALLATION

The controller can be inserted and will function in any LSI-11 bus slot provided that both interrupt and DMA continuity are maintained. Since these signals are daisy chained through the bus slots, no unused slots between the LSI-11 processor and the floppy controller may exist. Determine the order that the priority chain flows by consulting the documentation suppied with the LSI-11 system. Note that when two interrupts of the same priority level are asserted, the closer a device is located to the processor, the higher its relative priority.

2.5. INITIAL OPERATION AND CHECKOUT

Before the following procedures are done, verify that the controller has been configured as described in Sections 2.1-2.2.

NOTE

The bootstrap must be disabled for the following procedures.

1. Apply AC and DC power to the drive(s). The in-use indicators on both drives should be off.

2. Place the Run/Halt switch on the processor to the Halt position and turn on the processor. An "@" character should be printed on the terminal signifying that console ODT has been entered. Both drives (first drive 1, then drive 0) will step the heads inward 2-tracks, then step the heads outward until the home signal is detected. The heads will not load, and, if the drive is configured as per Section 2-2, the in-use indicators will not light. If the above events do not occur, check the cabling and drive power supplies.

3. Place a preformatted scratch diskette in drive 0.

4. If the standard address assignment is selected, open the CS register using ODT by typing 177170/ on the terminal. The processor will display the contents of the CS register. If the controller is operating properly a 004040(8) should be printed. Deposit a 40000(8) in the CS register by typing 40000 CR. This command will initialize the controller. Both drives should calibrate for home position. First, drive 1 steps inward 2 tracks then outward one track at a time until the drive indicates track 0 has been reached. The procedure is repeated on drive 0. After both drives are calibrated, the head on drive 0 is loaded. Sector 1 of track 1 on drive 0 is read into the controller buffer. This operation is indicated by the in-use LED on drive 0. The LED will remain on for a short time after the read operation is complete.

If, after initializing, the drives do not calibrate or the LED is not activated, check the cabling and power supplies.

5. Reopen the CS register (location 177170(8)) using ODT as described above. The contents of this location should be 004040(8). Examining the next location 177172(8) by using the linefeed key or typing in 177172/ should yield either a 204(8) or 244(8). For a detailed description of the register protocol and bit definition, refer to Section 3.

6. If the above procedures function as described, the controller is ready for use. Either diagnostics or an operating system can be booted. For details on bootstrapping refer to Section 4.2.

7. If the above procedures cannot be validated, consult the factory or your local representative for assistance.

NOTE

For an LSI 11/23 the CS and DB registers are addressed as 777170(8) and 777172(8), respectively.

Section 3
Functional Description

3. GENERAL

This section describes device registers and command protocol for the MXV22M.

All software control of the MXV22M is performed by means of two device registers: the command and status (MXVCS) register and a multipurpose data buffer (MXVDB) register. These registers are assigned the bus address 177170(8) and 177172(8), respectively. The registers can be read or loaded, with a few exceptions, using any instruction referring to their addresses.

The MXVCS register passes control information from the CPU to the controller and reports status and error information from the controller to the CPU. The MXVDB is provided for additional control and status information between the CPU and the controller. The information that is present in the MXVDB at any given time is a function of the controller operation in progress.

The controller contains a sector buffer capable of storing a complete sector. For read/write operations the buffer is either "filled" before a write command or "emptied" after a read command under DMA control. During a write command the controller locates the desired sector and the buffer information is transferred to the diskette. During a read operation the desired sector is located and the sector data are transferred to the buffer.

3.1. REGISTER DEFINITIONS

3.1.1. MXVCS - Command and Status Register (177170(8))

The format of the MXVCS register is shown below. Functions are initiated by loading the command and status (CS) register, when not busy (bit 5 = 1), with bit 0 = 1. Command protocol is discussed in detail in section 3.2.

```
 15  14  13 12  11  10   9   8   7   6   5   4   3   2   1   0
 ---------------------------------------------------------------
|ERR|INT|EXT ADD|RX |22 |   |DEN|TR |INT|DN |UNT| FUNCTION |GO |
|   |   |       |02 |EBL|   |   |   |ENB|   |SEL|          |   |
 ---------------------------------------------------------------
```

BIT DESCRIPTION

15 ERROR: This bit is set by the controller to indicate that an error has occurred during an attempt to execute a command. This bit is cleared by the initiation of a new command or by setting the initialize bit. When an error is detected the MXVES is read into the MXVDB. This bit is a read-only bit.

14 MXV22M INITIALIZE: This bit is set by the program to initialize the controller without initializing all the devices on the LSI-11 bus.

CAUTION

Loading the lower byte of the MXVCS will also load the upper byte of the MXVCS.

When this bit is set, the controller will negate Done and move the head position mechanism of drive 1 (if two drives are available) to track 0. When completed, the controller will repeat the operation on drive 0.

The controller will then clear the error and status register, set Initialize Done, and set Drive Ready if drive 0 is ready. Finally, the controller will read sector 1, track 1, of drive 0.

When in 22-bit mode, this bit is set by the controller to indicate that the controller is ready to accept the extended address bits for a full/empty buffer or read error code command.

13-12 EXTENDED ADDRESS BITS: These bits are used to specify an extended bus address. Bit 12 = MA16. Bit 13 = MA17. These are write-only bits.

11 RX02: This bit is asserted by the controller to indicate that this is an RX02 type system. This is a read-only bit.

10 22 BIT ADDRESSING ENABLED: This bit is normally read as a zero. If 22 bit addressing is enabled and fill/empty buffer or read error code commands are initiated this bit will be set along with transfer ready (TR).

09 RESERVED: Must be written as zero.

08 DENSITY SELECT: This bit selects either single or double density operation. When cleared, single density is selected; when set, double density is selected. This is a read/write bit.

07	TRANSFER REQUEST: This bit signifies that the controller needs data or has data available. This is a read-only bit.
06	INTERRUPT ENABLE: This bit is set by the program to enable an interrupt when the controller has completed an operation and asserted the Done bit. The condition of this bit is cleared by initialize. This is a read/write bit.
05	DONE: This bit indicates the completion of a function. Done will generate an interrupt when asserted if interrupt enable (MXVCS bit 6) is set. This is a read-only bit.
04	UNIT SELECT: This bit selects one of the two possible disks for execution of the desired function. This is a read/write bit.
03-01	FUNCTION SELECT: These bits code one of the eight possible functions described in detail within this section. These are write-only bits.

000	Fill Buffer
001	Empty Buffer
010	Write Sector
011	Read Sector
100	Set Media Density/Format
101	Read Status
110	Write Deleted Data Sector
111	Read Error Code

00	GO: Initiates a command (write-only bit).

3.1.2. MXVDB - Data Buffer (177172(8))

This register serves as a general purpose data path between the controller and the LSI-11. It will represent one of seven registers according to the protocol of the function in process. These registers include the MXVDB, MXVTA, MXVSA, MXVWC, MXVBA, MXVBAE and MXVES.

This register is a read/write register if the controller is not in the process of executing a command (i.e., it may be manipulated without affecting the controller). When the controller is executing a command, the register can only be accessed when MXVCS bit 7 (TR) is set.

