

DJ/DMA Floppy Disk Controller  
Technical Reference Manual

Revision 1  
April 1982

**MORROW DESIGNS**



---

Copyright (C) 1982 by Morrow Designs, Inc.

All rights reserved.

---

No part of this publication may be reproduced, transmitted, transcribed, stored in a retrieval system, or translated into any language or computer language, in any form or by any means, electronic, mechanical, magnetic, optical, chemical, manual or otherwise, without prior written permission of Morrow Designs, Inc.

---

#### DISCLAIMER

No representations or warranties, express or implied, are made with respect to the contents hereof, including, but not limited to, the implied warranty of merchantability or fitness for a particular purpose. Further, Morrow Designs, Inc., reserves the right to revise this publication and to make changes from time to time in the content hereof without obligation to notify any person of such revision.

---

Morrow Designs  
600 McCormick St.  
San Leandro, CA 94577  
(415) 430-1970

## IMPORTANT WARRANTY INFORMATION

### LIMITED WARRANTY

Morrow Designs, Incorporated, warrants its products to be free from defects in workmanship and materials for the periods indicated below. This warranty is limited to the repair and replacement of parts only and liability is limited to the wholesale list price of the product.

**Limitation of Liability:** The foregoing warranty is in lieu of all other warranties, expressed or implied, including, but not limited to, the implied warranties of merchantability and fitness for a particular purpose. In no event will Morrow Designs, Incorporated, be liable for consequential damages even if Morrow Designs, Incorporated, has been advised of the possibility of such damages.

This warranty is void if, in the sole opinion of Morrow Designs, Incorporated, the product has been subject to abuse or misuse.

Circuit boards - Parts, including the printed circuit board, purchased as factory assemblies, are warranted for a period of ninety (90) days from the original invoice/purchase date.

Electro-mechanical peripherals - Peripheral equipment such as floppy disk drives, etc., not manufactured by Morrow Designs, Incorporated, carry their own manufacturers' warranties. Please contact Morrow Designs' Customer Service Department for information regarding these peripheral devices.

Software/Firmware - Morrow Designs, Incorporated, makes no representations or warranties whatsoever with respect to software or firmware associated with its products and specifically disclaims any implied or expressed warranty of fitness for any particular purpose or compatibility with any hardware, operating system, or software/firmware. Morrow Designs, Incorporated, reserves the right to alter or update any program, publication or manual without obligation to notify any person of such changes.

## WARRANTY RETURN PROCEDURE

Should a customer experience a defect in either workmanship or materials during the warranty period, Morrow Designs, Incorporated will replace or repair the product at its expense only if the product is promptly returned to Morrow Designs, Incorporated for repair or replacement, and the following procedure for returning the product is followed:

1. Phone Morrow Design's Customer Service Department at (415) 430-1970. Inform the customer service staff of the nature of your problem and obtain a RETURN AUTHORIZATION NUMBER. No return shipment will be accepted without this number.
2. Repack the equipment and ship it to Morrow Designs, Incorporated, care of the Customer Service Department, 600 McCormick St., San Leandro, California, 94577. All freight charges must be prepaid by the customer, as well as all related charges, such as customs clearance and documentation. CODs WILL BE REFUSED. Indicate the RETURN AUTHORIZATION NUMBER on the waybill and shipping label. Include with the equipment a copy of proof of purchase showing the date the equipment was purchased. Any shipment received without proof of purchase will be billed as non-warranty repairs. Please also include a brief written description of the problem experienced with the equipment.
3. Morrow Designs, Incorporated, will repair or replace defective items and return the product to the customer via UPS surface, prepaid rates. Any other form of delivery or shipment required or requested by the customer is at the sole expense of the customer.
4. SHIPPING DAMAGE: (a) Morrow Designs, Incorporated, is not responsible for damage to goods in transit. (b) If you return a product because of shipping damage, have the product inspected by the carrier before returning it to us. Failure to do so may result in denial of your claim by the carrier. (c) Always ship the product in its original packing material. If the original packing material has been damaged or lost, new packing material can be purchased from Morrow Designs. Many shippers will not honor damage claims if the product is not adequately packaged.
5. Morrow Designs, Incorporated, is not responsible for the integrity of any data recorded on any media returned for service or repair. It is the responsibility of the user to back-up all necessary information.

# DJ/DMA Floppy Disk Controller

## Technical Manual

Revision 1

### Table of Contents

1. INTRODUCTION.....	1
2. PROGRAMMING SPECIFICATIONS.....	3
2.1. The Channel Concept.....	3
2.2. The Start Channel Command.....	4
2.3. The Channel Command Address.....	4
2.4. Command Structure.....	5
2.5. DJDMA Controller Commands.....	5
2.6. Controller Command Specifications.....	6
2.6.1. SET DMA ADDRESS.....	6
2.6.2. READ SECTOR.....	7
2.6.3. WRITE SECTOR.....	9
2.6.4. SENSE DRIVE STATUS .....	9
2.6.5. SET INTERRUPT REQUEST.....	12
2.6.6. SET ERROR RETRY COUNT.....	13
2.6.7. SET LOGICAL DRIVE.....	13
2.6.8. SET HEAD UNLOAD/DRIVE DESELECT TIMEOUT.....	14
2.6.9. READ TRACK.....	14
2.6.10. WRITE TRACK.....	15
2.6.11. OUTPUT TO SERIAL PORT.....	16
2.6.12. SERIAL INPUT ENABLE/DISABLE.....	16
2.6.13. CONTROLLER HALT.....	17
2.6.14. BRANCH IN CHANNEL.....	17
2.6.15. SET CHANNEL ADDRESS.....	17
2.6.16. SET TRACK SIZE.....	18
2.6.17. READ CONTROLLER MEMORY .....	18
2.6.18. WRITE CONTROLLER MEMORY.....	19
2.6.19. EXECUTE CONTROLLER ROUTINE.....	19
2.7. Command Summary.....	20
2.8. Status Codes.....	20
3. IEEE 696 (S-100) BUS CONSIDERATIONS.....	21
4. INTERRUPTS.....	21
5. I/O CONNECTORS.....	22
6. JUMPED SETTINGS.....	22
6.1. EPROM Replacement.....	22
6.2. Bootstrap Program.....	23

Table of Contents, Cont.

7. BOOTSTRAP LOAD.....	23
8. BOOTING THE DJDMA.....	24
9. FORMATTING DISKETTES.....	24
Software Listing.....	L-1
Component Layout/Schematic.....	S-1
Parts List.....	P-1
Subject Index.....	I-1

List of Tables

2-1. Status Byte Codes.....	8
2-2. STATUS BYTE 1: Drive Characteristic Byte.....	10
2-3. STATUS BYTE 2: Sector Length Code - 0, 1, 2, or 3.....	10
2-4. STATUS BYTE 3: Drive Status/Characteristic Byte.....	11
2-5. Supported Commands.....	20
2-6. Unsupported Commands.....	20
2-7. Status Code Summary.....	20
7-1. 19-Byte Handshake Routine.....	23

## 1. INTRODUCTION

The Disk Jockey/Direct Memory Access (DJDMA) Floppy Disk Controller is a single board S-100 subsystem. It communicates with both 8 inch and 5 1/4 inch floppy disk drives. Up to eight drives may be connected to the controller - with the limitation that no more than four of each type can be accommodated.

Special programmable bipolar LSI logic makes it possible to read and write media with almost any format, be it hard or soft sectored. Presently, the controller supports soft-sectored IBM compatible 8 inch media and hard-sectored North Star compatible 5 1/4 inch media. In the spring of 1982, IBM and Radio Shack 5 1/4 inch soft-sectored media will also be supported. Existing controllers in the field can be upgraded by replacing two of the ICs on the unit. This is done at moderate cost to the user.

The controller has its own Z-80 4MHz microprocessor which is used to supervise data transfers between the disk drive and the system memory without intervention of the main CPU. This relieves the main CPU of time consuming processes which include head positioning, rotational delays, and the usual byte-by-byte transfer of data from the diskette to main memory. As a result, transfers are faster and more efficient. Moreover, the main CPU has more time for data processing, and thus, supports more users and/or tasks.

The main advantage of the DJDMA controller over almost all the others is its "glitch free" direct memory access channel. This advanced channel concept allows the controller to communicate with S-100 memory by "stealing" bus cycles from the main CPU. This idea of an intelligent I/O channel was first implemented by IBM on their famous 370 mainframes. Now for the first time, this powerful concept has been implemented on the S100 bus.

The channel has the full 24-bits of memory addressing as described in the proposed IEEE standard for the S-100 bus. Also, a great deal of care has been taken in the design of the interface circuitry so it conforms in every detail to this new standard and still allows the controller to work well with existing systems designed before the standardization effort was started.

The controller is a temporary bus master, meaning that it has the same access to memory as the CPU whenever it has control. It also features priority logic which allows it to contend with up to sixteen other "temporary" masters that may also want to "steal" bus cycles from the main CPU, or the "permanent" master.

The controller acts as a temporary master (TMA). A temporary master may take control of the bus to perform a DMA operation. This is possible because both the TMA and the CPU drive control lines. The CPU, as permanent master, monitors signals from the TMA. When the TMA wants control, it first asserts a HOLD/ signal to the CPU. Assuming the TMA has priority, the CPU acknowledges

this signal upon completion of the present bus cycle by returning a processor hold acknowledge (pHLDA) signal. Upon receipt of this signal, the TMA enables its control line and asserts a control disable (CDSB) signal, disabling the CPU's control line. The TMA then disables the CPU's data-out, address and status lines using DODSB/, ADSB/ and SDSB/ signals. At that point the TMA has complete control to perform its DMA operation.

To return control to the CPU, the TMA first disables its own data-out, address and status lines, then re-enables the CPU's control lines, and simultaneously, its data-out, address and status lines. The TMA then releases its control line and makes false the HOLD/ signal, thus returning full control to the CPU.

So far, the process has been described as if only one temporary master wanted control of the bus. There can be up to 16 temporary masters on the bus. When there is more than one temporary master, they use the four DMA lines to decide who gets to assert HOLD/. Any device requesting the bus places its TMA priority level on the bus, and circuitry on the device decides if it has the highest priority. The device with the highest priority (0F hex is highest) asserts HOLD/. It removes its priority from the DMA lines when it receives pHLDA from the permanent master.

The features associated with the intelligent channel on the controller make it exceptionally desirable in multi-tasking and multi-user applications. In fact, many were tailored to enhance the performance of Morrow Designs new, powerful DECISION I multi-processing IEEE 696/S-100 machine. The DJDMA is an integral part of this advanced microcomputer system which incorporates many of the concepts originally introduced by IBM in their famous 370 series mainframes.

The DJDMA can boot itself up on the bus and even has a primitive serial port which is intended for diagnostic purposes or possibly even integrating the controller into a larger S-100 system that has I/O that the boot disk is not aware of. Under no circumstances can it be used as a general purpose serial port to the system, however, since it is inactive during disk activity.

All in all, there is nothing on the market in the way of an S-100 bus floppy disk controller that comes anywhere near the performance and versatility of the DJDMA. For that matter, we here at Morrow Designs know of no other floppy disk controller on any bus that can match the DJDMA in price, power, performance, and flexibility.

Good luck with this product. One of the purposes of this document is to detail how the DJDMA controller can improve the speed and performance of your system. If we've missed anything, please let us know.



## 2. PROGRAMMING SPECIFICATIONS

### 2.1. The Channel Concept

The IBM 370 mainframe was the first computer system to make use of the channel concept. In the traditional setting, an I/O controller, even one with direct memory access ability, was normally sent commands one at a time. Status was then reported through I/O ports after a command had completed.

One of the things a Direct Memory Access Controller does (and should do well) is communicate with main memory. Having realized this, someone very clever at IBM reasoned that if a controller could communicate with memory all that easily, why shouldn't it pick up its commands from memory as well? For that matter, why not have it lay down its status information in the CPU's main memory also?

Once the idea of picking up one command from memory is accepted, it is only a small step to think about placing strings of commands in memory and having the controller begin treating memory in the same way as the CPU does itself! That is, memory should be used for both instructions and data.

There is one detail missing in the above discussion. How is the controller to be started and stopped? A CPU starts running when power is turned on and continues (in theory) forever. But then there is the situation of a device whose primary job it is to transfer information to and from main memory and a mass storage device of some kind; it should remain idle until the CPU tells it otherwise.

A possible solution to the problem above is to have the device sample a memory location for a start command. At power-up, however, solid state memory does not have a predictable pattern. A start command could be present before it was actually issued by the CPU. The only foolproof way to issue a start command is through an I/O port. But doesn't that put us right back where we started? Actually, no.

It takes very little I/O circuitry to issue a simple pulse which can serve as a start command. It is also a small price to pay in cost and circuit board real estate for the flexibility and efficiency that is obtained.

Stop commands are much easier. Simply build an instruction into the controller's command set that forces it back to the idle state it was in just prior to the initial start pulse issued by the CPU.

Obviously, a channel type of controller needs some kind of on-board intelligence. At the time that IBM first built this kind of device, it was expensive both in terms of dollars and in circuit board real estate to implement this intelligence. Today

## Programming Specifications

however, the situation is quite different. Microprocessors are inexpensive and take only a modest amount of space on a circuit board.

In theory, the only limitation to the power and flexibility of a channel driven controller is the size of the memory local to the resident microprocessor. Since memory is getting denser and cheaper, it would seem that time will favor the channel approach to I/O controllers.

### 2.2. The Start Channel Command

Just as in the general case discussed above, there is a single primitive I/O port on the DJDMA. It resides at location EF (hex) unless a custom unit has been ordered with a special I/O address. This port's only purpose is to send start pulses to the DJDMA controller. **Any output instruction to port EF (hex) starts the DJDMA.** It doesn't matter what value is sent nor does it matter what kind of device sends the data. Any time any output reference is made to this port by the main CPU permanent master, or even by a temporary master, the DJDMA begins fetching and executing commands. Where these commands come from and how they work is taken up below.

### 2.3. The Channel Command Address

When the DJDMA first powers up or is reset, there is a three-byte pointer initialized in its local memory. This pointer determines where the controller picks up its first command when a start pulse is issued via I/O port EF (hex).

There are actually two of these three-byte values the DJDMA maintains. The first points to where it should start its command sequence. The second points to where it should get its next command in the event that the current one is not a halt command. The user needs to be aware of both of these pointers as he sets up command sequences for the controller to execute.

The second pointer has the same function as the program counter of the main CPU: it always points to the next command that the controller will execute. The first pointer is similar to the value forced into the program counter (PC) of the main CPU when a reset signal is issued. In most cases, a reset signal forces a 0 into the PC. The processor commences to fetch instructions at this value.

The same is true for the DJDMA, except that the value is not zero. Also, unlike the CPU, this initial location can be changed by a sending the proper command to the controller. **The initial location that the DJDMA controller begins fetching commands from is 50 (hex).** The command that alters this starting location is described in the next section.

#### 2.4. Command Structure

Commands to the DJDMA controller are at least two bytes long. The first byte is always the command code. Parameter lists follow the command byte (if needed) and the command status byte (if needed) comes at the end of the command string. The length of a command string varies with the command. Unless a branch in channel command is issued, commands must be arranged in memory one after the other with no gaps between the end of one command and the beginning of another. Sequences of commands must be terminated with either a controller halt command or a branch in channel command. If a sequence ends with a branch in channel command, another sequence of commands must be present at the location specified in the address parameter list of the branch in channel command.

#### 2.5. DJDMA Controller Commands

The Disk Jockey DMA controller recognizes the following commands:

- SET DMA ADDRESS
- READ A SECTOR
- WRITE A SECTOR
- SENSE DRIVE STATUS
- SET INTERRUPT REQUEST
- SET ERROR RETRY COUNT
- READ TRACK
- WRITE TRACK
- OUTPUT SERIAL PORT
- SERIAL INPUT ENABLE/DISABLE
- CONTROLLER HALT
- BRANCH IN CHANNEL
- SET CHANNEL ADDRESS
- SET TRACK SIZE
- SET DRIVE DESELECT/HEAD UNLOAD TIMEOUT
- SET LOGICAL DRIVE
- READ CONTROLLER MEMORY
- WRITE CONTROLLER MEMORY
- BRANCH TO CONTROLLER ROUTINE

The last three commands require great care to use. They are used to format diskettes and will be used to support media formats which are not yet implemented. Improper use of any of the last three commands could produce unpredictable results and may cause the loss of information on write-enabled diskettes in drives connected to the controller. It could also cause the controller to be inoperative until a bus reset is performed.

Morrow Designs will have a separate document (at extra cost) that describes the firmware on the DJDMA controller. This information should be available at the end of first quarter 1982 or early second quarter. Thus, users with special applications will have a way to extend the command structure of the DJDMA controller. However, extended commands will not be supported by Morrow Designs and we cannot stress too strongly that efforts in this direction will require a great deal time and expertise to complete and debug.

## 2.6. Controller Command Specifications

Specifications for each of the controller commands are described in the following sections. In many instances, examples are given to fully illustrate use of the command.