__Data Buffer Register (MXVDB)__

All information transferred to and from the floppy media passes through the MXVDB register and is addressable only under the protocol of the function in progress.

```
 15  14  13  12  11  10   9   8   7   6   5   4   3   2   1   0
 ----------------------------------------------------------------
|   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |
|   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |
 ----------------------------------------------------------------
\---------------------------\/----------------------------------/
                        READ/WRITE DATA

                             MXVDB FORMAT
```

__Track Address Register (MXVTA)__

This register is loaded to indicate on which of the 115(8) (77 decimal) tracks a given function is to operate. It can be addressed only under the protocol of the function in progress. Bits 8 through 15 are not used and are ignored.

```
 15  14  13  12  11  10   9   8   7   6   5   4   3   2   1   0
 ----------------------------------------------------------------
|///|///|///|///|///|///|///|///| 0 |   |   |   |   |   |   |   |
|///|///|///|///|///|///|///|///|   |   |   |   |   |   |   |   |
 ----------------------------------------------------------------
\---------------\/---------------/\--------------\/--------------/
             NOT USED                          0-114(8)
                             MXVTA FORMAT
```

__Sector Address Register (MXVSA)__

This register is loaded to indicate on which of the 32(8) (26 decimal) sectors a given function is to operate. It can be addressed only under the protocol of the function in progress. Bits 8 through 15 are not used and are ignored.

```
 15  14  13  12  11  10   9   8   7   6   5   4   3   2   1   0
 ----------------------------------------------------------------
|///|///|///|///|///|///|///|///| 0 | 0 | 0 |   |   |   |   |   |
|///|///|///|///|///|///|///|///|   |   |   |   |   |   |   |   |
 ----------------------------------------------------------------
\---------------\/---------------/\--------------\/--------------/
             NOT USED                          1-32(8)
                             MXVSA FORMAT
```

Word Count Register (MXVWC)

This register is loaded with the number of words (maximum of 128 decimal) to be transferred. At the end of each transfer the word count register is decremented. When the contents of the register are decremented to zero transfers are terminated; Done is set (MXVCS bit 5); and, if enabled, an interrupt is requested. If the word count is greater than the limit for the density specified, the controller asserts a Word Count Overflow (bit 10 of the MXVES). This register can be addressed only under the protocol of the function in progress. Bits 8 through 15 are not used and are ignored.

```
 15  14  13  12  11  10   9   8   7   6   5   4   3   2   1   0
 ---------------------------------------------------------------
|///|///|///|///|///|///|///|///|   |   |   |   |   |   |   |   |
|///|///|///|///|///|///|///|///|   |   |   |   |   |   |   |   |
 ---------------------------------------------------------------
\--------------\/---------------/\------------\/----------------/
           NOT USED                          0-200(8)
                          MXVWC FORMAT
```

Bus Address Register (MXVBA)

This register is used to generate the bus address which specifies the location to and from which data are to be transferred. The register is incremented after each transfer. It will increment across 32K boundary lines via the extended address bits in the control and status register and the bus address extension register. Systems with only 16 address bits will "wrap around" to location zero when the extended address bits are incremented. This register can be addressed only under the protocol of the function in progress. Bit 0 is not used and is ignored.

```
 15  14  13  12  11  10   9   8   7   6   5   4   3   2   1   0
 ---------------------------------------------------------------
|   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |////|
|   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |////|
 ---------------------------------------------------------------

                          MXVBA FORMAT
```

Bus Address Extension Register (MXVBAE)

This register contains the extended address bits (A18-A21) when 22 bit addressing is enabled. Bits 4 thru 15 are not used and are ignored.

```
 15  14  13  12  11  10   9   8   7   6   5   4   3   2   1   0
 ---------------------------------------------------------------
|////|////|////|////|////|////|////|////|////|////|////|////|A21|A20|A19|A18|
|////|////|////|////|////|////|////|////|////|////|////|////|   |   |   |   |
 ---------------------------------------------------------------

                          MXVBAE FORMAT
```

Error and Status Register (MXVES)

This register contains the current error and status conditions of the drive selected by bit 4 (Unit Select) of the MXVCS. This read-only register can be addressed only under the protocol of the function in progress. The MXVES is loaded in the MXVDB upon completion of a function.

```
  15  14  13  12  11  10   9   8   7   6   5   4   3   2   1   0
 ----------------------------------------------------------------
|////|////|////|////|NXM|WC |   |UNT|DRV|DD |DRV|DEN|AC |ID |   |CRC|
|////|////|////|////|   |OVF|   |SEL|RDY|   |DEN|ERR|LO |   |   |   |
 ----------------------------------------------------------------
\-------\/-------/
     NOT USED
```

MXVES FORMAT

BIT DESCRIPTION

15-12 Not Used.

11 NONEXISTENT MEMORY ERROR: This bit is asserted by the controller when the memory address specified for a DMA operation is nonexistent.

10 WORD COUNT OVERFLOW: This bit indicates that the word count specified is greater than the limit for the density selected. Upon detecting this error the controller terminates the fill or empty buffer operation and asserts the Error and Done bits.

09 NOT USED

08 UNIT SELECT: This bit indicates the drive currently selected. If cleared, it indicates drive 0; if set, it indicates drive 1.

07 DRIVE READY: This bit is asserted if the unit currently selected exists, is properly supplied with power, has a diskette installed correctly, has its door closed, and has a diskette up to speed. This bit is only valid when retrieved via a read status function or at the completion of initialize when it indicates the status of drive 0.

06 DELETED DATA: During data recovery, the identification mark preceding the data field was decoded as a deleted data mark.

05 DRIVE DENSITY: The bit indicates the density of the diskette in the selected drive. When zero, it indicates single density; when set to one, it indicates double density.

04 DENSITY ERROR: A density error was detected as the information was retrieved from the data field of the diskette (a density error occurs when the density selected differs from that of the data field). Upon detecting this error the controller loads the MXVES into the MXVDB and asserts the Error and Done bits.

03 ACLO: Set by the controller to indicate a power failure.

02	INITIALIZE DONE: This bit is asserted to indicate completion of the initialize routine, which can be caused by system power failure or programmable LSI-11 bus initialize.
01	NOT USED.
00	CRC ERROR: A cyclic redundancy check error was detected as information was retrieved from a data field of the diskette. The information stored in the buffer should be considered invalid. Upon detection of this error the controller loads the MXVES into the MXVDB and asserts the Error and Done bits.

3.1.3. Extended Status Registers

The controller has four internal status registers. These registers provide specific error information in the form of error codes as well as drive status information depending upon the general error type. The registers can be retrieved by a read error code function as described in Section 3.2.8.

Word 1 <7:0> - Definitive Error Code

Octal Code	Error Code Meaning
040	Tried to access a track greater than 76.
050	Home was found before desired track was reached.
070	Desired sector could not be found after looking at 52 headers (2 revolutions).
120	A preamble could not be found.
150	The header track address of a good header does not compare with the desired track.
160	Too many tries for an IDAM (identifies header).
170	Data AM not found in allotted time.
200	CRC error on reading the sector from the disk.
240	Density Error
250	Wrong Key word for Set Media Density Command
260	Illegal Data AM
270	Invalid POK during write sequence
300	Drive not ready.
310	Drive write protected.

Word 1 <15:8> - Not Used

This register is always cleared by the controller.

Word 2 <7:0> - Current Track Address of Drive 0

This register is cleared during the initialize command in order to synchronize with actual track position. The register is updated with each seek on drive 0 and maintains current track position.

Word 2 <15:8> - Current Track Address of Drive 1

This register is cleared during the initialize command in order to synchronize with actual track position. The register is updated with each seek on drive 1 and maintains current track position.

Word 3 <7:0> - Target Track of Current Disk Access

If legal, the track specified for the last read/write command is saved in this register.

Word 3 <15:8> - Target Sector of Current Disk Access

The sector specified for the last read/write command is saved in this register.

Word 4 <15:8> - Track Address of Selected Drive

This register contains the track address read from the sector header of the desired sector during the last read/write command.

3.2. COMMAND PROTOCOL

Data storage and recovery using the MXV22M controller is accomplished by careful manipulation of the MXVCS and MXVDB registers according to the strict protocol of the individual functions. The penalty for violation of protocol can be permanent loss of data. Each of the functions are encoded and written into the command and status register bits 1-3 as described in Section 3.1.1. The detailed protocol for each function is described below.

3.2.1. Fill Buffer (000)

This function is used to fill the controller buffer with data from the host processor. The number of words to transfer is specified by the host. The command density bit determines the buffer size (64 or 128 words). The controller zero-fills the remaining buffer space. If the word count is too large for the density selected, the function is aborted, Error and Done are asserted and the Word Count Overflow bit is set in the MXVES.

The contents of the buffer may be written on the diskette with a subsequent write sector command or returned to the host processor using an empty buffer command.

When the command is loaded, MXVCS bit 5 (Done) is negated. MXVCS bit 8 (density) must be set to define the buffer size. MXVCS bits 12 and 13 (extended address bits A16 and A17) must also be asserted to define the extended memory segment used with the buffer address, yet to be specified, to form the absolute memory address of the data to be transferred. MXVCS bit 4 (unit select) and bit 9 (head select) are ignored since no drive operation is required. When MXVCS bit 7 (TR) is first asserted, the program must move the word count into the MXVDB which will negate TR.

When the controller again asserts TR, the program must move the buffer address into the MXVDB. If 22 bit addressing is enabled, MXVCS bit 10 (22 EBL) is set, the controller asserts TR or INIT (either bit may be set) and the program can move the extended address bits (A18-A21) into the MXVDB. If nothing is moved into the MXVDB (BAE REG) within a timeout period the controller assumes zeroes defaulting to 18 bit addressing mode. The controller then negates TR, initiates a DMA cycle, and transfers the first word from the host processor to the controller buffer. At the end of the transfer the word count register is decremented and the buffer address is incremented by two. This cycle is repeated until the word count register becomes zero. The controller zero-fills the remaining buffer space, sets the Done bit, and if enabled, causes an interrupt request. After Done is asserted the MXVES is moved into the MXVDB.

During the Data Transaction, if any non-existent memory is addressed, the controller will time out and abort the function. The Error and Done bits will be asserted. MXVES bit 11 (NXM) will be set and the MXVES will be moved into the MXVDB; if enabled, an interrupt request will be generated.

3.2.2. Empty Buffer (001)

This function is used to transfer the contents of the controller to the host processor. The number of words to transfer is specified by the host. The command density bit determines the maximum legal word count. If the word count specified is too large for the density selected the function is aborted, Error and Done are asserted and the Word Count Overflow bit is set in the MXVES.

The contents of the buffer may be transfered to the host as many times as desired or may be written on the diskette with a subsequent write sector command. Unless a fill buffer or read sector command is issued, the controller buffer is not destroyed.

When the command is loaded, MXVCS bit 5 (Done) is negated, MXVCS bit 8 (density) must be set to allow the proper word count limit. MXVCS bits 12 and 13 (extended address bits A16 and A17) must also be asserted to define the extended memory segment used with the buffer address, yet to be specified, to form the absolute memory destination address. MXVCS bit 4 (unit select) and bit 9 (head select) are ignored since no drive operation is required. When MXVCS bit 7 (TR) is first asserted the program must move the word count into the MXVDB which will negate TR. When the controller again asserts TR the program must move the buffer address into the MXVDB. If 22 bit addressing is enabled, MXVCS bit 10

(22 EBL) is set, the controller asserts TR or INIT (either bit may be set) and the program can move the extended address bits (A18-A21) into the MXVDB. If nothing is moved into the MXVDB (BAE REG) within a timeout period the controller assumes zeroes defaulting to 18 bit addressing mode. The controller then negates TR, initiates a DMA, and transfers the first word of the buffer to the host processor. At the end of the transfer, the word count register is decremented and the buffer address register is incremented by two. This cycle is repeated until the word count register becomes zero. The controller then sets the Done bit and if enabled causes an interrupt request. After Done is asserted the MXVES is moved into the MXVDB.

During the DMA transaction, if any non-existent memory is addressed, the controller will time out and abort the function. The Error and Done bits will be asserted. MXVES bit 11 (NXM) will be set and the MXVES will be moved into the MXVDB. If enabled, an interrupt request will be generated.

3.2.3. Write Sector (010)

This function is used to locate a desired track and sector and write the sector with the contents of the internal sector buffer. When the MXVCS is loaded with this command, the MXVES is cleared and both the TR and Done bits are negated. When TR is first asserted the program must load the desired sector address into the MXVDB which will negate TR. When TR is again asserted the program must load the desired track address into the MXVDB which will negate TR. The controller then seeks the desired track and attempts to locate the desired sector. The desired track is compared with the track field of the sector header. If they do not match the operation is aborted, the Error and Done bits are asserted, the MXVES is moved into the MXVDB, and if enabled the controller will assert an interrupt request.

A data address mark is read to determine the diskette density. If the densities of the function and the diskette do not agree, the controller will abort the operation, assert the Error and Done bits and set MXVES bit 4 (Density Error) and load the MXVES into the MXVDB. If enabled, an interrupt request will be generated.

If the densities agree but the controller is unable to locate the desired sector within two diskette revolutions, the controller will abort the operation, move the contents of MXVES into MXVDB, assert the Error and Done bits, and if enabled, assert an interrupt request.

If the desired track and sector are located and the densities agree, the controller will write the contents of the internal sector buffer followed by a CRC character, all in the function selected density. The controller completes the operation by moving the MXVES to the MXVDB, asserts Done, and if enabled, asserts an interrupt request.

CAUTION

The contents of the internal sector buffer are lost during a power failure. However, after power is brought back to normal, a write sector command will cause the random contents of the buffer to be written on the diskette with a valid CRC character.

NOTE

The contents of the sector buffer are not destroyed by a write sector operation.