### 2.6.1. SET DMA ADDRESS

Command code:	23 (hex)
Command length:	4 bytes
Command parameter list length:	3 bytes
Command status list length:	0 bytes

The command length is four bytes. The first byte is the command code: 23 (hex). The next three bytes specify a 24-bit address in main memory where data is written to or read from during subsequent disk transfers. This field must be arranged so that the least significant byte of the address directly follows the command byte. The byte of next highest significance follows. The highest order byte of the address is last. The last byte specifies an extended page as defined in the proposed IEEE standard for the S-100 bus and allows memory addressing to be extended to 16 million bytes.

In systems that do not support this new extended addressing, the value of this high order byte is not important. However, it must be present - whether it is used or not. Other commands which have three byte address fields in their parameter list require the same byte significance order as described above. The firmware that processes commands on the DJDMA expects all address fields to be three bytes long - even if only two of the three have effect on the address bus of the system.

The following example is a command that sets the DMA address of the controller to location 80 (hex) - the default disk data buffer of the popular CP/M operating system:

23 80 00 00 (hex).

2.6.2. READ SECTOR

Command code:	20 (hex)
Command length:	5 bytes
Command parameter list length:	3 bytes
Command status list length:	1 byte

The three-byte parameter field following the command code consists of

1. track
2. side/sector
3. drive

in that order. The side select is encoded in the high order bit of the sector field and merged together to form the second byte in the parameter list. The third byte determines which of eight possible drives are read. If the system has been booted up from a 5 1/4 inch drive, drives 0 through 3 specify this; drives 4 through 7 specify 8 inch drives. If the system has been booted from an 8 inch drive, the numbering is reversed with the first four being 8 inch drives and the last four being 5 1/4 inch. The following example is a command that reads data from sector 3 of track 5 on side 1 of drive 0:

20 05 83 00 00

The last zero is provided so that the controller can fill in the status of the transfer after it has completed the read. Here is a second example that reads sector 2 from track 6 on side 0 of drive 1:

20 06 02 01 00

Again, the last byte is for status reporting and it must be there.

The length of the sector (and consequently a valid range of sector values) depends on what size drive is being addressed and how the media has been formatted. In the media currently supported, the following sector values and data field lengths are relevant:

5 1/4" hard sectored single density:	0 - 9	256 bytes
5 1/4" hard sectored double density:	0 - 9	512 bytes
8" soft sectored single density:	1 - 26	128 bytes
8" soft sectored double density:	1 - 26	256 bytes
8" soft sectored double density:	1 - 15	512 bytes
8" soft sectored double density:	1 - 8	1024 bytes

The numbers in the above list are all decimal. The sector size, density, and valid range of values for the sector

## Command Specifications

number are all determined automatically by the controller. The controller can inform the system of these parameters by executing the SENSE DRIVE STATUS command which is taken up below. These details are presented here because it is necessary to know how much space the controller will use when data is read from the disk into main memory. Also, an error occurs if incorrect values are specified for the sector, track, or drive.

All 8 inch drives presently have 77 tracks numbered 0 through 76. This is not the case with 5 1/4 inch drives. Some have 35 tracks numbered 0 through 34, others have 40 tracks numbered 0 through 39, and finally, the new double track density 5 1/4 inch drives have 80 tracks numbered 0 through 79. The default value for 5 1/4 inch drives on the DJDMA is 40. However, this value can be changed by executing a SET TRACK SIZE command which is discussed below.

The last byte in the read sector command is called the status byte. This byte should be filled with some value other than what the controller might use when it reports status after the command is completed. A 0 is ideal since the controller does not use this value. For that matter, it does not use FF either. Either of these values are handy since they can be tested easily. By testing the status byte, the system can determine when a read command (among others) has completed. Below is a list of status byte codes along with their meanings. All values are in hex.

Table 2-1. Status Byte Codes

40 -	normal completion - no errors
80 -	improper command code
81 -	illegal disk drive value
82 -	drive not ready
83 -	illegal track value
84 -	unreadable media
85 -	improper sector header - no sync byte
86 -	CRC error in sector header read
87 -	seek error
88-8D -	compare error in sector header scan
8E -	CRC error in data field
8F -	illegal sector value for current media
90 -	media is write protected (writing only)
91 -	lost data - DMA channel did not respond
92 -	lost command - channel did not respond

The above list is complete and applies to any command that reports status in its last byte. Not all codes apply to all commands. For example, 90 (hex) never appears as the status reported by the READ SECTOR command.

## 2.6.3. WRITE SECTOR

Command code:	21 (hex)
Command length:	5 bytes
Command parameter list length:	3 bytes
Command status list length:	1 byte

The three-byte parameter field and the status byte have the same properties as those in the read sector command. All the items discussed in the read sector command apply to the write sector command with the exception that the write sector command can report a media write protect error (90 hex).

## 2.6.4. SENSE DRIVE STATUS

Command code:	22 (hex)
Command length:	6 bytes
Command parameter list length:	1 byte
Command status list length:	4 bytes

The single byte in the parameter list specifies a drive. Legal values range from 0 to 7. The last byte of the status list has codes which were listed above in the READ SECTOR command. The first three bytes of status are peculiar to a specific drive and are detailed below. However, unless the last status byte contains a 40 (hex), the preceding three bytes do not accurately reflect the condition and characteristics of the drive whose status was supposed to be sensed.

If any value other than 40 (hex) is present, nothing can be learned from the first three status bytes. When the final byte contains a 40 (hex), the first three describe characteristics and status concerning the drive specified in the parameter byte of the command.

## Command Specifications

**Table 2-2. STATUS BYTE 1: Drive Characteristic Byte**

Each bit in this byte describes a different characteristic of the drive specified in the parameter field of the command.

- Bit 0 - Information internal to the controller.
- Bit 1 - If the media is hard-sectored, this bit is a 1. When the media in the drive is soft-sectored this bit will be a 0.
- Bit 2 - If the drive is 5 1/4 inch, this bit is a 1. If the drive is 8 inch, the bit is a 0.
- Bit 3 - If the drive has a DC motor with an ON/OFF switch, this bit is a 1. If there is no ON/OFF switch, or if the drive motor is AC, this bit is a 0.
- Bit 4 - If the media in the drive is double density, this bit is a 1. It is 0 only if the media is single density.
- Bit 5 - If this bit is a 1 there is no "drive ready" signal supplied by the drive. For drives with no "ready" signal, the DJDMA firmware tests for the presence of sector/index holes. If the drive has an active "ready" signal, this bit is a 0.
- Bit 6 - If there is no "head load" command line to the drive, the controller assumes that the head(s) are always loaded against the media and this bit is a 1. If there is a "head load" command line to the drive, this bit is a 0.
- Bit 7 - If the head(s) are currently loaded against the media, this bit is a 1. If the head(s) are not loaded, this bit is a 0.

**Table 2-3. STATUS BYTE 2: Sector Length Code - 0, 1, 2, or 3**

The 0 indicates a sector length of 128 bytes, 1 stands for a length of 256 bytes, 2 means that the length is 512 bytes, and 3 indicates that the sector is 1024 bytes long. These are all decimal numbers.



Table 2-4. STATUS BYTE 3: Drive Status/Characteristic Byte

There is an input port on the controller which can examine status signals transmitted directly from the selected drive.

The third status byte is a direct image of this port.

- Bit 0 - Used internally by the controller and is of no meaning to the system.
- Bit 1 - Current status of the serial input line from an RS-232 device which may be attached to connector P3, the serial port of the controller.
- Bit 2 - This bit indicates that a double-sided 8 inch drive is currently selected and that double-sided media is present in the drive. This line is not driven by 5 1/4 inch drives; thus, an indirect means must be employed to determine if a 5 1/4 inch drive is double-sided and has double-sided media in it.
- Bit 3 - Currently not used.
- Bit 4 - This is the index/sector hole indicator. If this bit is a 1, the drive has sensed the presence of either an index hole or a sector hole.
- Bit 5 - If this bit is a 1, the head(s) of the drive are at Track 0. If the head(s) are positioned over some other track, this bit is a 0.
- Bit 6 - This bit is a 1 if the media in the drive is write protected. A zero indicates that the media is not write protected and disk write commands do not produce "write protect" errors.
- Bit 7 - This is the drive ready bit. Most 5 1/4 inch drives have no signal on this line; thus, it is not a good "drive ready" indicator in this case.

All 8 inch drives produce a "ready" signal at this bit. If the current drive is an 8 inch and this bit is 1, the drive is "ready" to accept read, write, or step commands. If it is a 0, the 8 inch drive is not "ready" and will not respond to commands from the controller.

## Command Specifications

### 2.6.5. SET INTERRUPT REQUEST

Command code:	24 (hex)
Command length:	2 bytes
Command parameter list length:	0 bytes
Command status list length:	1 byte

This command generates an interrupt to the system bus. There is a bus driver on the DJDMA circuit board whose output terminates at a jumper pad near the lower edge of the board (the exact location is described later in the manual). This jumper pad is arranged so that the driver can be connected to the main interrupt line of the system bus (PINT\*) or any one of the eight vectored interrupt lines (VI0\*, VI1\*, ... VI7\*).

The controller is shipped from the factory with the driver uncommitted. If the DJDMA is to generate interrupts to the system, this driver must be connected to one of the nine interrupt lines. If the driver is not connected, the INTERRUPT REQUEST command causes the controller to pause until another start pulse is issued by the system. However, once an INTERRUPT REQUEST command is executed, the controller is put into a special state where the board responds differently to the start pulse than it usually does.

Normally a start pulse causes the controller to begin fetching commands at the location specified by the most recent channel command word address. When the DJDMA executes an INTERRUPT REQUEST, it activates the interrupt bus driver on the circuit board. It then pauses with this bus driver still active.

Upon receipt of the next start pulse, the controller turns off the bus driver generating the interrupt and fetches the command **which immediately follows the interrupt request command**. The controller thus treats the first start pulse issued after the interrupt request command has completed as an **INTERRUPT ACKNOWLEDGE handshake signal**. This is the only circumstance in which a start pulse to the controller does not cause the command pointer to be reset.

The system can test the status byte following the command code to determine when the command has completed. When the command completes, it fills the status byte with a 40 (hex). When the interrupt request bus driver is not connected, an interrupt request command causes the controller to pause until the next start pulse is received, at which time it resumes executing commands where it left off.

## 2.6.6. SET ERROR RETRY COUNT

Command Code:	28 (hex)
Command length:	2 bytes
Command parameter list length:	1 byte
Command status list length:	0 bytes

This command specifies how many times a sector is read in the event that a CRC error occurs in the data field. At least one read always takes place, so the smallest value that should appear in the parameter byte is a 1. This value can be as high as 255 (decimal). The default value is 10 (decimal).

This command's main purpose is to ensure that the value can be made smaller for diagnostic purposes. It is also useful when a diskette becomes worn and data recovery becomes more difficult. In this case, the value is made larger.

## 2.6.7. SET LOGICAL DRIVE

Command code:	2E (hex)
Command length:	3 bytes
Command parameter list length:	1 byte
Command status list length:	1 byte

This command allows the user to change the logical numbering assigned to the 8 inch and 5 1/4 inch drives. The default values assigned to the 8 inch drives are 0 through 3, while the 5 1/4 inch drives are assigned values 4 through 7.

If a 4 appears in the parameter list of this command, the 5 1/4 inch drives are assigned drive values 0 through 3, while the 8 inch drives have their values changed to 4 through 7. A 0 in the parameter field reverses these values to the original default values. There is no status byte associated with this command and bit-2 in the parameter field is the only part of the byte examined by the command.

The status byte reported by the command reflects the logical value of the first physical 8 inch drive prior to the execution of the SET LOGICAL DRIVE command. If the status is 40 (hex), the previous logical value of the first physical 8 inch drive was 0. If the status is 44 (hex), the old value was 4.

The logical values assigned to the drives are also affected by performing a bootstrap operation which is discussed later.

## Command Specifications

### 2.6.8. SET HEAD UNLOAD/DRIVE DESELECT TIMEOUT

Command Code:	2F (hex)
Command length:	2 bytes
Command parameter list length:	1 byte
Command status list length:	0 bytes

In order to conserve power and maximize diskette life, during periods of disk inactivity the controller unloads the drive head(s) and deselects the drive after a certain number of revolutions of the diskette. Normally, the controller waits sixteen revolutions before it deselects a drive. This command allows the user to change this situation. The value in the parameter list determines how many revolutions occur after no disk activity before the head(s) are unloaded and the drive is deselected. A disk transfer operation requires more time if the drive is not selected and so, under certain conditions, it may be desirable to extend the time before a drive is deselected after a transfer occurs. This command makes it possible to affect this situation. The value in the parameter field should be between 1 and 255 (decimal). However, when the heads are loaded for extended periods of time with the motor running, diskette media life is shortened considerably.

### 2.6.9 READ TRACK

Command code:	29 (hex)
Command length:	8 bytes
Command parameter list length:	6 bytes
Command status list length:	1 byte

This command reads an entire track into main memory starting at the value specified by the most recent SET DMA ADDRESS command. The transfer begins with the first full sector encountered by the controller. Thus, the buffer may not fill from the beginning.

As an example, suppose that the diskette had eight 1024 byte sectors and the first full sector of data encountered was Sector 6. In this case the last 3072 bytes of the buffer would be filled with Sectors 6, 7, and 8. The DJDMA memory pointer would then be reset to the start of the track buffer and Sectors 1 through 5 would be transferred.

The first three bytes of the parameter list specify

1. track
2. side
3. drive

in that order. The side bit must appear in the most significant bit of the byte. Thus, the second byte in the parameter list is either 0 or 80 (hex). The last three bytes of the parameter list form a memory pointer to a sector table.

There must be an entry in this table for each sector on the track.

As an example, if the diskette in the selected drive had 512 byte sectors, there would be fifteen entries and the table length would also be fifteen. This table should be initialized with 0s, 80s (hex), or FFs (hex).

As a sector of the track is read, the controller fills the byte of the table corresponding to the sector with status information concerning that particular sector (assuming the initial entry was 0). Thus, the system can determine error information individually, sector by sector.

If the controller encounters an FF (hex) entry in the sector table, it skips that sector which corresponds to the entry.

If a whole section of the table has FFs, the sectors corresponding to this section are not read.

If the controller encounters an entry in the table of 80 (hex), the READ TRACK command terminates at that point. An example should illustrate these ideas.

Suppose side 1 of track 23 (decimal) is to be read into a track buffer starting at location 00E000 (hex) from drive 2 and that a set DMA address command with this value has already been executed. Suppose also that there are 1024 byte sectors on the diskette and that the sector table is to immediately precede the track buffer in memory. The command to read the track would then appear as follows:

29 17 80 02 F8 DF 00 00

The sector table address of 00DFF8 (hex) has a value of eight less than 00E000 (hex) since there are eight sectors on the track of the diskette. The last byte (indicated with a value of 00) is the overall status byte for the command. The status codes are the same as the READ SECTOR COMMAND where they are listed.

#### 2.6.10. WRITE TRACK

Command Code:	2A (hex)
Command length:	8 bytes
Command parameter list length:	6 bytes
Command status list length:	1 byte

The write track command is similar to the READ TRACK command. The six bytes of the parameter list are exactly the same and even the sector table entries work the same. Normally, the table has 0s as entries. Sectors that are not to be written (or rewritten) are marked with FFs (hex) while an 80 (hex) causes the command to terminate.

## Command Specifications

As with the read track command, the starting address of the track buffer is initialized with a SET DMA ADDRESS command.

### 2.6.11. OUTPUT TO SERIAL PORT

Command code:	2B (hex)
Command length:	3 bytes
Command parameter list length:	1 byte
Command status list length:	1 byte

This command communicates with the output portion of the bit serial port on the DJDMA. The parameter byte is filled with the ASCII value that is to be transmitted to the RS-232 device connected to the port. The status byte should be initialized to either 0 or FF (hex). The command fills the status byte with a 40 (hex) when all eight data bits and two stop bits have been transmitted.

The speed of this serial port is 9600 baud and cannot be changed. Also, it is vital that the system refrain from sending new start pulses to the controller until this command has completed. Otherwise, transmission of the serial stream is aborted before any or all of the bits have been sent.

The main purpose of the port in this subsystem is to allow a user to boot-up in a system where I/O devices are not defined on the boot diskette. This port is not adequate as a system consul port and will cause the controller to run less efficiently while the port is active (there is no disk activity while the serial port is engaged in data transmission). Input serial data can also be easily lost if the controller is supervising data transfer to or from a disk drive.

The input side of this serial port does not work the same as the output and is discussed in the next command.

### 2.6.12. SERIAL INPUT ENABLE/DISABLE

Command Code:	2C (hex)
Command length:	2 bytes
Command parameter list length:	1 byte
Command status list length:	0 bytes

This command enables or disables input from the bit serial RS-232 port on the controller. Serial input operates in a slightly different manner than serial output. If the input side of the port is enabled, characters received by the port are deposited at location 00003E (hex).

After loading a new character at this location, the controller writes 40 (hex) at location 00003F (hex). This second location serves as a status flag for serial input and should be reset to some other value after reading the character.

In the enable/disable command, the value of the parameter byte determines whether the port is to be enabled or disabled. A 0 in this byte instructs the controller to turn off the port, while a 1 forces the DJDMA to enable input. At boot-up, input is enabled, but if there is no terminal connected to the board, it is automatically disabled.