3.2.4. Read Sector (011)

This function is used to locate the desired track and sector and transfer the contents of the data field into the controller's internal sector buffer. When the MXVCS is loaded with this command, the MXVES is cleared and both the TR and Done bits are negated. When TR is first asserted the program must load the desired sector address into the MXVDB which will negate TR. When TR is again asserted the program must load the desired track address into the MXVDB which will negate TR.

Both the TR and Done bits remain negated while the controller attempts to locate the desired sector. If after two revolutions the controller is unable to locate the desired sector, the operation is aborted. The controller will move the MXVES into the MXVDB, assert the Error and Done bits, and if enabled, assert an interrupt request.

When the desired sector is located, the controller will then compare the desired track with the track field of the sector header. If they do not match, the operation is aborted. The Error and Done bits are asserted, the MXVES is moved into the MXVDB, and if enabled, the controller asserts an interrupt request.

If a legal data address mark is located and the densities of the diskette and function agree, the controller will read the data from the sector into the internal buffer. If the data address mark indicated a deleted data field, MXVES bit 6 (DD) is set. As data are stored in the internal buffer, a CRC is computed on the data and the CRC bytes recorded. A non-zero result indicates a read error. When a CRC error is encountered, the controller sets MXVES bit 0 (CRC), moves the MXVES into the MXVDB, asserts the Error and Done bits, and if enabled, asserts an interrupt request.

If the desired sector is located, the density of the diskette and function agree, and the data are transferred with no CRC error, the controller will assert Done, and if enabled, will assert an interrupt request.

3.2.5. Set Media Density (100)

This function is dual purpose. The controller can set the media density by revriting all the data address marks (single or double density) and writing zero data fields in the selected density. The controller can also "reformat" the entire diskette by revriting both the sector headers and the data fields. The data fields are written in the selected density preceded by the corresponding data address mark. Both commands are initiated by the set media function but differ in the keyword required by the controller to execute the command.
When the MXVCS is loaded with the command, the MXVES is cleared and the Done bit is negated. When TR is set, the program must respond with a keyword. This keyword must be deposited in the MXVDB to complete the protocol. When the controller recognizes this character, it begins executing the command. If an illegal keyword is used, the operation is aborted. The MXVES is moved into the MXVDB, the Error and Done bits are set, and if enabled, the controller asserts an interrupt request.
If the keyword used is a 111(8), the controller initiates a set media density operation. This operation starts at track 0, sector 1. Each sector header is located and a write operation is initiated. A data field is written with zero data in the density selected. If an error occurs reading any header, the operation is aborted. The MXVES is moved into the MXVDB, the Error and Done bits are set, and if enabled, the controller asserts and interrupt request. If the operation is successfully completed, Done is set and if enabled, the controller asserts an interrupt request.

If the keyword used is a 222(8), the controller initiates a format operation. This function starts at the physical index of track 0. Each track is written first with an index address mark, then 26 sector headers are written sequentially about the cylinder. When each track has been written, the controller initiates a set media density function as described above.

The following input string will format the selected unit, in the desired density.

777170/	4040	XXXX <LF>
177172/	000000	222 <CR>

CAUTION

The set media density function takes about 35 seconds and the format function takes about 95 seconds. Neither should be interrupted. If either operation is interrupted, an illegal diskette has been generated, and the operation should be repeated. If an error occurs during a set media density function or a format function, an illegal diskette has been generated. The operation should be repeated.

3.2.6. Read Status (101)

This function is used to update the drive status information and is initiated by loading the command into the MXVCS. The Done bit is negated. MXVES bit 7 (Drive Ready) is updated by sampling the drive ready status line. Drive density is updated by loading the head of

the selected drive and reading the first data address mark. The controller then moves the MXVES into the MXVDB, asserts Done, and if enabled, asserts an interrupt request. This operation requires about 250ms to complete.

3.2.7. Write Deleted Data Sector (110)

This operation is identical to Write Sector (010) with one exception. The data address mark preceding the data is not the standard data address mark. A single or double density deleted data address mark is written according to the density of the function.

3.2.8. Read Error Code (111)

This function is used to retrieve the extended status registers and is initiated by loading the MXVCS with the command. The Done bit is negated. When TR is asserted, the program must load the Bus Address into the MXVDB. If 22 bit addressing is enabled, MXVCS bit 10 (22 EBL) is set, the controller asserts TR or INIT (either bit may be set) and the program can move the extended address bits (A18-A21) into the MXVDB. If nothing is moved into the MXVDB (BAE REG) within a timeout period the controller assumes zeroes defaulting to 18 bit addressing mode. The controller then negates TR and assembles one word at a time and, under DMA control, transfers them to memory starting at the address specified.

If non-existent memory is encountered during the transfer, the operation is aborted. The Error and Done bits will be asserted, MXVES bit 11 (NXM) will be set, and the MXVES will be moved into the MXVDB. If enabled, an interrupt request will be generated.

When all four words have been transferred the Done bit is set and if enabled, an interrupt request is generated.

Section 4

Controller Operations

4. GENERAL

This section provides the user pertinent information concerning the description and use of the controller functions. The functions covered include: bootstrapping, formatting, fill/write operations, read/empty operations, write current control, write precompensation, and power fail protection.

4.1. BOOTSTRAPPING THE CONTROLLER

If the bootstrap is enabled as described in Section 2.1.3, the controller will respond to the standard bootstrap address 173000(8). The controller is bootstrapped by typing 773000G while in console ODT. This causes a bus INIT and transfers program execution to location 173000(8). An alternate method is to strap the LSI-11 processor to power up Mode 2. In this mode, when a power up occurs, the processor automatically starts execution at 173000(8). Power-up strapping procedures for the LSI-11 processor can be found in the Microcomputer Processors Handbook.*

To boot either a single or double density diskette use the following procedure:

1. Place the diskette in drive 0.

2. If the processor is strapped for power-up Mode 2, operate the INIT (boot) switch or cycle DC power OFF and ON.
3. If the processor is not strapped for power up Mode 2 while in console ODT, type 773000G.

*Published by Digital Equipment Corporation. Maynard, Mass., 1979.

4.1.1. Bootstrap Operation

The bootstrap is not a standard ROM program. It uses the controller's microprocessor to capture the bus; to read block 0 of the diskette into memory starting at location 0; and finally to transfer program execution to memory location 0.