#### 2.6.13. CONTROLLER HALT

Command code:	25 (hex)
Command length:	2 bytes
Command parameter list length:	0 bytes
Command status list length:	1 byte

This command is used to halt the DJDMA controller. There are no parameters. The status byte should be initialized to 0 or FF (hex). The controller fills this byte with a 40 (hex) when the command completes. As mentioned previously, this command resets the command pointer. Hence, the next start pulse causes the controller to begin fetching commands from the channel command word address which has an initial value of 000050 (hex). This value can be changed with a command that is described below.

#### 2.6.14. BRANCH IN CHANNEL

Command code:	26 (hex)
Command length:	4 bytes
Command parameter list length:	3 bytes
Command status list length:	0 bytes

The three parameter bytes specify a branch address for the controller. This address is the location from where the controller fetches its next command. The address bytes are arranged so that the low order byte immediately follows the command code, the middle order byte is next and the high order byte is last. There is no status code and immediately after execution, the controller picks up the next command from the branch address.

#### 2.6.15. SET CHANNEL ADDRESS

Command code:	27 (hex)
Command length:	4 bytes
Command parameter list length:	3 bytes
Command status list length:	0 bytes

The three parameter bytes of this command specify a memory address. After this command has executed, start pulses from the system cause the controller to fetch its first instruction at this address. The order of the bytes is the same as the branch in channel command. There is no status byte associated with this command.

## Command Specifications

### 2.6.16. SET TRACK SIZE

Command Code:	2D (hex)
Command length:	4 bytes
Command parameter list length:	2 bytes
Command status list length:	1 byte

This command allows the system to change the number of tracks that the controller assumes are on a disk drive. The first byte in the parameter list describes a drive and should have values between 0 and 7. Other values cause the command to return an error and not change the track value of any drive.

The second byte must contain a hex number which is one larger than the largest numerical track on the diskette. For 35 track drives, this value is 35 since the track numbering starts at zero. For the same reason, the value is 40 for 40 track drives, 77 for 77 track drives, and 80 for 80 track drives. (All the numbers used in this paragraph are decimal. They must be changed to hexadecimal when incorporated into the command string.)

It is possible to damage a drive if seeks are performed to tracks which extend beyond the boundaries of the seek mechanism. The controller has no way to determine if a particular value is improper for a given drive. The user must exercise care in executing this command and Morrow Designs takes no responsibility for damage that occurs through its misuse.

### 2.6.17. READ CONTROLLER MEMORY

Command Code:	A0 (hex)
Command length:	8 bytes
Command parameter list length:	7 bytes
Command status list length:	0 bytes

The first three bytes of the parameter list specify a main memory address with bytes in ascending order (just like the other commands that required a three-byte address field.)

The next two bytes specify a count which can have values anywhere between 0 and FFFF (hex). The last two bytes specify an address in the memory of the on-board Z-80A microprocessor. This command transfers local memory to main memory which allows the main CPU to read the controller's memory. It is not advisable to read locations 4001 (hex), 8001 (hex), A000 (hex), etc., since this type of reference causes the controller to hang waiting for data from a drive when none is selected. The only way to reliably recover from this fault is to issue a reset to the system. Morrow Designs does not recommend use this command and does not support applications that make use of this command or the two that follow. This command reports no status.



## 2.6.18. WRITE CONTROLLER MEMORY

Command Code:	A1 (hex)
Command length:	8 bytes
Command parameter list length:	7 bytes
Command status list length:	0 bytes

The first three bytes of the parameter list specify a main memory address in ascending order (just like the other commands that required a three-byte address field.)

The next two specify a count that can range between 0 and FFFF (hex).

The last two bytes specify an address in the memory space of the on-board Z-80A microprocessor. This command transfers data from main memory to the memory of the controller. There are only 1024 bytes of RAM on the controller board. This RAM starts at location 1000 (hex). The only locations safe to write in are between 1030 and 127F (hex). Writing in other locations produces unpredictable results and can lead to loss of data on diskettes which are not write protected and are inserted in drives connected to the controller. **Morrow Designs does not support the use of this command. This command is used in diskette format programs (included in this manual) but we strongly recommend that it not be used for other purposes).** There is no status byte associated with this command.

## 2.6.19. EXECUTE CONTROLLER ROUTINE

Command Code:	A2 (hex)
Command length:	3+ bytes
Command parameter list length:	2 bytes
Command status list length:	0+ bytes

The two bytes in the parameter list specify an address in the memory space of the on-board Z-80A microprocessor. This command forces the on-board processor to branch to and begin executing instructions at this address. As with the previous command, it is extremely dangerous and should not be used by anyone except those well versed with the inner workings of the controller. The status list length is given as 0+ bytes because the length and type of status varies depending on the nature of the routine at the specified address. **As with the previous two commands, Morrow Designs does not support use of this command.**

## Command Summary

### 2.7. Command Summary

The following tables summarize commands that are both supported and unsupported by the DJDMA.

**Table 2-5. Supported Commands**

- Set DMA (low, med, high)
- Read Sector (track, side/sector, drive, status)
- Write Sector (track, side/sector, drive, status)
- Sense Status (dstat1, dstat2, dstat3, status)
- Set Interrupt Request (status)
- Set Error Retry Count (count)
- Set Logical Drive (drive, type)
- Set Head Unload/Drive Deselect Timeout (revolution count)
- Read Track (track, side, drive, low, med, high, status)
- Write Track (track, side, drive, low, med, high, status)
- Serial Port Output (ASCII byte)
- Serial Input Enable/disable (control byte)
- Controller Halt (status)
- Branch in Channel (low, med, high)
- Set Channel Address (low, med, high)
- Set Track Size (drive, hitrack)

**Table 2-6. Unsupported Commands**

- Read CMemory (tlow, tmed, thigh, lcnt, hcnt, slow, shigh)
- Write CMemory (slow, smed, shigh, lcnt, hcnt, tlow, thigh)
- Execute Controller Routine (low, high, ..., ...)

### 2.8. Status Codes

The following table summarizes the DJDMA status codes.

**Table 2-7. Status Code Summary**

STATUS CODE	DESCRIPTION
40 .....	Normal completion - no error encountered
80 .....	Improper Command Code
81 .....	Improper Disk Drive Value
82 .....	Disk Drive Not Ready
83 .....	Improper Track Value
84 .....	Unreadable Media
85 .....	Improper Sector Header - No Sync Byte(s)
86 .....	CRC Error in Sector Header Scan
87 .....	Seek Error
88 - 8D .....	Compare Error in Sector Header Scan
8E .....	CRC Error in Data Field
8F .....	Improper Sector Value
90 .....	Media Write Protected
91 .....	Lost Data - DMA Channel did not respond
92 .....	Lost Command - Channel did not respond

### 3. IEEE 696 (S-100) BUS CONSIDERATIONS

The DJDMA controller has been designed to meet the IEEE/696 proposed standard for the S-100 bus and will operate properly in any S-100 mainframe which meets this proposed standard and can accommodate temporary bus masters. In fact, the DJDMA runs in most existing S-100 systems in operation today. However, we cannot guarantee that the controller will operate in a system unless it meets all the specifications contained in the IEEE/696 document.

In transferring data from a floppy disk directly into main memory, the DJDMA assumes that the permanent master in the system will respond to bus requests by the controller fast enough so that data will not be lost. If an 8 inch double density drive is connected to the controller, a byte of data is read or written every 16 microseconds.

The transfer rate for single density 8 inch drives and double density 5 1/4 inch drives is a byte every 32 microseconds.

Single density 5 1/4 inch drives have a transfer rate of one byte every 64 microseconds. If some device, such as a front panel, holds the READY line of the bus down for extended periods during disk transfers, data is lost and the controller cannot function properly.

Morrow Designs assumes that the user has made the proper determination concerning the ability of his system to respond to bus requests from the DJDMA so that data is not lost during disk transfers. Morrow Designs is not responsible for operation of the controller in systems that cannot respond to bus requests at least as fast as those detailed above for the various types of floppy disk drives.

### 4. INTERRUPTS

At the lower left area of the DJDMA circuit board, just above the edge connector fingers, is a jumper area designed so users can connect the board's interrupt request bus driver to one of the nine interrupt request lines: VI0\*, VI1\*, VI2\*, VI3\*, VI4\*, VI5\*, VI6\*, VI7\*, or PINT\* (See the component layout for an illustration of this area).

If the system does not use interrupts, there is no need to connect J3 to any of these lines. If J3 is not jumpered, it appears to the system that the controller has entered a pause state when it executes an interrupt request command. All activity stops (just as it does after a halt command). When the next start pulse is sent to the controller, it picks up its next instruction from the memory location immediately following the status byte of the interrupt request command (this is not the same as a halt command).

The DJDMA is shipped from the factory without any jumpering between J3 and the interrupt request lines. If the controller is to generate interrupt requests, the user must determine which of the nine possible connections is appropriate for his system. The DECISION I user reference manuals contain information about how the DJDMA communicates with the interrupt controller on the MULT-I/O and WUNDERBUSS I/O boards, and should serve as an example of how interrupts from the DJDMA could work in other systems.

## 5. I/O CONNECTORS

Refer to the component layout drawing included in this manual for a more complete understanding of the discussion in this section.

There are three I/O connectors at the top of the DJDMA circuit board: P1, P2, and P3.

P3 is at the top left-hand side of the board and is the connector for the bit serial RS-232 port. It has three pins, numbered 1 through 3 from left to right. Pin-1 is the RS-232 ground signal, pin-2 is the input and pin-3 is the RS-232 output signal.

To the right of P3 is P2. P2 has 34 pins and is used to connect 5 1/4 inch drives to the controller. The pins are arranged in two rows - the odd numbered pins being just above the even numbered ones. The pins are numbered 1 through 33, odd from right to left, and 2 through 34, even from right to left. All the odd numbered pins are connected to ground while the even numbered pins carry information to and from 5 1/4 inch floppy disk drives.

P1 is the right-most connector and has 50 pins. This connector is used to connect 8 inch drives to the controller and has pins arranged in two rows, the same as P2. The upper pins are odd and are numbered 1 through 49, right to left. The lower pins are even and are numbered 2 to 50, right to left. As before, all odd pins are grounds while even pins carry signals between the controller and 8 inch drives.

## 6. JUMPERED SETTINGS

Refer to the component layout drawing included in this manual for a more complete understanding of the discussion in this section.

### 6.1. EPROM Replacement

The jumpered setting at J1 (located in the upper right hand corner of the board) is factory set B to C for a 2732 EPROM. It may be jumpered A to B, effectively replacing it with a 2716 EPROM. But please note that the factory setting must be maintained for proper system operation. The optional setting reduces the address space available and is only to be used in special, limited applications.

## 6.2. Bootstrap Program

J2 (located in the lower mid-section of the board) is jumpered B to C for conditional bootstrap operation. This mode is used for the Decision I and controllers are shipped from the factory with a jumper between these two pins.

J2 is jumpered A to B for non-bootstrap mode in systems which cannot allow a temporary master to hog the bus and intend to boot the DJDMA controller by external means.

## 7. BOOTSTRAP LOAD

The DJDMA performs an automatic bootstrap load at reset or power-on if J2 is jumpered B to C and a shunt jumper is placed between pins 1 and 2 of P3, or if a terminal is connected to P3. In either case, the controller halts the main CPU by taking control of the bus and reads the first 38 (hex) locations in main memory into its own local memory. Next it loads 0s into these first 38 (hex) bytes and places a short, 19 byte (decimal) handshake routine between 000038 and 00004A (hex). The bus is then released. When the main CPU executes the first part of the handshake routine, the controller restores the first 38 (hex) locations of main memory to its original state. Next, 80 (hex) bytes are loaded between 000080 and 0000FF (hex) from the first sector on Track 0 of the disk. Finally, the controller writes a control byte to the handshake routine which causes the main CPU to branch to location 000080 (hex). A listing of the 19-byte handshake routine is given below.

Table 7-1. 19-Byte Handshake Routine

000038	21 4A 00	START:	LXI	H, 4A
00003B	36 00		MVI	M, 0
00003D	7E	LOOP:	MOV	A, M
00003E	B7		ORA	A
00003F	CA 3D 00		JZ	LOOP
000042	FE 40		CPI	40H
000044	C2 3D 00		JNZ	LOOP
000047	C3 80 00		JMP	80H
00004A	FF		DB	0FFH

The controller will boot from either the first drive connected to the 8 inch port or the first drive connected to the 5 1/4 inch port. The decision as to which port to choose is determined by testing for a "drive ready" signal. The 8 inch port is tested first. The controller will alternately continue to test for "drive ready" indefinitely to allow the user time to insert a diskette. This is evidenced by the indicator lights on the disk drives. They will alternately blink as the controller checks for the ready signal.

## 8. BOOTING THE DJDMA

The following is the proper procedure for booting the DJDMA:

1. Open the door of any drive the DJDMA could boot from.
2. Insert a bootstrap diskette in the boot drive WITHOUT closing the driver door.
3. Depress the RESET switch.
4. While the RESET switch is depressed, close the drive door.
5. Release the RESET switch.

It is possible that the above procedure will have to be repeated twice depending on the value of location 0.

If a shunt jumper across pins 2 and 3 of P3 is not in place or if a terminal is not connected to P3, the controller powers itself up in normal "cycle steal" mode and waits for commands from the system.

## 9. FORMATTING DISKETTES

There are no firmware commands on the DJDMA to format diskettes for two reasons: Formatting is a dangerous operation. If a diskette is in a drive with valuable information written on it, an accidental format command could destroy this data. The controller is also capable of formatting a wide variety of diskettes and the EPROM is not large enough to accommodate both the command processor code and all of the desirable format routines.

For these reasons, the format routines are loaded from main memory using the WRITE CONTROLLER MEMORY command and executed using the EXECUTE CONTROLLER ROUTINE command. A listing of two format programs for IBM soft-sectored 8 inch diskettes and North Star hard-sectored 5 1/4 inch diskettes appears as an appendix to this manual. These programs are also available on diskettes for a modest cost for those who wish to avoid using controller commands not supported in the field.

When a CP/M operating system is shipped with either a lone DJDMA controller or a disk system which includes a DJDMA controller, there are built-in commands on the system diskette which will format both types of diskettes.

```

0000' 31 059E' SP,ECODE+30H
0003' 21 1030' HL,1030H
0006' 22 0161' (DOTCMD+1),HL
0009' 21 113A' HL,SDADVT
000C' 22 0167' (ATCMD+1),HL
000F' 21 016F' HL,SMESSG
0012' CD 011E' CALL OUTM
0015' CD 012A' CALL INPUT
0018' D2 0024' NC,DATAOK
001B' 21 01BA' HL,BMESSG
001E' CD 011E' CALL OUTM
0021' C3 0000' JP START
0024' 32 045D' (SINGLE+1),A
0027' 21 01ED' HL,DMESSG
002A' CD 011E' CALL OUTM
002D' CD 012A' CALL INPUT
0030' DA 001B' C,DEXIT
0033' E6 01' 1
0035' 32 032A' (DENSITY),A
0038' CA 0065' Z,SIDE
003B' 21 0225' HL,LMESSG
003E' CD 011E' CALL OUTM
0041' CD 012A' CALL INPUT
0044' DA 001B' C,DEXIT
0047' FE 03' 3
0049' CA 001B' Z,DEXIT
004C' 16 00' D,0
004E' 5F 3C' E,A
0050' 32 03C5' (DLCODE-DDFMT+DOUBLE),A
0053' 21 016C' HL,STABLE
0056' 19' HL,DE
0057' 7E' A,M
0058' 32 0407' (DLAST-DDFMT+DOUBLE),A
005B' 3E 20' A,20H
005D' 87' A,A
005E' 1D' E
005F' F2 005D' P,DCNST
0062' 32 03EF' (DSIZE-DDFMT+DOUBLE),A
0065' 21 0265' HL,HMESSG
0068' CD 011E' CALL OUTM
006B' CD 012A' CALL INPUT
006E' DA 001B' C,DEXIT
0071' E6 01' AND
0073' 32 041C' (DDBIT-DDFMT+DOUBLE),A
0076' 32 0532' (SDBIT-SDFMT+SINGLE),A
0079' 21 0151' HL,LSDCMD
007C' 06 0A' B,0AH
007E' CD 00FB' LCMC
0081' 21 0160' HL,DOTCMD
0084' 06 06' B,6
0086' CD 00FB' LCMC
0089' CA 00A8' Z,PROCED
008C' 21 029A' HL,RMESSG
008F' FE 52' 82H
0091' CA 0097' Z,$+6
0094' 21 02D6' HL,WMESSG

START: LD SP,ECODE+30H
LD HL,1030H
LD (DOTCMD+1),HL
LD HL,SDADVT
LD (ATCMD+1),HL
LD HL,SMESSG
CALL OUTM
CALL INPUT
NC,DATAOK
HL,BMESSG
CALL OUTM
JP START
(SINGLE+1),A
HL,DMESSG
CALL OUTM
CALL INPUT
C,DEXIT
1
(DENSITY),A
Z,SIDE
HL,LMESSG
CALL OUTM
CALL INPUT
C,DEXIT
3
Z,DEXIT
D,0
E,A
A
(DLCODE-DDFMT+DOUBLE),A
HL,STABLE
HL,DE
A,M
(DLAST-DDFMT+DOUBLE),A
A,20H
A,A
E
P,DCNST
(DSIZE-DDFMT+DOUBLE),A
HL,HMESSG
CALL OUTM
CALL INPUT
C,DEXIT
1
(DDBIT-DDFMT+DOUBLE),A
(SDBIT-SDFMT+SINGLE),A
HL,LSDCMD
B,0AH
LCMC
HL,DOTCMD
B,6
LCMC
Z,PROCED
HL,RMESSG
82H
Z,$+6
HL,WMESSG

;initialize the stack pointer
;initialize command address

;start of program message
;send the message
;get response to drive number
;test for valid input
;invalid input message
;send the message
;go back to start of program
;store the drive number in code
;type of density message
;send the message
;wait for response
;test for improper input
;density encoded in bit 0
;save for later use
;skip sector size if single density
;sector length message
;send the message
;wait for input
;test for improper input
;further test for improper input
;error exit
;form offset into sector table

;adjust for sector length code
;store in format code

;fetch number of sectors
;store in format code
;sector length code is 80,100, or 0

;decrement the sector type
;test for cycle done
;store 1/4 length in format code
;double sided media message
;send the message
;wait for input
;test for improper input
;discard all but bit 0
;store in format code double density
;store in format code single density
;load single density code command
;command length
;load the code
;format track 0 command
;command length
;execute the command
;zero => no error
;drive not ready message
;drive not ready error code
;test for drive not ready
;drive must be write protected

```

```

0097. CD 011E.
009A. CD 012A.
009D. DA 001B.
00A0. E6 01
00A2. CA 0000.
00A5. C3 0079.
00A8. 21 0327.
00AB. CD 011E.
00AE. 21 1050
00B1. 3A 032A.
00B4. B7
00B5. CA 00C9.
00B8. 21 0147.
00BB. 06 0A
00BD. CD 00FB.
00C0. 21 1159
00C3. 22 0167.
00C6. 21 1030
00C9. 22 0161.
00CC. 3E 2A
00CE. CD 0114.
00D1. 21 0166.
00D4. 06 06
00D6. CD 00FB.
00D9. FE 4D
00DB. C2 00E7.
00DE. 21 0312.
00E1. CD 011E.
00E4. C3 0000.
00E7. 21 0160.
00EA. 06 06
00EC. CD 00FB.
00EF. CA 00CC.
00F2. 21 029A.
00F5. CD 011E.
00F8. C3 00DE.

00FB. 11 0050
00FE. 7E
00FF. 12
0100. 23
0101. 13
0102. 05
0103. C2 00FE.

0106. D3 EF
0108. 1B
0109. 1A
010A. B7
010B. CA 0109.
010E. 3A 0053
0111. FE 40
0113. C9

0114. 21 015C.
0117. 06 05
0119. 77
011A. 2B
011B. C3 00FB.

CALL
CALL
JP
AND
1
Z,START
LOADC
HL,CRLF
OUTM
HL,SDRDY
A,(DENSITY)
A
Z,CONTUE
HL,LDDCMD
B,0AH
LCMD
HL,DDADV
(ATCMD+1),HL
HL,1030H
(DOTCMD+1),HL
A,"*"
OUTPUT
HL,ATCMD
B,6
LCMD
CP
4DH
NZ,FMTRCK
HL,FMESG
OUTM
START
HL,DOTCMD
B,6
LCMD
Z,CONTUE+3
HL,FMESG
OUTM
ENDFMT
JP

DE,50H
A,M
(DE),A
HL
DE
B
NZ,LCMD+3
JP

(0EFH),A
DEC
LD
A,(DE)
OR
A
Z,ECHD+3
A,(53H)
40H
RET

LD
LD
LD
INC
INC
DEC
JP

OUT
DEC
LD
OR
JP
LD
CP
RET

LD
LD
LD
LD
DEC
JP

HL,SOCHD+1
B,5
M,A
HL
LCMD

```

```

;send the message
;wait for input
;test for improper input
;discard all but bit 0
;zero => start the program over
;carriage return and line feed
;output the string
;adjusted execution address of format

;test for double density
;make no adjustments for single density
;load double density format command
;command length
;load the code into controller
;advance track execute address
;update the command execute address
;format execute address
;update track format execute address
;send a star for a track done

;advance track command
;command length
;load the command and execute
;last track value (77 decimal)
;zero => formatting done
;send final message

;go format another disk
;format a track command
;command length
;load and execute the command
;loop back for more tracks
;drive has become not ready
;stop the formatting

;start of command sequence
;get command data
;load into command area
;advance the pointers

;test for transfer done

;start the controller
;pointer for status byte of halt cmd

;test for command string done

;status byte for execute command
;test for no error

;data byte of serial output command
;serial output command string length
;store the data
;back up to pointer
;load the command and execute

```

PROCEED: LD

CONTUE: LD

ENDFMT: LD

FMTRCK: LD

LCMD:

ECMD:

OUTPUT: LD



```

011E' 7E
011F' B7
0120' C8
0121' E5
0122' CD 0114'
0125' E1
0126' 23
0127' C3 011E'

012A' 21 003F
012D' 3E 40
012F' 96
0130' C2 012D'
0133' 77
0134' 2B
0135' 7E
0136' F5
0137' CD 0114'
013A' F1
013B' E6 7F
013D' FE 30
013F' DB
0140' FE 34
0142' 3F
0143' D8
0144' E6 03
0146' C9

;get current byte of message
;test for end of message
;return at end of message
;save the character pointer
;output the character
;recover the character pointer
;advance the character pointer
;go get the next character

;serial input status byte
;test value for status
;test for character ready
;zero => new character ready
;back up pointer to the character
;pickup the character
;save the data
;echo the data

;turn it into ASCII
;test for smaller than zero
;test for larger than three

;change ASCII to binary

```

```

LD A,M
OR A
RET Z
PUSH HL
CALL OUTPUT
POP HL
INC HL
JP OUTM

LD HL,3FH
LD A,40H
M
SUB NZ, INPUT+3
LD M,A
DEC HL
LD A,M
PUSH AF
CALL OUTPUT
POP AF
AND 7FH
CP 30H
RET C
CP 34H
CCF
RET AND
AND
RET AND
PAGE

```

```

OUTM:
INPUT:

```

0147	A1	LDDCMD: DB	0A1H	;write controller memory command
0148	032B	DW	DOUBLE	;main memory address pointer
014A	00	DB	0	
014B	0131	DW	SINGLE-DOUBLE	;byte count
014D	1030	DW	1030H	;controller memory address pointer
014F	25	DB	25H	;controller halt command
0150	00	DB	0	;halt command status byte
0151	A1	LSDCMD: DB	0A1H	
0152	045C	DW	SINGLE	
0154	00	DB	0	
0155	0112	DW	ECODE-SINGLE	
0157	1030	DW	1030H	
0159	25	DB	25H	
015A	00	DB	0	
015B	2B	SOCMD: DB	2BH	;output character to controller cmd
015C	00	DB	0	;output data
015D	00	DB	0	;output character command status
015E	25	DB	25H	;controller halt command
015F	00	DB	0	;halt command status byte
0160	A2	DOTCMD: DB	0A2H	;execute controller routine command
0161	1030	DW	1030H	;format a track address
0163	00	DB	0	;execute command status
0164	25	DB	25H	;halt command
0165	00	DB	0	;status byte
0166	A2	ATCMD: DB	0A2H	;advance the track value address
0167	113A	DW	SDADVTT	
0169	00	DB	0	
016A	25	DB	25H	
016B	00	DB	0	
016C	1B	STABLE: DB	1BH	;26 sectors per track (256 bytes)
016D	10	DB	10H	;15 sectors per track (512 bytes)
016E	09	DB	9	;8 sectors per track (1024 bytes)

PAGE

```

016F' 0D0A
0171' 49 42 4D 20
0175' 43 6F 6D 70
0179' 61 74 61 62
017D' 6C 65 20 38
0181' 20 69 6E 63
0185' 68 20 46 6F
0189' 72 6D 61 74
018D' 20 50 72 6F
0191' 67 72 61 6D
0195' 0D0A
0197' 53 65 6C 65
019B' 63 74 20 61
019F' 20 44 72 69
01A3' 76 65 20 28
01A7' 20 30 2C 20
01AB' 31 2C 20 32
01AF' 2C 20 6F 72
01B3' 20 33 20 29
01B7' 3A 20
01B9' 00
01BA' 0D0A
01BC' 49 6D 70 72
01C0' 6F 70 65 72
01C4' 20 69 6E 70
01C8' 75 74 20 2D
01CC' 20 72 65 74
01D0' 75 72 6E 69
01D4' 6E 67 20 74
01D8' 6F 20 73 74
01DC' 61 72 74 20
01E0' 6F 66 20 70
01E4' 72 6F 67 72
01E8' 61 6D
01EA' 0D0A
01EC' 00
01ED' 0D0A
01EF' 53 65 6C 65
01F3' 63 74 20 64
01F7' 6F 75 62 6C
01FB' 65 20 64 65
01FF' 6E 73 69 74
0203' 79 20 28 20
0207' 31 20 29 20
020B' 6F 72 20 73
020F' 69 6E 67 6C
0213' 65 20 64 65
0217' 6E 73 69 74
021B' 79 20 28 20
021F' 30 20 29 3A
0223' 20
0224' 00
0225' 0D0A
0227' 53 65 6C 65
022B' 63 74 20 74
022F' 68 65 20 62
0233' 79 74 65 20

```

```

SMESG: DW
DB
CRLFS
"IBM Compatible 8 inch Format Program"

```

```

DW
DB
CRLFS
"Select a Drive ( 0, 1, 2, or 3 ) : "

```

```

DB
DW
BMESG: DB
CRLFS
"Improper input - returning to start of program"

```

```

DW
DB
DMESG: DW
DB
CRLFS
"Select double density ( 1 ) or single density ( 0 ) : "

```

```

DB
DW
LMESG: DB
CRLFS
"Select the byte length of a sector ( 0=256, 1=512, 2=1024 ) : "

```

```

0237. 6C 65 6E 67
023B. 74 68 20 6F
023F. 66 20 61 20
0243. 73 65 63 74
0247. 6F 72 20 28
024B. 20 30 3D 32
024F. 35 36 2C 20
0253. 31 3D 35 31
0257. 32 2C 20 32
025B. 3D 31 30 32
025F. 34 20 29 3A
0263. 20
0264. 00
0265. 0D0A
0267. 53 65 6C 65
026B. 63 74 20 73
026F. 69 6E 67 6C
0273. 65 20 28 20
0277. 30 20 29 20
027B. 6F 72 20 64
027F. 6F 75 62 6C
0283. 65 20 28 20
0287. 31 20 29 20
028B. 73 69 64 65
028F. 64 20 6D 65
0293. 64 69 61 20
0297. 3A 20
0299. 00
029A. 0D0A
029C. 44 72 69 76
02A0. 65 20 6E 6F
02A4. 74 20 72 65
02A8. 61 64 79 20
02AC. 2D 20 72 65
02B0. 73 74 61 72
02B4. 74 20 70 72
02B8. 6F 67 72 61
02BC. 6D 3F 20 28
02C0. 20 30 20 29
02C4. 20 6F 72 20
02C8. 63 79 63 6C
02CC. 65 20 28 20
02D0. 31 20 29 3A
02D4. 20
02D5. 00
02D6. 0D0A
02D8. 57 72 69 74
02DC. 65 20 70 72
02E0. 6F 74 65 63
02E4. 74 65 64 20
02E8. 2D 20 72 65
02EC. 73 74 61 72
02F0. 74 20 70 72
02F4. 6F 67 72 61
02F8. 6D 3F 20 28
02FC. 20 30 20 29
0300. 20 6F 72 20
0304. 63 79 63 6C
0308. 65 20 28 20

```

```

DB
HMESG: DW
DB

```

```

0
CRLFS
"Select single ( 0 ) or double ( 1 ) sided media : "

```

```

DB
RMESG: DW
DB

```

```

0
CRLFS
"Drive not ready - restart program? ( 0 ) or cycle ( 1 ) : "

```

```

DB
WMESG: DW
DB

```

```

0
CRLFS
"Write protected - restart program? ( 0 ) or cycle ( 1 ) : "

```

030C' 31 20 29 3A  
 0310' 20  
 0311' 00  
 0312' 000A  
 0314' 46 6F 72 6D  
 0318' 61 74 74 69  
 031C' 6E 67 20 66  
 0320' 69 6E 69 73  
 0324' 68 65 64  
 0327' 000A  
 0329' 00  
 032A' 00

FMESG: DW DB  
 DB DB

0 CRLF  
 "Formatting finished"

CRLF: DW CRLF  
 DB DB  
 DENSITY: DB DB  
 PAGE

032B\*

```

1030 21 4003 EQU .PHASE $
1033 CB 7E LD HL,STATUS
1035 3E 82 BIT 7,M
1037 C8 LD A,82H
1038 CB 76 RET Z
103A 3E 90 BIT 6,M
103C C0 LD A,90H
103D DD 36 0B 00 NZ
1041 3A 10C4 LD (IX+0BH),0
1044 FD BE 01 LD A,(DTRCK)
1047 F5 CP (IY+1)
1048 C4 00A3 PUSH AF
104B 21 4001 CALL HL,DISK
104E 11 4007 LD DE,CONTRL
1051 F1 POP AF
1052 FE 2B CP 2BH
1054 3E 04 LD A,4
1056 3B 02 JR C,LOADPC
1058 3E 14 LD A,14H
105A 32 1081 LD (PRECMP),A
105D 9F A,A
105E F6 FE OR 0FEH
1060 FD A6 02 AND (IY+2)
1063 F6 02 OR 2
1065 FD 77 02 LD (IY+2),A
1068 F6 0C OR 0CH
106A 32 4005 LD (4005H),A
106D 06 50 LD B,50H
106F 3A 4003 LD A,(STATUS)
1072 E6 10 AND INDEX
1074 20 F9 JR NZ,DDLBL1
1076 3A 4003 LD A,(STATUS)
1079 E6 10 AND INDEX
107B 28 F9 JR Z,DDLBL2
107D 3E 90 LD A,90H
107F 12 LD (DE),A
1080 3E 00 LD A,0
1081 32 4006 EQU S-1
1082 36 4E LD (4006H),A
1085 10 FC LD M,4EH
1087 06 0C DJNZ DDLBL3
108B 36 00 LD B,0CH
108D 10 FC DJNZ M,0
108F 3E 80 LD DDLBL4
1091 12 LD A,80H
1092 36 52 LD (DE),A
1094 36 24 LD M,52H
1096 36 52 LD M,24H
1098 36 24 LD M,24H
109A 36 52 LD M,52H
109E 12 LD A,90H
109F 36 24 LD (DE),A
10A1 36 FC LD M,0FCH

```

```

;check that the drive is ready
;drive not ready error code
;error exit
;test for write protected
;write protected error code
;error exit
;reset index counter
;get the new track value
;compare with current track
;save the track
;move the head(s) if needed
;pointer to disk shift register
;pointer to control port
;recover the tack
;compare with track 43
;no write precompensation
;carry > track is less than 43
;write precompensation bit set
;setup the write precompensation byte
;push carry bit throughout accumulator
;merge with drive pattern
;select side 0
;restore drive pattern
;turn off step command
;update the drive register
;preamble length

;look for index pulse
;wait for no index pulse present

;wait for leading edge of new indes pulse
;control byte - normal write/no CRC
;initialize control port

;write precompensation & controller start
;start the controller

;write the preamble
;zero preamble length

;write the zero preamble
;control byte for 16 bit write
;change mode
;first half of C2
;second half of C2
;another C2

;the third C2
;control byte 8 bit write
;change mode
;finish the sync bytes
;index mark