Any attempt to read location 173000(8) will result in a non-existent memory trap. The controller only responds to this address immediately after a bus INIT. For this reason the bootstrap is called "transparent". When the processor attempts to fetch location 173000(8) following a bus INIT, the controller responds by passing the processor a "CLEAR R0" instruction. The processor clears R0 and then attempts to fetch location 173002(8). The controller passes the processor a "LOAD IMMEDIATE" instruction with R1 as the destination. The processor then attempts to fetch the source operand from location 173004(8). The controller passes the the device address 177170(8) if the standard address is selected. The processor moves the address into R1 and then attempts to fetch location 173006(8). The controller first asserts a Direct Memory Access Request (DMR) then passes the processor a "CLEAR PC" instruction. Before the processor executes the instruction it passes bus mastership to the controller. The controller moves a "BRANCH TO CURRENT LOCATION" instruction (777(8)) into memory location 0 under DMA control. When the controller releases bus mastership the processor executes the "CLEAR PC" instruction and, in so doing, transfers program execution to location 0. The processor is thus forced to loop at location 0. The controller initiates a Read Status function on drive 0 to determine diskette density. If the diskette is single density the controller reads sectors 1,3,5, and 7 of track 1 of drive 0 into locations 2 through 176, 200 through 376, 400 through 576, and 600 through 776 respectively. If the diskette is double density the controller reads sectors 1 and 3 of track 1 of drive 0 into locations 2 through 376, and 400 through 776 respectively. Finally, the controller DMA's location 0 with a NOP instruction (240(8)) allowing the processor to execute the system bootstrap. If there is no diskette in drive 0 nothing will be transferred to memory and the processor will continue to loop at location 0 until halted.

The format command selects diskette density and unit. Table 4-1 lists the various command word formats.

	Unit 0	Unit 1
Single Density	11(8)	31(8)
Double Density	411(8)	431(8)

Table 4-1: Command Word Formats

Figure 4-1 illustrates a format subroutine. The format command is loaded into MXVCS. When TR is set, the keyword 222(8) is loaded into MXVDB. When the diskette has been formatted a return is made.

```
FORMAT:
        MOV  #11, CMD          ;FORMAT
        BIS  DENS, CMD         ;DENSITY
        BIS  UNIT, CMD         ;UNIT
        MOV  CMD, @#MXVCS      ;SELECT FUNCTION
        JSR  PC, TRWAIT        ;WAIT FOR TR
        MOV  #222, @#MXVDB     ;KEYWORD
        JSR  PC, DNWAIT        ;WAIT FOR DONE
        TST  @#MXVCS           ;ERROR
        BMI  FRMERR            ;BR IF SO
        RTS  PC
FRMERR:
```

Figure 4-1: Format Subroutine

Alternatively a diskette can be formatted using console ODT. Open the CS register and deposit the appropriate command. Then deposit the format key word 222(8) in the DB register. The following is an example of formatting unit 0 in double density.

```
177170/      004040        411 <LF>
177172/      000000        222 <CR>
```

4.2. FILL/WRITE OPERATIONS

Figure 4-2 illustrates subroutines to write data on a diskette which is done by performing a fill buffer operation followed by a write sector.

The Fill Buffer command, specifying single or double density is loaded into the MXVCS. When TR is set, the word count is loaded into the MXVDB. When TR is again set, the bus address of the data is loaded into the MXVDB. If 22 bit addressing is enabled TR or INIT is set and the extended address bits are moved into the MXVDB. A return is made when the controller's sector buffer is filled. The Write Sector command (specifying density and unit) is loaded into the MXVCS. When TR is set the sector address is loaded into the MXVDB. When TR is again set, the track address is loaded into the MXVDB. When the contents of the controller's sector buffer are written at the selected sector, a return is made.

```
FILLBF:
        MOV     #1, CMD                 ;FILL BUFFER
        BIS     DENS, CMD               ;DENSITY
        MOV     CMD, @#MXVCS            ;SELECT FUNCTION
        JSR     PC, TRWAIT              ;WAIT FOR TR
        MOV     COUNT, @#MXVDB          ;WORD COUNT
        JSR     PC,TRWAIT
        MOV     #BUFOUT, @#MXVDB        ;BUS ADDRESS OF DATA
        JSR     PC, TRINIT              ;WAIT FOR TR or INIT*
        MOV     #EXTAD,@#MXVDB          ;EXTENDED ADDRESS BITS*
        JSR     PC, DNWAIT              ;WAIT FOR DONE
        TST     @#MXVCS                 ;ERROR
        BMI     ERFIL                   ;BR IF SO
        RTS     PC
ERFIL:

WSECT:
        MOV     #5, CMD                 ;WRITE, SECTOR
        BIS     DENS, CMD               ;DENSITY
        BIS     UNIT, CMD               ;UNIT
        MOV     CMD, @#MXVCS            ;SELECT FUNCTION
        JSR     PC, TRWAIT              ;WAIT FOR TR
        MOV     SECTOR, @#MXVDB         ;SECTOR
        JSR     PC, TRWAIT
        MOV     TRACK @#MXVDB           ;TRACK
        JSR     PC, DNWAIT              ;WAIT FOR DONE
        TST     @#MXVCS                 ;ERROR
        BMI     WSERR                   ;BR IF SO
        RTS     PC

WSERR:
```

*Only required if 22 bit addressing is enabled!

Figure 4-2: Write Data Subroutines

4.3. READ/EMPTY OPERATIONS

Figure 4-3 illustrates subroutines to read data from a diskette which is done by performing a Read Sector operation followed by an Empty Buffer operation.

The Read Sector command (specifying density and unit) is loaded into the MXVCS. When TR is set the sector address is loaded into the MXVDB. When TR is again set, the track address is loaded into the MXVDB. When the contents of the selected sector are read into the controller's sector buffer, a return is made.

The Empty Buffer command, specifying density, is loaded into the MXVCS. When TR is set, the word count is loaded into the MXVDB. When TR is again set, the bus address of storage buffer is loaded into the MXVDB. If 22 bit addressing is enabled TR or INIT is asserted and the extended address bits are loaded into the MXVDB. A return is made after the contents of the controller's buffer are transferred to the memory storage buffer.

```
RSECT:  MOV     #7, CMD             ;READ SECTOR
        BIS     DENS, CMD           ;DENSITY
        BIS     UNIT, CMD           ;UNIT
        MOV     CMD, @#MDVCS        ;SELECT FUNCTION
        JSR     PC, TRWAIT          ;WAIT FOR TR
        MOV     SECTOR, @#MDVDB     ;SECTOR
        JSR     PC, TRWAIT
        MOV     TRACK, @#MXVDB      ;TRACK
        JSR     PC, DNWAIT          ;WAIT FOR DONE
        TST     @#MXVCS             ;ERROR
        BMI     RSERR               ;BR IF SO
        RTS     PC
RSERR:

EMPBF:  MOV     #3, CMD             ;EMPTY BUFFER
        BIS     DENS, CMD           ;DENSITY
        MOV     CMD, @#MXVCS        ;SELECT FUNCTION
        JSR     PC, TRWAIT          ;WAIT FOR TR
        MOV     COUNT, @#MXVDB      ;WORD COUNT
        JSR     PC, TRWAIT
        MOV     #BUFFIN, @#MXVDB    ;BUS ADDRESS FOR DATA
        JSR     PC, TRINIT          ;WAIT FOR TR or INIT*
        MOV     #EXTAD, @#MXVDB     ;EXTENDED ADDRESS BITS*
        JSR     PC, DNWAIT          ;WAIT FOR DONE
        TST     @#MXVCS             ;ERROR
        BMI     EREMP               ;BR IF SO
        RTS     PC
EREMP:
```

*Only required if 22 bit addressing is enabled!