```

```

10A3 06 0C
10A5 36 4E
10A7 10 FC

10A9 06 0C
10AB 36 00
10AD 10 FC
10AF 3E 81
10B1 12
10B2 36 44
10B4 36 89
10B6 36 44
10B8 36 89
10BA 36 44
10BC 3E 91
10BE 12
10BF 36 89
10C1 36 FE
10C3 36 00
10C4
10C5 36 00
10C6
10C7 36 01
10C8
10C9 36 01
10CA
10CB 3E A1
10CD 12
10CE 77
10CF 77
10D0 3E 90
10D2 12
10D3 06 16
10D5 36 4E
10D7 10 FC
10D9 06 0C
10DB 36 00
10DD 10 FC
10DF 3E 81
10E1 12
10E2 36 44
10E4 36 89
10E6 36 44
10E8 36 89
10EA 36 44
10EC 3E 91
10EE 12
10EF 36 89
10F1 36 FB
10F3 06 40
10F4
10F5 36 E5
10F7 36 E5
10F9 36 E5
10FB 36 E5
10FD 10 F6
10FF 3E A1
1101 12
1102 77

DDLBL5: LD B,32H
          LD M,4EH
          DJNZ DDLBL5

          LD B,0CH
          LD M,0
          DJNZ DDLBL6
          LD A,81H
          LD (DE),A
          LD M,44H
          LD M,89H
          LD M,44H
          LD M,89H
          LD M,44H
          LD A,91H
          LD (DE),A
          LD M,89H
          LD M,0FEH
          LD M,0
          EQU $-1
          LD M,0
          EQU $-1
          LD M,1
          EQU $-1
          LD M,1
          EQU $-1
          LD A,0A1H
          LD (DE),A
          LD M,A
          LD M,A
          LD A,90H
          LD (DE),A
          LD B,16H
          LD M,4EH
          DJNZ DDLBL7
          LD B,0CH
          LD M,0
          DJNZ DDLBL8
          LD A,81H
          LD (DE),A
          LD M,44H
          LD M,89H
          LD M,44H
          LD M,89H
          LD M,44H
          LD A,91H
          LD (DE),A
          LD M,89H
          LD M,0FBH
          LD B,40H
          EQU $-1
          LD M,0ESH
          LD M,0ESH
          LD M,0ESH
          LD M,0ESH
          DJNZ DDLBL9
          LD A,0A1H
          LD (DE),A
          LD M,A
          LD M,A

          ;postamble length
          ;write the postamble
          ;zero preamble length
          ;write the preamble
          ;16 bit write mode w/CRC
          ;change mode
          ;first half of A1
          ;second half of A1
          ;third A1
          ;8 bit write mode w/CRC
          ;change mode
          ;finish sync bytes
          ;sector header ID byte
          ;write the track number
          ;write the side
          ;write the sector number
          ;sector length code
          ;mode to write CRC bytes
          ;change mode
          ;write the CRC bytes
          ;reset CRC generator
          ;change mode
          ;4E postamble length
          ;write the postamble
          ;data field preamble
          ;write the preamble
          ;16 bit write w/CRC
          ;change mode
          ;first half of A1
          ;second half of A1
          ;third A1
          ;8 bit write w/CRC
          ;change mode
          ;finish the 3 sync bytes
          ;data header ID byte
          ;sector length divided by four
          ;empty sector data byte
          ;write four fill bytes
          ;test for data field write done
          ;CRC control byte
          ;change mode
          ;write the CRC bytes
    
```

```

1103 77 M,A
1104 3E 90 LD
1106 12 A,90H LD
1107 3A 10C8 LD (DE),A
1108 3C A,(DSECT) LD
1109 FE 1B INC
110C CP 1BH
DLAST EQU $-1
110D 36 4E LD M,4EH
110F 20 02 JR NZ,$+4
1111 3E 01 LD A,1
1113 32 10C8 LD (DSECT),A
1116 06 35 LD B,35H
1118 36 4E LD M,4EH
111A 10 FC DJNZ
111C 20 0B JR NZ,DMLOOP
111E 36 4E LD M,4EH
1120 06 00 LD B,0
1121 LD $-1
1122 3A 10C6 LD A,(DSIDE)
1125 A8 XOR B
1126 32 10C6 LD (DSIDE),A
1129 36 4E LD M,4EH
112B 06 4F LD B,4FH
112D 08 EX AF,AF'
112E 36 4E LD M,4EH
1130 3A 4003 LD A,(STATUS)
1133 E6 10 AND INDEX
1135 28 F7 JR Z,DLBLB
1137 08 EX AF,AF'
1138 28 0F JR Z,DDLBC
113A FD 7E 02 LD A,(IY+2)
113D F6 0C OR 0CH
113F E6 FD AND 0FDH
1141 32 4005 LD (4005H),A
1144 36 4E LD M,4EH
1146 C3 1085 JIP DDLBL3
1149 36 4E LD M,4EH
114B 36 4E LD M,4EH
114D 36 4E LD M,4EH
114F AF XOR A
1150 12 LD (DE),A
1151 3E 06 LD A,6
1153 32 4006 LD (4006H),A
1156 3E 40 LD A,40H
1158 C9 RET
DDADVT: LD A,(DTRCK)
1159 3A 10C4 INC A
115C 3C 10C4 LD (DTRCK),A
115D 32 10C4 RET
1160 C9 .DEPHASE
PAGE

```

```

;turn off the CRC generator
;change mode
;get the sector number
;test for last sector +1
;first byte of postamble
;zero => all sectors written
;update the sector number
;postamble length less one
;write the postamble
;first fill byte
;double sided bit test
;conditionally switch the side byte
;update the side byte
;second fill byte
;preamble length less one
;save the double sided status
;write a fill byte
;wait for the index pulse
;recover the double sided status
;zero => track write is done
;drive pattern
;turn off the step command
;change read/write heads
;update the command register
;first preamble byte
;format the other side
;trailing fill byte
;trailing fill byte
;trailing fill byte
;turn off the write gate
;turn off the controller
;status code
;get the current track value
;increment
;restore the new value
;return with current track value

```



```

045C'
1030 3E 00 S
1032 CD 00A6 LD A,0
1035 C0 SDRIVE
1036 FD 7E 02 NZ
1039 F6 0F OR A,(IX+2)
103B 32 4005 LD 0FH
103E 21 0000 LD (4005H),A
1041 2B HL,0
1042 7C HL
1043 B5 A,H
1044 20 FB OR L
1046 DD 77 0B JR NZ,SDWAIT
1049 CD 00A0 LD (IX+0BH),A
104E 28 0E CB 6E HOME
1050 21 4003 JR 5,M
1053 CB 7E LD Z,SNREXT
1055 3E 82 BIT HL,STATUS
1057 C8 REF 7,M
1058 CB 76 BIT A,82H
105A 3E 90 BIT Z
105C C0 LD 6,M
105D DD 36 0B 00 LD A,90H
1061 3A 10B6 LD (IX+0BH),0
1064 FD BE 01 CP A,(STRCK)
1067 C4 00A3 CALL (IX+1)
106A 21 4001 LD NZ,SEEK
106D 11 4007 LD HL,DISKD
1070 06 28 LD DE,CONTRL
1072 3A 4003 LD B,28H
1075 E6 10 LD A,(STATUS)
1077 20 F9 AND INDEX
1079 3A 4003 JR NZ,SDLBL1
107C E6 10 AND A,(STATUS)
107E 28 F9 INDEX
1080 3E 90 JR Z,SDLBL2
1082 12 LD A,90H
1083 3E 44 LD (DE),A
1085 32 4006 LD A,44H
1088 36 FF LD (4006H),A
108A 10 FC M,0FFH
108C 3E 80 SDLBL3
108E 12 LD A,80H
108F 06 0C LD (DE),A
1091 36 AA LD B,0CH
1093 10 FC M,0AAH
1095 36 F7 SDLBL4
1097 3E 90 LD M,0F7H
1099 12 LD A,90H
109A 36 7A LD (DE),A
109C 06 1A LD M,7AH
109E 36 FF LD B,1AH
10A0 10 FC M,0FFH
10A2 3E 80 SDLBL5

SINGLE EQU .PHASE
SDFMT: LD A,0
CALL SDRIVE
RET NZ
LD A,(IX+2)
OR 0FH
LD (4005H),A
HL,0
SDWAIT: DEC HL
LD A,H
OR L
JR NZ,SDWAIT
LD (IX+0BH),A
HOME
5,M
Z,SNREXT
HL,STATUS
7,M
A,82H
Z
6,M
A,90H
NZ
(IX+0BH),0
A,(STRCK)
(IX+1)
NZ,SEEK
HL,DISKD
DE,CONTRL
B,28H
A,(STATUS)
INDEX
NZ,SDLBL1
A,(STATUS)
INDEX
Z,SDLBL2
A,90H
(DE),A
A,44H
(4006H),A
M,0FFH
SDLBL3
A,80H
(DE),A
B,0CH
M,0AAH
SDLBL4
M,0F7H
A,90H
(DE),A
M,7AH
B,1AH
M,0FFH
SDLBL5
DUNZ

SMLOOP: LD A,80H
;second byte filled with proper drive number
;select the new drive
;return if wrong value
;get the drive pattern
;side 0 and no step command
;update drive control register
;delay for the head load

;reset the index counter
;calibrate the head(s)
;test for track zero

;test for the drive ready
;drive not ready code
;error exit
;write protect bit
;write protect error code

;reset the index counter
;get the new track
;compare with current track
;do track seek if necessary
;controller data register
;control register
;preamble length

;wait for no index pulse

;wait for leading edge of new index pulse
;clear the CRC register & turn on write gate
;change modes
;single density & start bit
;start the controller

;write the preamble
;16 bit write mode
;change modes
;zero preamble length
;half a zero cell
;write the zero preamble
;first half of FC
;8 bit write mode
;change modes
;second half of FC
;postamble length
;write the postamble
;16 bit write mode

```

```

10A4 12
10A5 06 0C
10A7 36 AA
10A9 10 FC
10AB 3E 81
10AD 12
10AE 36 F5
10B0 3E 91
10B2 12
10B3 36 7E
10B5 36 00
10B6
10B7 36 00
10B8
10B9 36 01
10BA
10BB 36 00
10BD 3E A1
10BF 12
10C0 77
10C1 77
10C2 3E 90
10C4 12
10C5 06 0B
10C7 36 FF
10C9 10 FC
10CB 3E 80
10CD 12
10CE 06 0C
10D0 36 AA
10D2 10 FC
10D4 3E 81
10D7 36 F5
10D9 3E 91
10DB 12
10DC 36 6F
10DE 06 80
10E0 36 E5
10E2 10 FC
10E4 3E A1
10E6 12
10E7 77
10E8 77
10E9 3E 90
10EB 12
10EC 3A 10BA
10EF 3C
10F0 FE 1B
10F2 36 FF
10F4 20 02
10F6 3E 01
10F8 32 10BA
10FB 06 1A
10FD 36 FF
10FF 10 FC
1101 20 9F
1103 36 FF
1105 06 00

      (DE),A
      B,0CH
      M,0AAH
SDLBL6: LD
      A,81H
      (DE),A
      M,0F5H
      A,91H
      (DE),A
      M,7EH
      M,0
      M,0
      M,0
      M,1
      M,1
      M,0
      A,0A1H
      (DE),A
      M,A
      M,A
      A,90H
      (DE),A
      B,0BH
      M,0FFH
SDLBL7: LD
      A,80H
      (DE),A
      B,0CH
      M,0AAH
      A,81H
      (DE),A
      M,6FH
      B,80H
      M,0E5H
SDLBL9: LD
      A,0A1H
      (DE),A
      M,A
      M,A
      A,90H
      (DE),A
      A,(SSECT)
      A
      1BH
      M,0FFH
      NZ,$+4
      A,1
      (SSECT),A
      B,1AH
      M,0FFH
      NZ,SMLOOP
      M,0FFH
      B,0

STRCK EQU
SSIDE EQU
SSECT EQU

SDLBL7: LD
SDLBL8: LD
SDLBL9: LD

      ;change modes
      ;sector header preamble length
      ;half a zero cell
      ;write the preamble
      ;enable CRC & 16 bit write
      ;change modes
      ;first half of FE
      ;enable CRC & 8 bit write
      ;change modes
      ;second half of FE
      ;write the track
      ;write the side byte
      ;write the sector number
      ;write the sector length code
      ;change modes
      ;write the CRC bytes
      ;reset the CRC
      ;change modes
      ;sector header postamble length
      ;write the postamble
      ;16 bit write mode
      ;change modes
      ;data field preamble length
      ;write the preamble
      ;enable CRC & 16 bit write
      ;change modes
      ;first half of FB
      ;8 bit write
      ;change modes
      ;second half of FB
      ;sector data field length
      ;write the data field
      ;change modes
      ;write the CRC bytes
      ;reset the CRC
      ;change modes
      ;get the current sector
      ;advance
      ;compare with 27
      ;first postamble byte
      ;zero => all sectors written
      ;update the sector
      ;postamble length less one
      ;write the postamble
      ;test for more sectors to format
      ;first fill byte
      ;side bit

```

```

1106      3A 10B8
1107      A8
1108      32 10B9
1109      36 FF
1110      06 19
1111      08
1112      36 FF
1113      3A 4003
1114      E6 10
1115      28 F7
1116      08
1117      28 0F
1118      FD 7E 02
1119      F6 0C
1120      E6 FD
1121      32 4005
1122      36 FF
1123      C3 1088
1124      36 FF
1125      AF
1126      12
1127      3E 06
1128      32 4006
1129      3E 40
1130      C9
1131      3A 1086
1132      3C
1133      32 10B6
1134      C9
1135      0562*
1136      EQU $
1137      END

SDSBIT EQU $-1
LD      A,(SSIDE)
XOR     B
LD      (SSIDE),A
LD      M,0FFH
LD      B,19H
EX      AF,AF'
LD      M,0FFH
LD      A,(STATUS)
AND     INDEX
JR      Z,SDLBLB
EX      AF,AF'
JR      Z,SDLBLC
LD      A,(IY+2)
OR      0CH
AND     0FDH
LD      (4005H),A
LD      M,0FFH
JP      SDLBL3
LD      M,0FFH
XOR     A
LD      (DE),A
LD      A,6
LD      (4006H),A
LD      A,40H
RFT
SDADVT: LD      A,(STRCK)
INC     A
LD      (STRCK),A
RET
.DEPHASE
EQU     $
END
;get the current side
;conditionally switch side bits
;update the side byte
;write second fill byte
;preamble length less one
;save the double sided status
;write a fill byte

;wait for the index hole
;recover the double sided status
;zero => single sided
;get the drive pattern
;turn off the step command
;turn on head one
;update drive control register
;write first preamble byte
;go format the other side
;trailing byte

;turn off write gate

;turn off the controller
;status code

;get the current track
;advance track value
;update the track value
;return with track value
    
```



0084.	83	ADD	A,E
0085.	5F	LD	E,A
0086.	D5	DE	
0087.	21 0202.	PUSH	
008A.	CD 013E.	LD	HL, RMESG
008D.	CD 014A.	CALL	OUTM
0090.	D1	CALL	INPUT
0091.	DA 0021.	POP	DE
0094.	E6 01	JP	C, DEXIT
0096.	CA 00AE.	AND	1
0099.	7B	JP	Z, LOADC
009A.	E6 80	LD	A,E
009C.	3E 10	AND	00H
009E.	CA 00A6.	LD	A, 10H
00A1.	21 0105.	JP	Z, STORED
00A4.	19	LD	HL, TYPE-00H
00A5.	7E	ADD	HL, DE
00A6.	32 0406.	LD	A,M
00A9.	3E E5	LD	(CPDATA-NSFMT+FORMAT), A
00AB.	32 0404.	LD	A, 0E5H
00AE.	21 0167.	LD	(DATA-NSFMT+FORMAT), A
00B1.	06 0A	LD	HL, LFDCMD
00B3.	CD 011B.	LD	B, 0AH
00B6.	21 0176.	LD	LCMD
00B9.	06 06	LD	HL, DOTCMD
00BB.	CD 011B.	LD	B, 6
00BE.	CA 00DD.	CALL	LCMD
00C1.	21 02F4.	JP	Z, PROCED
00C4.	FE 82	LD	HL, RMESG
00C6.	CA 00CC.	CP	02H
00C9.	21 0330.	JP	Z, \$+6
00CC.	CD 013E.	LD	HL, WMESG
00CF.	CD 014A.	CALL	OUTM
00D2.	DA 0021.	CALL	INPUT
00D5.	E6 01	JP	C, DEXIT
00D7.	CA 0000.	AND	1
00DA.	C3 00AE.	JP	Z, START
00DD.	21 0381.	JP	LOADC
00E0.	CD 013E.	LD	HL, CRLF
00E3.	21 104F.	LD	OUTM
00E6.	22 0177.	LD	HL, ENTRY
00E9.	3E 2A	LD	(DOTCMD+1), HL
00EB.	CD 0134.	LD	A, ++
00EE.	21 017C.	LD	OUTPUT
00F1.	06 06	LD	HL, ATCMD
00F3.	CD 011B.	LD	B, 6
00F6.	47	CALL	LCMD
00F7.	3A 03DE.	LD	B,A
00FA.	B8	LD	A, (STRACK-NSFMT+FORMAT)
00FB.	C2 0107.	CP	B
00FE.	21 036C.	JP	NZ, FMTRCK
0101.	CD 013E.	LD	HL, FMESG
0104.	C3 0000.	CALL	OUTM
0107.	21 0176.	JP	START
010A.	06 06	LD	HL, DOTCMD
010C.	CD 011B.	LD	B, 6
010F.	CA 00E9.	CALL	LCMD
0112.	21 02F4.	JP	Z, CONTUE
0115.	CD 013E.	LD	HL, RMESG

DATA:

STORED:

LOADC:

PROCED:

CONTUE:

ENDFMT:

FMTRCK:

0118'	C3 00FE'	JP	ENDFMT
011B'	11 0050	LD	DE, 50H
011E'	7E	LD	A, M
011F'	12	LD	(DE), A
0120'	23	INC	HL
0121'	13	INC	DE
0122'	05	DEC	B
0123'	C2 011E'	JP	NZ, LCMD+3
0126'	D3 EF	OUT	{0EFH}, A
0128'	1B	DEC	DE
0129'	1A	LD	A, (DE)
012A'	B7	OR	A
012B'	CA 0129'	JP	Z, ECMD+3
012E'	3A 0053	LD	A, (53H)
0131'	FE 40	CP	40H
0133'	C9	RET	
0134'	21 0172'	LD	HL, SOCMD+1
0137'	06 05	LD	B, 5
0139'	77	LD	M, A
013A'	2B	DEC	HL
013B'	C3 011B'	JP	LCMD
013E'	7E	LD	A, M
013F'	B7	OR	A
0140'	C8	RET	Z
0141'	E5	PUSH	HL
0142'	CD 0134'	CALL	OUTPUT
0145'	E1	POP	HL
0146'	23	INC	HL
0147'	C3 013E'	JP	OUTM
014A'	21 003F	LD	HL, 3FH
014D'	3E 40	LD	A, 40H
014F'	96	SUB	M
0150'	C2 014D'	JP	NZ, INPUT+3
0153'	77	LD	M, A
0154'	2B	DEC	HL
0155'	7E	LD	A, M
0156'	F5	PUSH	AF
0157'	CD 0134'	CALL	OUTPUT
015A'	F1	POP	AF
015B'	E6 7F	AND	7FH
015D'	FE 30	CP	30H
015F'	D8	RET	C
0160'	FE 34	CP	34H
0162'	3F	CCF	
0163'	D8	RET	C
0164'	E6 03	AND	3
0166'	C9	RET	

0167:	A1	LFDCMD:	DB	0A1H
0168:	0384	DW	DW	FORMAT
016A:	00	DB	DB	0
016B:	00EE	DW	DW	ECODE-FORMAT
016D:	1030	DW	DW	1030H
016F:	25	DB	DB	25H
0170:	00	DB	DB	0
0171:	2B	SOCMD:	DB	2BH
0172:	00	DB	DB	0
0173:	00	DB	DB	0
0174:	25	DB	DB	25H
0175:	00	DB	DB	0
0176:	A2	DOTCMD:	DB	0A2H
0177:	1030	DW	DW	1030H
0179:	00	DB	DB	0
017A:	25	DB	DB	25H
017B:	00	DB	DB	0
017C:	A2	ATCMD:	DB	0A2H
017D:	1114	DW	DW	ADVTRK
017F:	00	DB	DB	0
0180:	25	DB	DB	25H
0181:	00	DB	DB	0
0182:	23	STABLE:	DB	35
0183:	28	DB	DB	40
0184:	50	DB	DB	80
0185:	90	TYPE:	DB	90H
0186:	A0	DB	DB	0A0H
0187:	C0	DB	DB	0C0H
0188:	00	DB	DB	0
0189:	F0	DB	DB	0F0H
018A:	D0	DB	DB	0D0H
018B:	E0	DB	DB	0E0H

```

018C' 0D0A 4E 6F 72 74
018E' 68 20 53 74
0192' 61 72 20 43
0196' 6F 6D 70 61
019A' 74 61 62 6C
019E' 65 20 35 20
01A2' 31 2F 34 20
01A6' 69 6E 63 68
01AA' 20 46 6F 72
01AE' 6D 61 74 20
01B2' 50 72 6F 67
01B6' 72 61 6D
01BA' 0D0A
01BD' 53 65 6C 65
01BF' 63 74 20 61
01C3' 20 44 72 69
01C7' 76 65 20 28
01CB' 20 30 2C 20
01CF' 31 2C 20 32
01D3' 2C 20 6F 72
01D7' 20 33 20 29
01DB' 3A 20
01DF' 00
01E1' 00
01E5' 0D0A
01E9' 49 6D 70 72
01ED' 6F 70 65 72
01F1' 20 69 6E 70
01F5' 75 74 20 2D
01F9' 20 72 65 74
01FD' 75 72 6E 69
01F1' 6E 67 20 74
0205' 6F 20 73 74
0209' 61 72 74 20
020D' 6F 66 20 70
0211' 72 6F 67 72
0215' 61 6D
0219' 0D0A
0223' 00
0227' 53 65 6C 65
022B' 63 74 20 64
022F' 6F 75 62 6C
0233' 65 20 64 65
0237' 6E 73 69 74
023B' 79 20 28 20
023F' 31 20 29 20
0243' 6F 72 20 73
0247' 69 6E 67 6C
024B' 65 20 64 65
024F' 6E 73 69 74
0253' 79 20 28 20
0257' 30 20 29 3A
025B' 20
025F' 00
0263' 0D0A
0267' 53 65 6C 65

```

```

SMESGG: DW
DB
CRLFS
"North Star Compatable 5 1/4 inch Format Program"

```

```

DW
DB
CRLFS
"Select a Drive ( 0, 1, 2, or 3 ) : "

```

```

BMESGG: DW
DB
CRLFS
"Improper input - returning to start of program"

```

```

DMESGG: DW
DB
CRLFS
"Select double density ( 1 ) or single density ( 0 ) : "

```

```

LMESGG: DW
DB
CRLFS
"Select the number of tracks ( 0=35, 1=40, 2=80 ) : "

```



0253. 63 74 20 74  
 0257. 68 65 20 6E  
 0258. 75 6D 62 65  
 025F. 72 20 6F 66  
 0263. 20 74 72 61  
 0267. 63 6B 73 20  
 0268. 28 20 30 3D  
 026F. 33 35 2C 20  
 0273. 31 3D 34 30  
 0277. 2C 20 32 3D  
 027B. 38 30 20 29  
 027F. 3A 20  
 0281. 00  
 0282. 000A

DB  
 NMESG: DW  
 DB

0  
 CRLFS  
 "Select North Star ( 0 ) or CP/M ( 1 ) data compatibility: "

0284. 53 65 6C 65  
 0288. 63 74 20 4E  
 028C. 6F 72 74 68  
 0290. 20 53 74 61  
 0294. 72 20 28 20  
 0298. 30 20 29 20  
 029C. 6F 72 20 43  
 02A0. 50 2F 4D 20  
 02A4. 28 20 31 20  
 02A8. 29 20 64 61  
 02AC. 74 61 20 63  
 02B0. 6F 6D 70 61  
 02B4. 74 69 62 69  
 02B8. 6C 69 74 79  
 02BC. 3A 20  
 02BE. 00  
 02BF. 000A  
 02C1. 53 65 6C 65  
 02C5. 63 74 20 73  
 02C9. 69 6E 67 6C  
 02CD. 65 20 28 20  
 02D1. 30 20 29 20  
 02D5. 6F 72 20 64  
 02D9. 6F 75 62 6C  
 02DD. 65 20 28 20  
 02E1. 31 20 29 20  
 02E5. 73 69 64 65  
 02E9. 64 20 6D 65  
 02ED. 64 69 61 20  
 02F1. 3A 20  
 02F3. 00  
 02F4. 000A  
 02F6. 44 72 69 76  
 02FA. 65 20 6E 6F  
 02FE. 74 20 72 65  
 0302. 61 64 79 20  
 0306. 2D 20 72 65  
 030A. 73 74 61 72  
 030E. 74 20 70 72  
 0312. 6F 67 72 61  
 0316. 6D 3F 20 28  
 031A. 20 30 20 29  
 031E. 20 6F 72 20  
 0322. 63 79 63 6C  
 0326. 65 20 28 20

DB  
 NMESG: DW  
 DB

0  
 CRLFS  
 "Select single ( 0 ) or double ( 1 ) sided media: "

DB  
 NMESG: DW  
 DB

0  
 CRLFS  
 "Drive not ready - restart program? ( 0 ) or cycle ( 1 ) : "

```

032A' 31 20 29 3A
032E' 20
032F' 00
0330' 0D0A
0332' 57 72 69 74
0336' 65 20 70 72
033A' 6F 74 65 63
033E' 74 65 64 20
0342' 2D 20 72 65
0346' 73 74 61 72
034A' 74 20 70 72
034E' 6F 67 72 61
0352' 6D 3F 20 28
0356' 20 30 20 29
035A' 20 6F 72 20
035E' 63 79 63 6C
0362' 65 20 28 20
0366' 31 20 29 3A
036A' 20
036B' 00
036C' 0D0A
036E' 46 6F 72 6D
0372' 61 74 74 69
0376' 6E 67 20 66
037A' 69 6E 69 73
037E' 68 65 64
0381' 0D0A
0383' 00

```

```

WMESG: DW DB
          DB

```

```

0 CRLFS
  "Write protected - restart program? ( 0 ) or cycle ( 1 ) : "

```

```

FMESG: DW DB
        DB

```

```

0 CRLFS
  "Formatting finished"

```

```

CRLF: DW DB PAGE
       DB 0

```

0384.

ADDRESS	OPERAND	FORMAT	EQU	PHASE	SYMBOL
1030	3E 00	NSFMT:	LD	1030H	
1032	CD 00A6		CALL	A,0	
1035	C0		RET	NZ	
1036	DD 36 0B 00		LD	(IX+0BH),0	
103A	FD 7E 02		LD	A,(IY+2)	
103D	F6 0E		OR	0EH	
103F	32 4004		LD	(4004H),A	
1042	CD 00A9		CALL	HSYNC	
1045	3E 82	NREXIT:	LD	A,82H	
1047	C8		RET	Z	
1048	CD 00A0	TRACK0:	CALL	HOME	
104B	CB 6E		BIT	5,M	
104D	28 F6		JR	Z,NREXIT	
104F	DD 36 0B 00		LD	(IX+0BH),0	
1053	3A 111C	ENTRY:	LD	A,(TRACK)	
1056	FD BE 01		CP	(IY+1)	
1059	C4 00A3		CALL	NZ,SEEK	
105C	3A 4003		LD	A,(4003H)	
105F	E6 40		AND	40H	
1061	3E 90		LD	A,90H	
1063	C0		RET	NZ	
1064	DD 36 0A 00		LD	(IX+0AH),00H	
1068	CD 00A9	WSECT0:	CALL	HSYNC	
106B	28 D8		JR	Z,NREXIT	
106D	AF		XOR	A	
106E	DD BE 0A		CP	(IX+0AH)	
1071	20 F5		JR	NZ,WSECT0	
1073	3E 90		LD	A,90H	
1075	32 4007		LD	(CONTR),A	
1078	21 4001		LD	HL,DISK	
107B	0E 00		LD	C,0	
107D	DD 71 09		LD	(IX+9),C	
1080	06 11		LD	B,11H	
1082	3E 00		LD	A,0	
1083		DEN1	EQU	\$-1	
1084	1F		RRA		
1085	3E 64		LD	A,64H	
1087	30 0F		JR	NC,CSTART	
1089	3E 18		LD	A,18H	
108A		STRACK	EQU	\$-1	
108B	1F		RRA		
108C	C6 05		ADD	A,5	
108E	FD BE 01		CP	(IY+1)	
1091	9F		SBC	A,A	
1092	E6 10		AND	10H	
1094	F6 24		OR	24H	
1096	06 20		LD	B,20H	
1098	32 4006	CSTART:	LD	(4006H),A	
109B	36 00	ZEROW:	LD	M,0	
109D	E3		EX	(SP),HL	
109E	E3		EX	(SP),HL	
109F	10 FA		DJNZ	ZEROW	
10A1	3A 1083		LD	A,(DEN1)	
10A4	B7		OR	A	

10A5 28 04  
 10A7 36 FB  
 10A9 E3  
 10AA E3  
 10AB 36 FB  
 10AD 06 5C  
 10AF 1E 20  
 10B0  
 10B1 16 20  
 10B2  
 10B3 AF  
 10B4 E3  
 10B5 E3  
 10B6 73  
 10B7 AB  
 10B8 07  
 10B9 10 F9  
 10BB 06 51  
 10BC  
 10BD E3  
 10BE E3  
 10BF 72  
 10C0 AA  
 10C1 07  
 10C2 08  
 10C3 7B  
 10C4 32 10B2  
 10C7 08  
 10C8 E3  
 10C9 E3  
 10CA 73  
 10CB AB  
 10CC 07  
 10CD E3  
 10CE E3  
 10CF 73  
 10D0 AB  
 10D1 07  
 10D2 E3  
 10D3 E3  
 10D4 73  
 10D5 AB  
 10D6 07  
 10D7 10 F4  
 10D9 E3  
 10DA E3  
 10DB 77  
 10DC 3A 10B3  
 10DF B7  
 10E0 06 11  
 10E2 28 02  
 10E4 06 20  
 10E6 E3  
 10E7 E3  
 10E8 73  
 10E9 3A 4003  
 10EC E6 10  
 10EE 28 F6  
 10F0 0C

JR LD Z, LASTS  
 EX M, 0FBH  
 EX (SP), HL  
 LD M, 0FBH  
 LD B, 5CH  
 LD E, 20H  
 EQU \$-1  
 LD D, 20H  
 EQU \$-1  
 XOR A  
 EX (SP), HL  
 EX (SP), HL  
 LD M, E  
 XOR E  
 RLC A  
 DJNZ LD B, 51H  
 EQU \$-1  
 EX (SP), HL  
 EX (SP), HL  
 LD M, D  
 XOR D  
 AF, AF'  
 EX A, E  
 LD (CPDATA), A  
 EX AF, AF'  
 EX (SP), HL  
 EX (SP), HL  
 LD M, E  
 XOR E  
 (SP), HL  
 EX (SP), HL  
 LD M, E  
 XOR E  
 RLC A  
 EX (SP), HL  
 EX (SP), HL  
 LD M, E  
 XOR E  
 RLC A  
 DJNZ LD A, (DEN1)  
 OR A  
 LD B, 11H  
 JR Z, \$+4  
 LD B, 20H  
 EX (SP), HL  
 EX (SP), HL  
 LD M, E  
 LD A, (STATUS)  
 AND INDEX  
 JR Z, ILOOP  
 INC C

LASTS:

DATA

CPDATA

D1LOOP:

DENZ

D2LOOP:

ILOOP:

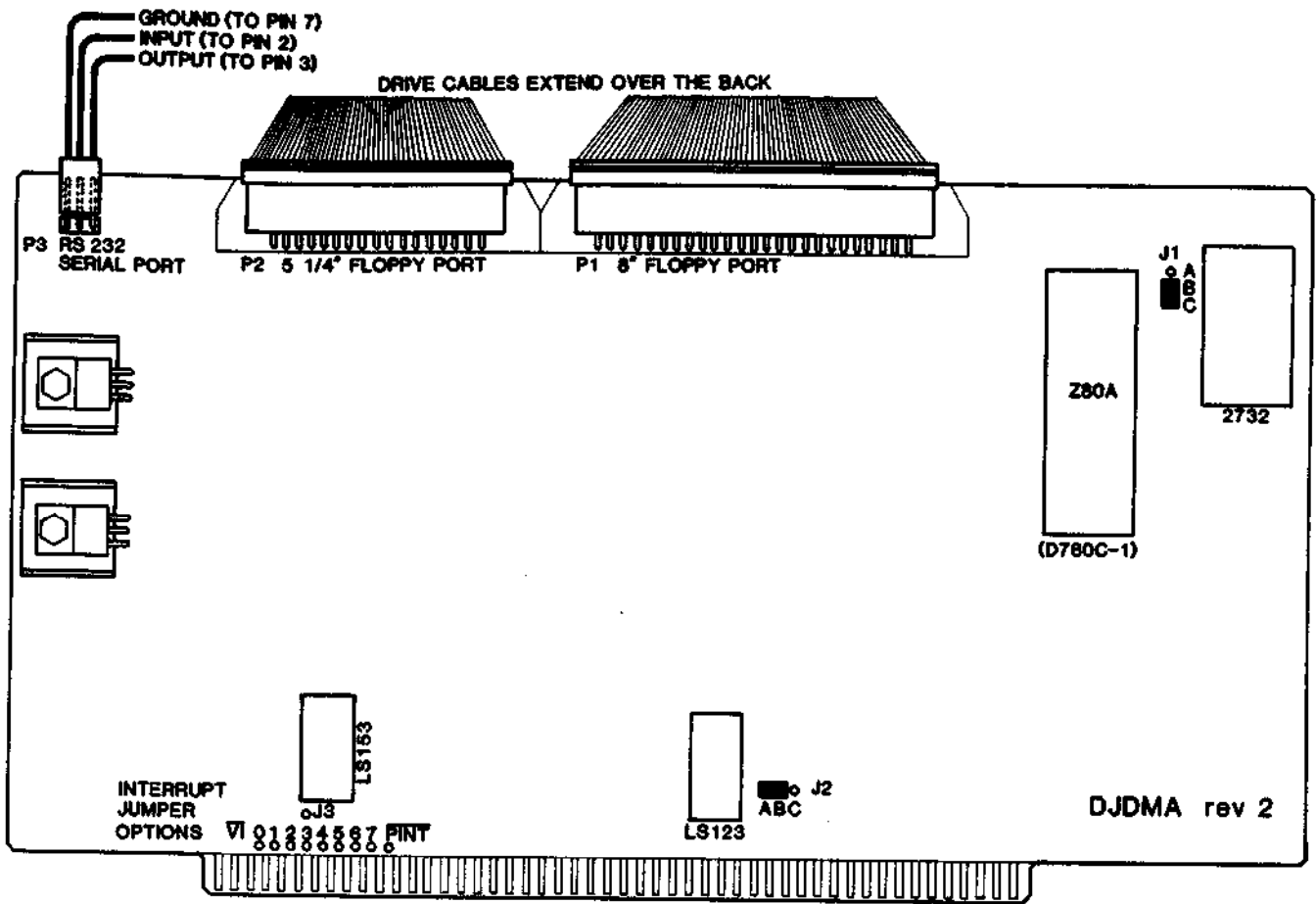
```

10F1 3E 0A          LD A,0AH
10F3 B9           CP C
10F4 20 A5       JR NZ,ZEROW
10F6 0E 00       LD C,0
10F8 3A 111D     LD A,{DSIDE}
10FB EE 00       XOR 0
10FC           EQU $-1
10FD 32 111D     LD {DSIDE},A
1100 28 0C       JR Z,FTDONE
1102 FD 7E 02   LD LD A,{1Y+2}
1105 F6 0E       OR 0EH
1107 E6 FD       AND 0FDH
1109 32 4004    LD (4004H),A
110C 18 8D       JR ZEROW
110E 32 4007    LD (CONTRL),A
1111 3E 40      A,40H
1113 C9         RET
1114 3A 111C     LD A,(TRACK)
1117 3C         INC A
1118 32 111C     LD (TRACK),A
111B C9         RET
111C 00         TRACK: 0
111D 00         DSIDE: 0
0472          .DEPHASE $
           EQU END

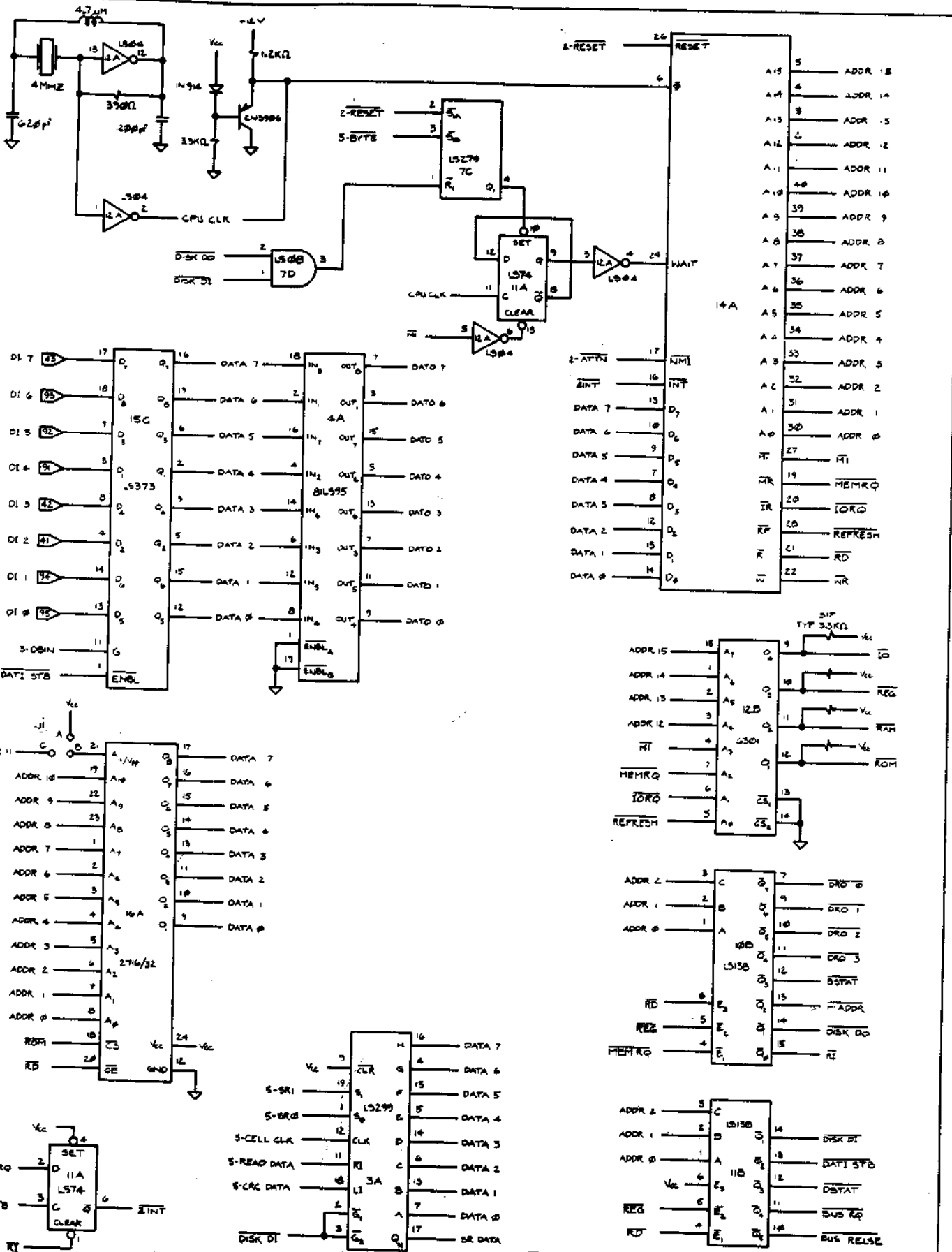
```

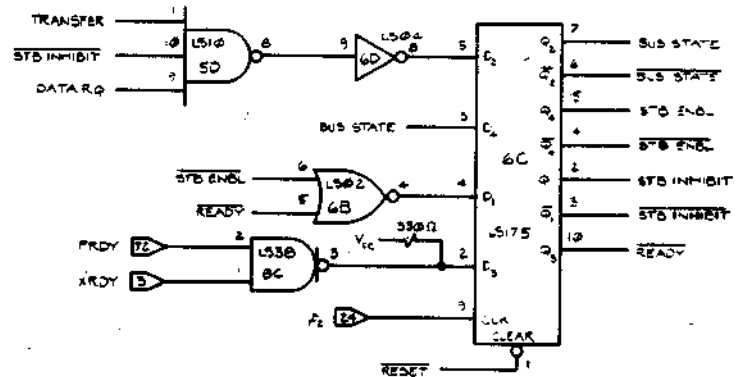
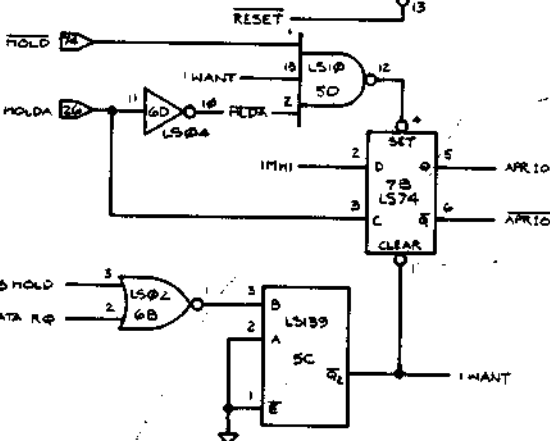
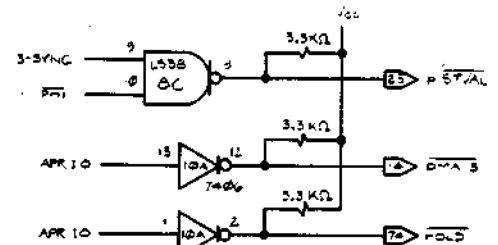
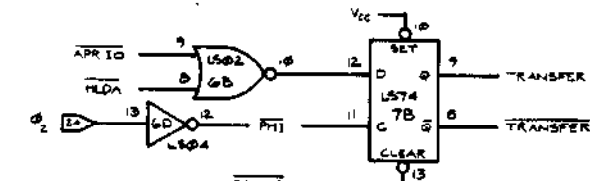
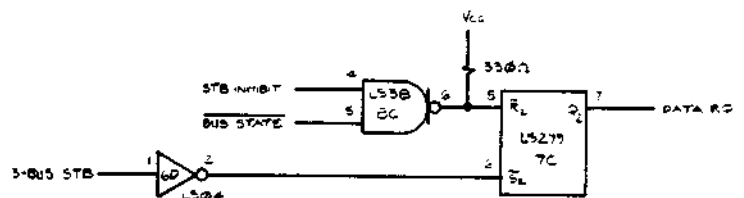
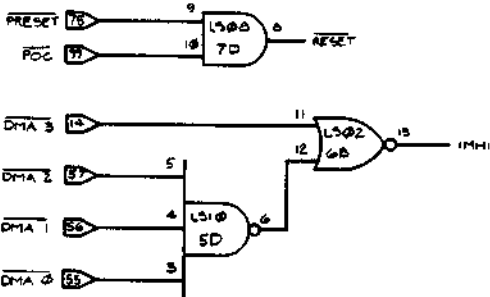
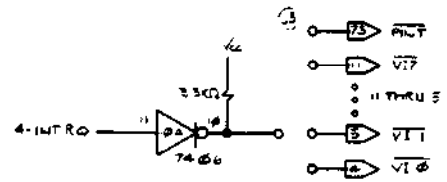
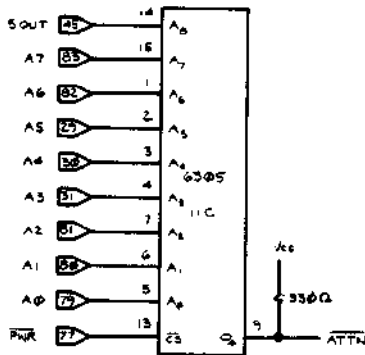
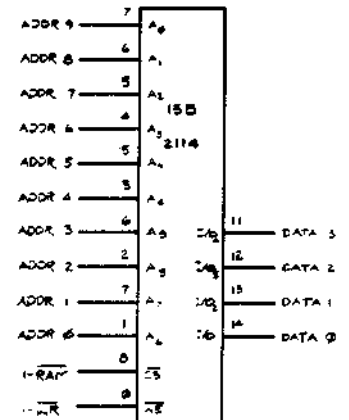
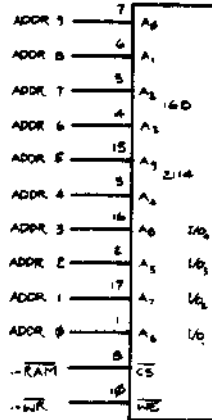
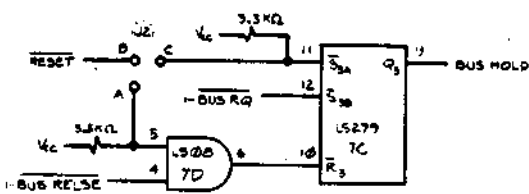
;turn off write gate

;get the current track  
;advance track value  
;update the track value  
;return with track value

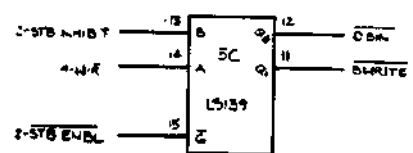
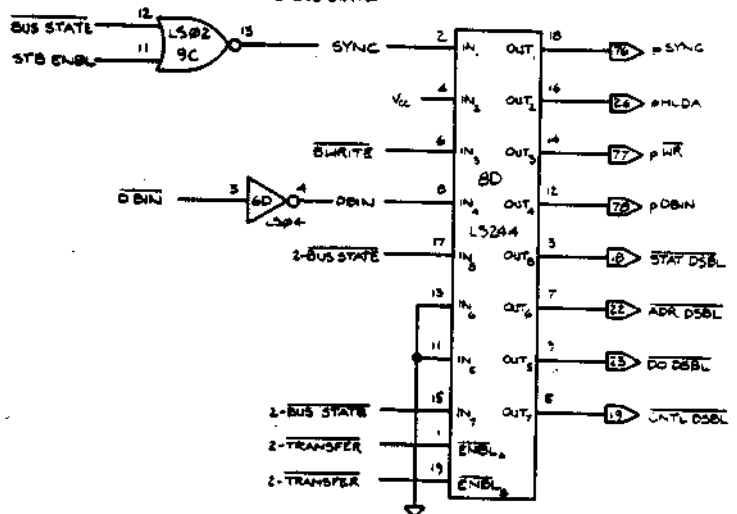
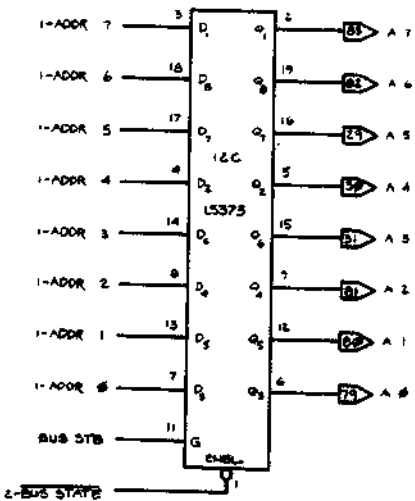
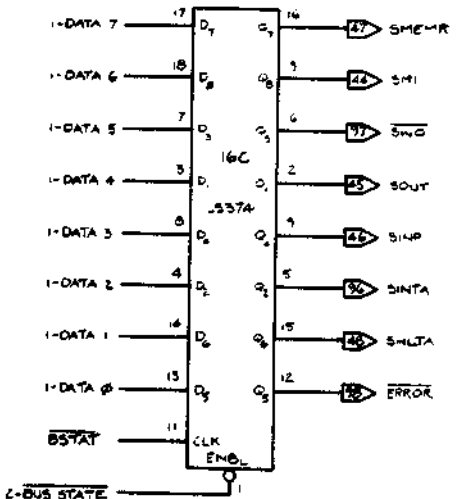
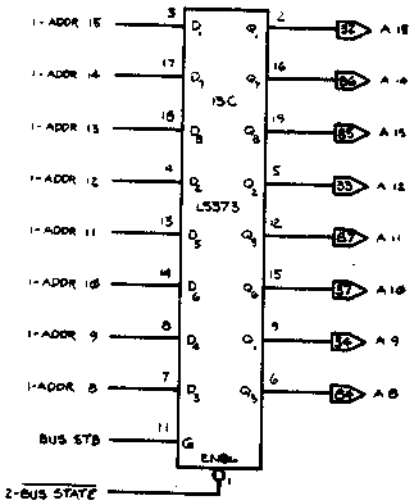
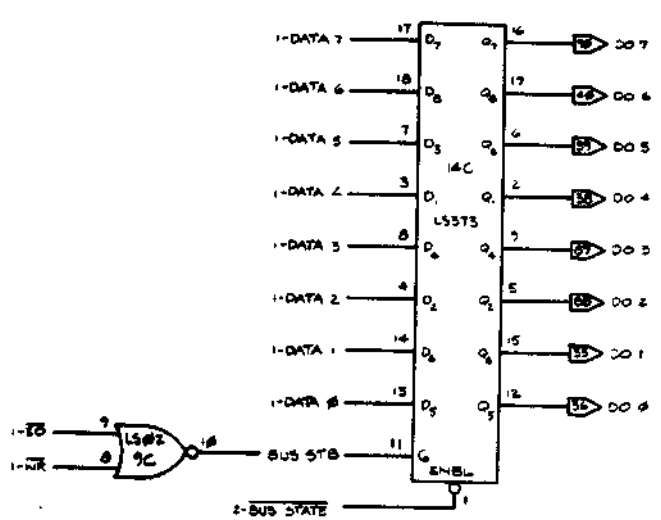
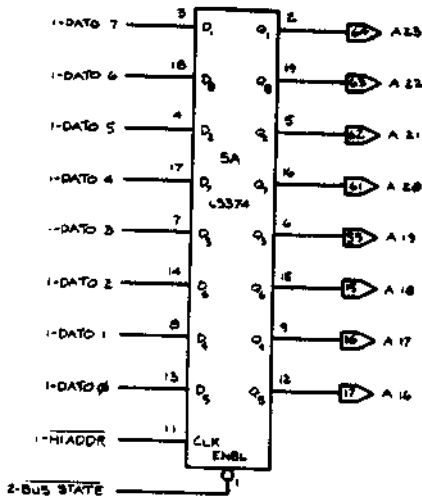