Figure 4-3: Read Data Subroutines

4.4. WRITE CURRENT CONTROL

The controller provides a write curent control signal (TG43) which is asserted whenever a track address greater than 43 is accessed. However, write current control is not used by most 5 1/4 inch floppy drives. This signal is provided on pin 2 of the 50 pin ribbon connector for the user option.

4.5. POWER FAIL PROTECTION

The controller continuously monitors both the BPOK and BDCOK bus signals. Refer to the Microcomputer Processors Handbook for detailed descriptions of these signals. When asserted, BPOK signals an impending DC power failure and guarantees 4ms of operation before BDCOK is asserted and DC power fails. Assertion of BDCOK indicates invalid DC power. This signal is hardwired in the controller as an interlock on the Write Gate signal. When BDCOK is asserted the Write Gate signal is blocked and write operations are prevented.

Before initiating a write sequence, the controller interrogates the BPOK line. If an impending DC failure is indicated the operation is aborted.

Section 5
Software Considerations

5. GENERAL

The MXV22M controller configured for 18-bit addressing can be used with all software designed to communicate with DEC's RXV21 controller. This is of particular importance when using software not supported by the driver changes presented in this section. This section describes the changes required to RT-11 and RSX-11M in order to take advantage of the 22-bit DMA support provided by the MXV22M controller. Once the changes described in this section have been incorporated into the applicable drivers the system software can be used with the MXV22M controller configured in either 18-bit or 22-bit modes.

5.1. OPERATION USING RT-11

Operations involving the MXV22M controller are logically equivalent to those of the RXV21 except a modified "DY" driver is required when configured in 22-bit mode. Several techniques can be used to incorporate the changes described in section 5.1.1; however the changes cannot be performed on the "DY" via the MSV22M controller without attention to the caution noted in section 5.

The MXV22M (and RX02) controller requires a different handler than the programmed I/O controllers. This new handler is configured to utilize the DMA transfer scheme of the controller. In addition, diskette density is determined by the handler without system intervention, allowing the use of either single or double density diskettes interchangeably.

This handler, designated "DY" , is available in RT11-V03B and later revisions.

Although earlier versions of RT-11 can be used, only the changes to V4.0 and V5.0/V5.1 are provided in this document. Changes to earlier versions can be accomplished using the methodology described here and the judicious placement of similar code. Earlier version will also require modification to the Bootstrap program BSTRAP.

5.1.1. Modifying RX02 Driver for RT11

Changes listed in Appendix A and B are those required to modify DEC's V4.0 and/or V5.0/V5.1 RX02 driver for operation with the MXV22M controller. Changes listed in the Appendix are in a format expected by the Source Language Patch program (SLP). Generate a file "DYMXV.DIF" using an editor of your choice containing the appropriate changes listed in the Appendix. Use the following steps to include the changes:

```
.RUN SLP
*DYMXV=DY,DYMXV
```

NOTE

Changes to applicable drivers are identified by level and/or version numbers. Be certain that the correct level and/or version is being used as input to the program SLP.

The file DYMXV.MAC will contain the new MXV22 compatible driver. Assemble, link and install the new driver using the method described in your RT-11 System Generation Manual.

5.2. OPERATION WITH LAYERED PRODUCTS

Using the driver modifications described for RT-11, layered products such as TSX-Plus V3.01, and SHARE 11 can be used to provide 22-bit system support with the MXV22 controller.

5.3. OPERATION WITH RSX11M

Extended address support provided by the MXV22M controller is incorporated by the DYDRV changes listed in Appendix E. An additional change is required to the device data I/O structure. The file SYSTB.MAC created by Phase I of SYSGEN must be edited prior to assembly and task building process. Characteristic word one of the Unit Control Block (UCB) must be edited to reflect 22-bit direct addressing support for the DYDRV device. Refer to section 4 of the Guide to Writing An I/O Driver for details. An alternate method of changing the UCB is to use the utility Task/File Patch program ZAP. Use the RSX11M.MAP file to locate the UCB entries ".DY0" and ".DY1". The fifth word of these tables should be amended to include the setting of bit 8:

LOCATION	OLD VALUE	NEW VALUE
.DY0 + 10	dddddd	dddddd!400
.DY1 + 10	dddddd	dddddd!400

5.3.1. Modifying RX02 Driver RSX11M

Changes to the RX02 driver program DYDRV.MAC are provided in Appendix C. The changes are referenced to the distributed Version 3.02. Changes listed in Appendix C are in the format expected by the Source Language Input Program (SLP). These change can be incorporated into the standard driver using methods described in Section 17 of the RSX-11 Utilities Manual.

5.4. OPERATION WITH OTHER DEC SOFTWARE

The MXV22M Controller used in 18-bit mode emulates the operation of the RXV21. This is particularly important when using programs or systems that might require access to the RX02 type device.

NOTE

Operation of the XXDP Diagnostic program require that the MXV22M be configured to emulate the RXV21 in 18-bit mode.

5.5. OPERATION WITH DEC DIAGNOSTIC

The MXV22M Controller operates with the following DEC XXDP Diagnostic programs:

1. CZRXDAO RX02 SS Performance Exercise.
2. CZRXDBO RX02 SS Performance Exerciser.
3. CZRXEAO RX02 Formatter Program.

5.5.1. Exceptions

The following changes to the DEC RX02 Diagostics may be required when testing the MXV22 Controller.

Program ZRXFAO

The watchdog timeout interval for the set media density function may have to be increased when the processor is an LSI 11/23 or the selected step rate is six (6) milliseconds.