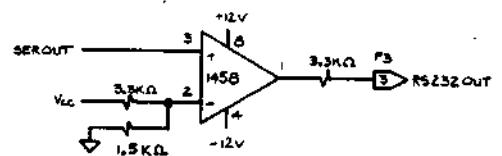
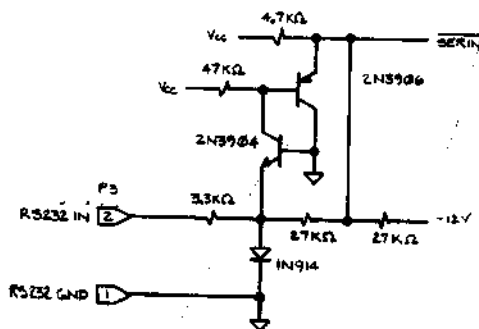
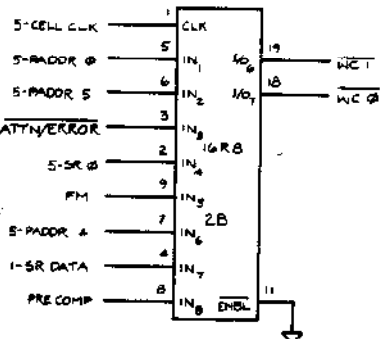
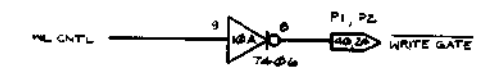
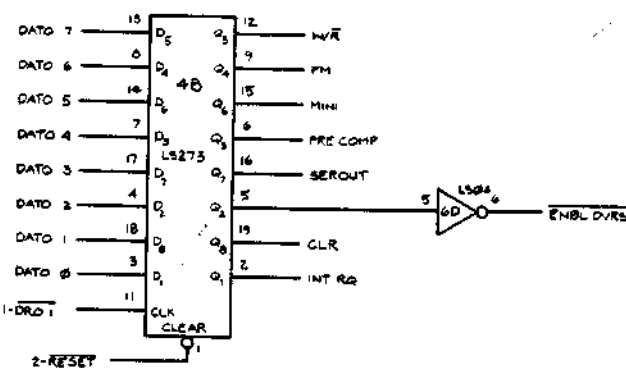
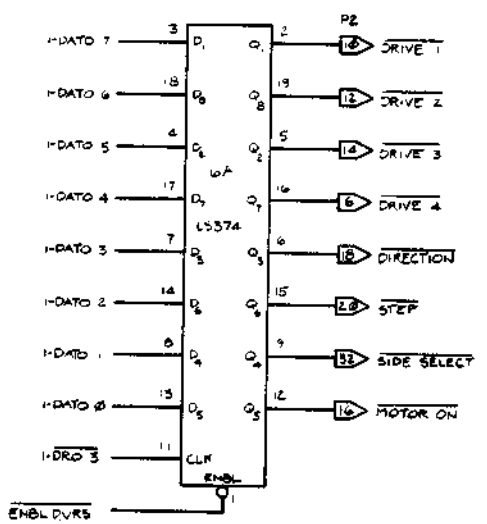
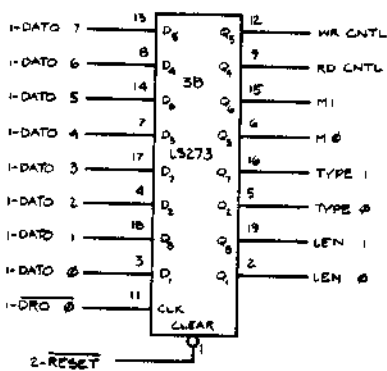
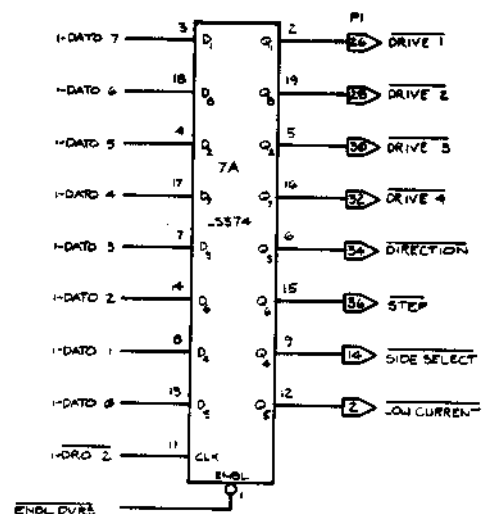
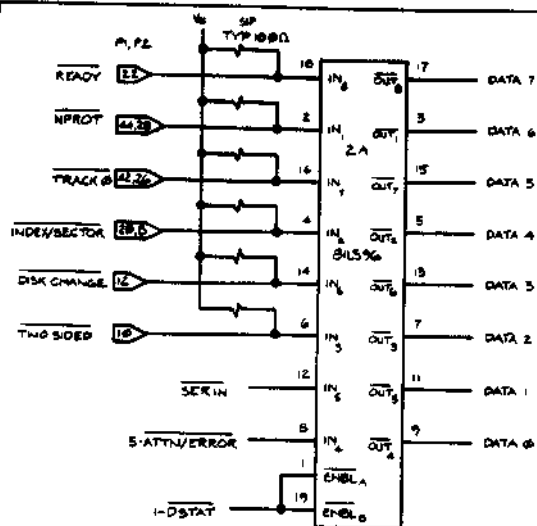
Disk Jockey / DMA Component Layout

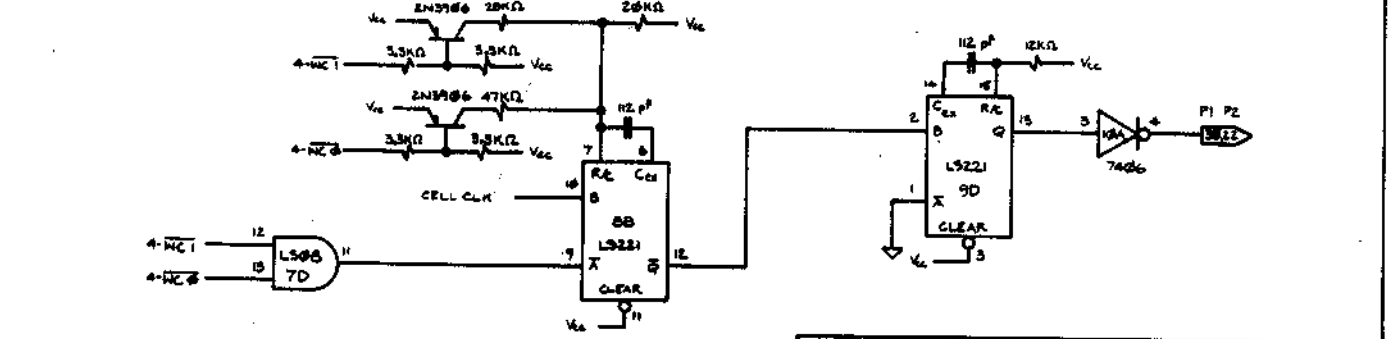
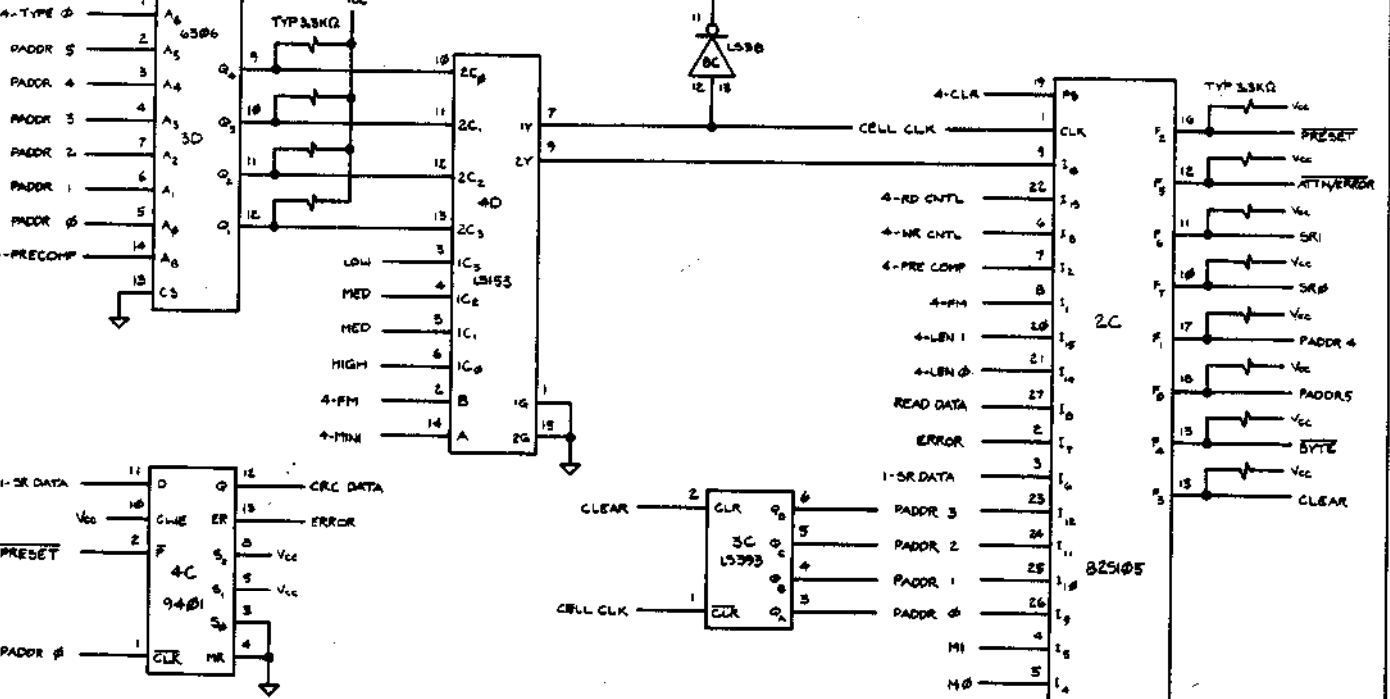
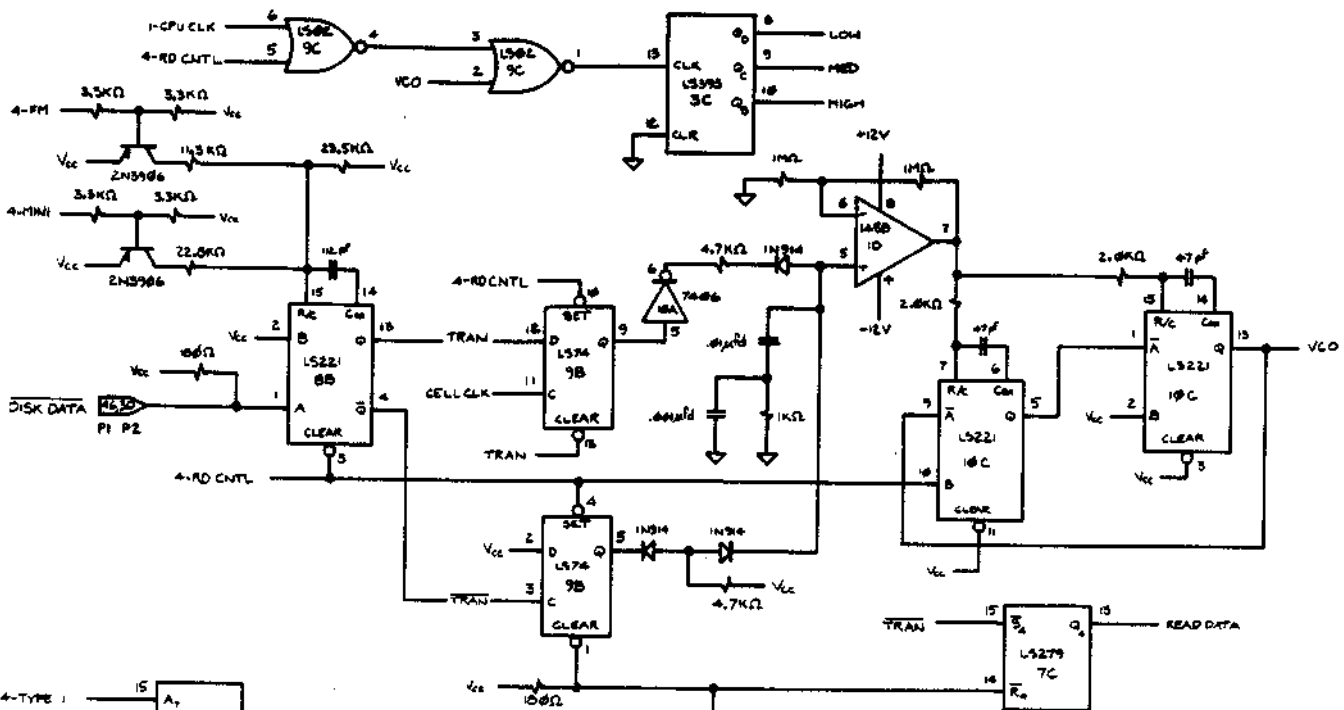












## Parts List

Amount	Function	Description
1	PC board	DJDMA
5	Diode	1N914
1	Transistor	2N3904
6	Transistor	2N3906
2	Regulator	+5 volts
1	Regulator	+12 volts
1	Regulator	-12 volts
1	Resistor	1K Ohm 1/4W 5%
2	Resistor	1 Meg Ohm 1/4W 5%
1	Resistor	12K Ohm 1/4W 5%
1	Resistor	1.2K Ohm 1/4W 5%
1	Resistor	1.5K Ohm 1/4W 5%
1	Resistor	180 Ohm 1/4W 5%
2	Resistor	27K Ohm 1/4W 5%
4	Resistor	330 Ohm 1/4W 5%
11	Resistor	3.3K Ohm 1/4W 5%
1	Resistor	390 Ohm 1/4W 5%
3	Resistor	4.7K Ohm 1/4W 5%
1	Resistor	47K Ohm 1/4W 5%
1	Resistor	2.0K Ohm 1/4W 1%
1	Resistor	20.0K Ohm 1/4W 1%
1	Resistor	28.0K Ohm 1/4W 1%
1	SIP	180K 1/8W 5% (10-pin)
1	SIP	3.3K 1/8W 5% (8-pin)
1	Inductor	4.7uh
1	Capacitor	.001mf ceramic disk
13	Capacitor	.1uf mono cap
1	Capacitor	.01 mylar cap
1	Capacitor	33pf silver/mica
2	Capacitor	47pf silver/mica
2	Capacitor	100pf silver/mica
1	Capacitor	1200pf silver/mica
1	Capacitor	620 pf silver/mica
8	Capacitor	1uf dip. tant.
1	Crystal	4 MHz
1	PCB Header	SIN RT> NHD 3
1	PCB Header	DIN RT> HD 34
1	PCB Header	DIN RT> HD 50
2	Slide Jumpers	
2	Screws	632 X 5/16 Pan Phil

Parts List

Parts List, Cont.

2	Hex Nuts	632
2	Heat Sinks	Low Profile 3 Fin
2	Heat Sinks	Slimline 5 prong
1	IC Socket	Low Profile (8-pin)
13	IC Sockets	Low Profile (14-pin)
12	IC Sockets	Low Profile (16-pin)
2	IC Sockets	Low Profile (18-pin)
15	IC Sockets	Low Profile (20-pin)
1	IC Socket	Low Profile (24-pin)
1	IC Socket	Low Profile (28-pin)
1	IC Socket	Low Profile (40-pin)
1	IC	1458
2	IC	2114-3 RAM
1	IC	7404
1	IC	7406
1	IC	74LS02
1	IC	74LS04
1	IC	74LS08
1	IC	74LS10
2	IC	74LS138
1	IC	74LS139
1	IC	74LS153
3	IC	74LS221
1	IC	74LS244
2	IC	74LS273
1	IC	74LS279
1	IC	74LS299
4	IC	74LS373
4	IC	74LS374
1	IC	74LS38
1	IC	74LS393
3	IC	74LS74
1	IC	74LS75
1	IC	81LS95
1	IC	81LS96
1	IC	PAL
1	IC	FPLA
5	IC	PROM

Subject IndexB

BRANCH IN CHANNEL, 17  
Board compatibility, 1

C

CONTROLLER HALT, 17  
CP/M data buffer, 6  
Command Pointer reset, 17  
Command parameter lists, 5  
Command status byte, 5  
Controller  
    DMA channel, 1  
    microprocessor, 1  
    supervision of data transfer, 1  
Cycle steal mode, 24

D

DJDMA self boot capability, 2  
DMA communication with main memory, 3  
Dangers of formatting diskettes, 24  
Data recovery, 13  
Data transfer, 21  
Drive values, 13

E

EXECUTE CONTROLLER ROUTINE, 19  
Extended addressing, 6

I

IEEE standards and board compatibility, 1  
Intelligent I/O channel, 1  
Interrupt request lines, 12

J

Jumpering interrupt request lines, 22

L

Listing of DJDMA Controller Commands, 5  
Listing of status byte codes, 8  
Listing of valid sector values, 7

## Subject Index

### M

#### Master

permanent, 1, 4  
temporary, 1, 4

### O

OUTPUT TO SERIAL PORT, 16

### P

Permanent master, 1, 4  
Port enable and terminal connection, 17  
Power-up or reset pointer, 4  
Primitive I/O port - DJDMA, 4  
Program Counter, 4

### R

READ CONTROLLER MEMORY, 18  
READ SECTOR, 7  
READ TRACK, 14

### S

Sector transfer sample, 14  
SENSE DRIVE STATUS, 9  
SERIAL INPUT ENABLE/DISABLE, 16  
SET CHANNEL ADDRESS, 17  
SET DMA ADDRESS, 6  
SET ERROR RETRY COUNT, 13  
SET HEAD UNLOAD/DRIVE DESELECT TIMEOUT, 14  
SET INTERRUPT REQUEST, 12  
SET LOGICAL DRIVE, 13  
SET TRACK SIZE, 18  
Serial port communication, 16  
Start command, 3  
Status flag for serial input, 16  
Stealing bus cycles, 1  
Stop command, 3

### T

Temporary master, 1, 4  
Track numbering, 8

U  
Undefined I/O devices, 16

W  
WRITE CONTROLLER MEMORY, 19  
WRITE SECTOR, 9  
WRITE TRACK, 15

Z  
Z-80A - memory transfer, 19