LOCATION	OLD VALUE	NEW VALUE
002474	000004	000200
012152	000004	000200
032720	000004	000200

Appendix A

The following changes are for incorporating 22-bit support into RT-11 Version 4.0 of DY.MAC Edit Level 2 via the Source Language Patch program (SLP):

```
-/.DRDEF/,,/;MXV22/
        .MCALL  .MTPS
-/.IIF NDF DY$DD/,,/;MXV22/
$22BIT = 1       ;REMOVE THIS LINE TO DISABLE 22-BIT ADDRESSING
.IIF NDF $22BIT, $22BIT = 0
.IF NE $22BIT
DYTYP=2000
.ENDC
-/@$MPPTR/+1,,/;MXV22/
.IF EQ $22BIT
-/BIS/,,/;MXV22/
.IFF
        MOV     @SP,-(SP)
        ASL     @SP
        ASL     @SP
        MOV     (SP)+,EXMBIT-1
        SWAB    @SP
        BIC     #^C<30000>,@SP
        BIS     (SP)+,R4
-/.ENDC/,,/;MXV22/
.ENDC
-/BUFRAD:/+3,,/;MXV22/
.IF NE $22BIT
        BIT     #40000,R0         ;BOUNDARY CROSSED?
        BEQ     8$                ;BRANCH IF NOT
        INCB    EXMBIT
        BIC     #40000,R0         ;REMOVE BIT
-/.ENDC/,,/;MXV22/
.ENDC
-/BPL   DYERR2/,.,/;MXV22/
        BPL     5$                ;BRIF DONE
.IF NE $22BIT
        BIT     #DYTYP,@R4        ;IN 22-BIT MODE
        BEQ     6$                ;BRIF NO
        MOV     EXMBIT-1,R0       ;EXTENDED ADDRESS BITS
        CLRB    R0                ;HOUSEKEEP
        BIS     R0,R3             ;INTO WORD COUNT REGISTER
6$:
.ENDC
```

```
-/BPL   DYERR2/,.,/;MXV22/
        BPL     5$              ;BRIF DONE
.IF NE $22BIT
        BIT     #DYTYP,@R4      ;PROTOCOL COMPLETE
        BNE     3$
2$:
.IFTF
-/MOV   R2,/,,/;MXV22/
.IFT
        .MTPS   #0              ;RESTORE STATUS
.ENDC
-/RTS/,,/;MXV22/
.IF NE $22BIT
3$:     .MTPS   #340            ;BLOCK INTERRUPTS
        MOV     R2,@R5
        MOVB    EXMBIT,R2
4$:     MOV     @R4,R0          ;GET STATUS
        BIT     #CSINIT!CSTR,R0 ;WAIT READY/INIT
        BNE     2$              ;LAST TRANSFER
        BIT     #CSDONE,R0      ;WAIT DONE
        BEQ     4$              ;FOR ONE
        .MTPS   #0              ;RESTORE STATUS
.IFTF
5$:     JMP     DYERR2
.IFT
        .BYTE   0
EXMBIT: .BYTE   0               ;EXTENDED MEMORY ADDR BITS
.ENDC
-/8$:/+2,,/;MXV22/
.IF NE $22BIT
        BIT     #DYTYP,@R4      ;22-BIT CONTROLLER?
        BEQ     22$             ;IF EQUAL NO!
        CLRB    23$
22$:
.ENDC
-/MOV/,,/;MXV22/
.IF NE $22BIT
23$:    BR      9$
        JSR     PC,WAIT         ;ONE MORE TRANSFER FOR 22-BIT
        CLR     @R5
.ENDC
-/WAIT:/,.,/;MXV22/
WAIT:   BIT     #CSINIT!CSTR,@R4;TRANSFER?
        BNE     11$
        BITB    #CSTR!CSDONE,@R4;TRANSFER OR DONE?
/
```

Appendix B

The following changes are for incorporating 22-bit support into RT-11 Version 5.0 or later of DY.MAC via the Source Language Patch program (SLP):

```
-/.DRDEF/,,/;22-BIT/
        .MCALL  .MTPS
-/DY$DD/,,/;22-BIT/
$22BIT = 1              ;REMOVE THIS LINE TO DISABLE 22-BIT ADDRESSING
.IIF NDF $22BIT, $22BIT =0      ;NO 22-BIT ADDRESSING
.IF NE $22BIT
DYTYP = 2000                    ;22-BIT CONTROLLER MODE
.ENDC
.IIF NDF DY$DS, DY$DS = 0       ;SINGLE HEADED FLOPPY DRIVE
.IIF NE DY$DS,  DY$DS = 1       ;DOUBLE SIDED FLOPPY DRIVE
-/@$MPPTR/+1,,/;22-BIT/
.IF EQ $22BIT                           ;NOT A MXV22
-/BIS   35$/,,/;22-BIT/
.IFF                                    ;MXV22
        MOV     @SP,-(SP)               ;MAKE ANOTHER COPY
        ASL     @SP                     ;POSITION ADDRESS INTO
        ASL     @SP                     ;UPPER BYTE
        MOV     (SP)+,EXMBIT-1          ;ODD ADDRESS CAPTURES ODD BYTE
        SWAB    @SP                     ;POSITION BITS TO 12 AND 13
        BIC     #^C<30000>,@SP          ;ISOLATE THEM
        BIS     (SP)+,R4                ;INCLUDE IN COMMAND WORD
.ENDC
-/ADD   #10000/,,/;22-BIT/
.IF NE $22BIT                           ;MXV22
        BIT     #40000,R0               ;BOUNDRY CROSSED?
        BEQ     8$                      ;BRANCH IF NOT
        INCB    EXMBIT                  ;DO 22-BIT ADDRESSING
        BIC     #40000,R0               ;REMOVE BIT
.ENDC
-/BPL   DYERR2/,.,/;22-BIT/
        BPL     5$                      ;BRIF DONE
.IF NE $22BIT
        BIT     #DYTYP,@R4              ;MXV22 IN 22-BIT MODE
        BEQ     6$                      ;BRIF NO
        MOV     EXMBIT-1,R0             ;EXTENDED ADDRESS BITS
        CLRB    R0                      ;HOUSEKEEP
        BIS     R0,R3                   ;INTO WORD COUNT REGISTER
6$:
.ENDC
```

```
-/BPL   DYERR2/,.,/;22-BIT/
        BPL     5$                      ;BRIF DONE
.IF NE $22BIT                           ;MXV22
        BIT     #DYTYP,@R4              ;PROTOCOL COMPLETE
        BNE     3$                      ;NO
2$:
.IFTF
-/MOV   R2,/,,/;22-BIT/
.IFT
        .MTPS   #0                      ;RESTORE SAVED STATUS
.ENDC
-.+1,,/;22-BIT/
.IF NE $22BIT
3$:     .MTPS   #340                    ;BLOCK INTERRUPTS
        MOV     R2,@R5
        MOVB    EXMBIT,R2               ;GET EXTENDED ADDRESS BITS
4$:     MOV     @R4,R0                  ;GET STATUS
        BIT     #CSINIT!CSTR,R0         ;WAIT READY/INIT
        BNE     2$                      ;LAST TRANSFER
        BIT     #CSDONE,R0              ;WAIT DONE
        BEQ     4$                      ;FOR ONE
        .MTPS   #0                      ;RESTORE STATUS
.IFTF
5$:     JMP     DYERR2                  ;GO PROCESS ERROR
.IFT
        .BYTE   0
EXMBIT: .BYTE   0                       ;EXTENDED MEMORY ADDR. BITS
.ENDC
-/8$:/+2,,/;22-BIT/
.IF NE $22BIT
        BIT     #DYTYP,@R4              ;22-BIT CONTROLLER?
        BEQ     22$                     ;IF EQUAL NO!
        CLRB    23$
22$:
.ENDC
-/MOV   R2,/,,/;22-BIT/
.IF NE $22BIT
23$:    BR      9$
        JSR     PC,WAIT                 ;ONE MORE TRANSFER FOR 22-BIT
        CLR     @R5
.ENDC
-/WAIT:/,.,/;22-BIT/
WAIT:   BIT     #CSINIT!CSTR,@R4        ;TRANSFER?
        BNE     11$
        BITB    #CSTR!CSDONE,@R4        ;TRANSFER OR DONE?
/
```

Appendix C

APPENDIX C
The following changes are for incorporating 22-Bit support for RSX-11M into DYDRV.MAC Version 3.02 via Source Language Input Program (SLP):

```
-/03.02/,,/;22-BIT/
MXV22   =       0                   ;ENABLE MXV22 CONDITIONAL CODE
-/SDEN/,,/;22-BIT/
ADREXT  =       2000                ; 22-BIT CONTROLLER BIT
-/M$$MGE/,,/;22-BIT/
        .IF DF  MXV22
        .IFF
-/.ENDC/,,/;22-BIT/
        .ENDC
-/M$$MGE/,,/;22-BIT/
        .IF DF  MXV22
        MOVB    U.BUF+1(R5),R0      ; EXTENDED MEMORY BITS
        ROR     R0
        ROR     R0
        MOVB    R0,I.PRM+16(R1)     ; SAVE BA18-BA21
        .REPT   3
        ROR     R0
        .ENDM
        BIC     #147777,R0          ; ISOLATE BA16 & BA17
        MOV     R0,U.BUF(R5)        ; INITIALIZE CSR WORD
        .IFF
-/.ENDC/,,/;22-BIT/
        .ENDC
-/M$$EXT/,,/;22-BIT/
        .IF DF  MXV22
        .IFF
-/.ENDC/,,/;22-BIT/
        .ENDC
-/140$:/,,/;22-BIT/
        BIT     #ADREXT,(R2)        ; 22-BIT
        BEQ     145$
        SWAB    R1                  ; GET UPPER BYTE
        BISB    I.PRM+16(R3),R1     ; SET BA 18-21
        SWAB    R1                  ; REPOSITION
 145$:
```

```
-/280$:/,.,/22-BIT/
        BR      281$            ; TRACK NO. COULD BE 111
280$:   CMP     R0,#'I          ; END REGISTER PROTOCOL
        BEQ     285$            ; IF EQ EXIT
281$:   BIT     #ADREXT,(R2)    ; 22-BIT CONTROLLER ?
        BEQ     285$            ; IF EQ NO
        MTPS    #340            ;;; BLOCK INTERRUPTS
        MOV     R0,RXDB(R2)     ;;; BUFFER ADDRESS
283$:   BIT     #TR!INIT,(R2)   ;;; READY FOR LAST WORD ?
        BNE     284$
        BITB    #TR!DONE,(R2)   ;;; READY FOR LAST WORD ?
        BEQ     283$            ;;;
        BMI     284$            ;;;
        MTPS    #0              ;;;
        BR      160$            ; ERROR, NO TRANSFER REQUEST
284$:   MOV     I.PRM+16(R3),R0 ;;; EXTENDED ADDRESS BITS
285$:   MOV     R0,RXDB(R2)     ; OR ;;; LOAD TRACK, BUFFER
                                ;;; ADDRESS, OR ASCII I
        MTPS    #0              ; OR ;;;
-/#10000/,,/;22-BIT/
        .IF DF  MXV22
        BIT     #40000,U.BUF(R5) ; CHECK FOR OVERFLOW
        BEQ     13$              ; NOT YET
        BIC     #40000,U.BUF(R5) ; HOUSEKEEP
        INC     I.PRM+16(R3)     ; UPDATE BA18-BA21
13$:
        .ENDC
/
```

Appendix F

The following changes are for incorporating double sided support for RSX-11M into DYDRV.MAC Version 3.02 via Source Language Input Program (SLP):

```
-/03.02/,,/;22-BIT/
-/RSAE/,,/;2SIDED/
SSIDED  =       20000           ;SECOND SIDE INDICATOR BIT (U.CW2)
-/DOUBLE/,,/;2SIDED/
DOUDOU  =       1976.           ; DOUBLE SIDED & DOUBLE DENSITY
-/SDEN/,,/;2SIDED/
DSIDED  =       1000            ; DOUBLE SIDED BIT (U.CW2)
-/CRCERR/,,/;2SIDED/
SIDES   =       2               ; DOUBLE SIDED MEDIA
-/220$:/,,/;2SIDED/
        BIT     #SSIDED,U.CW2(R5) ; IS IT SECOND SIDE OPERATION?
        BEQ     225$            ; IF EQ NO
        BIS     #DSIDED,U.BUF(R5) ; USE SECOND SIDE
225$:
-/420$:/,.,/;2SIDED/
420$:   BIC     #SILO!SCHAR!SSIDED!ERR1,U.CW2(R5) ; CLEAR BITS
-/440$:/,.,/;2SIDED/
;
;       RETRY WITH CORRECT DENSITY TO ENSURE VALID DISKETTE STATUS
;
        MOV     #DEN,R0         ; CHANGE DENSITY FOR RETRY
        XOR     R0,U.CW2(R5)    ; SET UP DENSITY BIT IN U.CW2(R5)
        BR      DYSEC           ; TRY AGAIN
440$:   BIC     #SCHAR!DEN!DSIDED,U.CW2(R5) ; CLEAR FLAGS
-/450$:/,.,/;2SIDED/
450$:   BIT     #SIDES,I.PRM+6(R1) ; IS IT DOUBLE SIDED?
        BEQ     455$            ; IF EQ NO
        MOV     #DOUDOU,U.CW3(R5)       ; DOUBLE THE MAXIMUM LBN'S
        BIS     #DSIDED,U.CW2(R5) ; SET THE DOUBLE SIDED BIT
455$:   MOV     #IS.SUC&377,R0  ; SET SUCCESS
-/460$:/,.,/;2SIDED/
460$:   BIT     #DEN,U.CW2(R5)  ; CHECK FOR CORRECT DENSITY
        BEQ     465$            ; SINGLE DENSITY
        BIS     #SDEN,R0        ; SET UP DOUBLE DENSITY
;
465$:   MOV     R0,(R2)         ; INITIATE FUNCTION
```

```
-/560$:/,.,/;2SIDED/
560$:   BIT     #SIDES,I.PRM+6(R1) ; IS IT DOUBLE SIDED?
        BEQ     565$               ; IF EQ NO
        BIS     #DSIDED,U.CW2(R5)  ; SET DOUBLE SIDED BIT
        MOV     #DOUDOU,U.CW3(R5)        ; DOUBLE MAX LBN'S
565$:   MOV     R0,I.PRM+10(R1)    ; STORE LOGICAL SECTOR NUMBER
-/#INTEBL/,.,/;2SIDED/
-/MOVB/,,/;2SIDED/
        BIS     #INTEBL,(R2)       ; ENABLE INTERRUPTS
-/,#77./,/BHI/,/;2SIDED/
        CMP     R0,#76.            ; IS IT SECOND SIDE?
-/BITB/,,/;2SIDED/
        BEQ     23$                ; IF EQ NO, IT'S A LOGICAL BLOCK
        CMP     #76.,R0            ; YES
        BEQ     30$                ; IF EQ ALLOW ACCESS TO #76.
        SUB     #77.,R0            ; CHANGE SIDES - PHYBLK ACCESS
        BR      25$                ; CHANGE READ HEADS
23$:    SUB     #76.,R0            ; ADJUST FOR SECOND SIDE
25$:    BIT     #DSIDED,U.CW2(R5)  ; TWO SIDE MEDIA?
-/BEQ/,,/;2SIDED/
        BIS     #SSIDED,U.CW2(R5)  ; SET HEAD 1 SELECT BIT
/
```

Micro Technology, Inc.

1620 Miraloma Avenue, Placentia, California 92670, (714) 632-7580