DJ/DMA Floppy Disk Controller
Technical Reference Manual

Revision 1
April 1982

MORROW DESIGNS

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

Table of Contents, Cont.

List of Tables

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 (ØF 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-1ØØ 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 37Ø 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-1ØØ 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-1ØØ 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

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 Ø 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 5Ø (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:	Ø 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-1ØØ 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 8Ø (hex) - the default disk data buffer of the popular CP/M operating system:

23 8Ø ØØ ØØ (hex).

2.6.2. READ SECTOR

Command code:	2Ø (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 Ø 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 Ø:

2Ø Ø5 83 ØØ ØØ

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 Ø of drive 1:

2Ø Ø6 Ø2 Ø1 ØØ

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:	Ø - 9	256 bytes
5 1/4" hard sectored double density:	Ø - 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	1Ø24 bytes

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

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 Ø through 76. This is not the case with 5 1/4 inch drives. Some have 35 tracks numbered Ø through 34, others have 4Ø tracks numbered Ø through 39, and finally, the new double track density 5 1/4 inch drives have 8Ø tracks numbered Ø through 79. The default value for 5 1/4 inch drives on the DJDMA is 4Ø. 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 Ø 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

4Ø -	normal completion - no errors
8Ø -	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
9Ø -	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 that reports status in its last byte. Not all codes apply to all commands. For example, 9Ø (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.

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 Ø - 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 Ø.

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 Ø.

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 Ø.

Bit 4 - If the media in the drive is double density, this bit is a 1. It is Ø 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 Ø.

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 Ø.

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 Ø.

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

The Ø 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 1Ø24 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 Ø - 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 Ø. If the head(s) are positioned over some other track, this bit is a Ø.

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 Ø, the 8 inch drive is not "ready" and will not respond to commands from the controller.

2.6.5. SET INTERRUPT REQUEST

Command code:	24 (hex)
Command length:	2 bytes
Command parameter list length:	Ø 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 (VIØ*, 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 4Ø (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:	Ø 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 1Ø (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 the the 8 inch drives are Ø 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 Ø through 3, while the 8 inch drives have their values changed to 4 through 7. A Ø 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 4Ø (hex), the previous logical value of the first physical 8 inch drive was Ø. 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.

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:	Ø 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 1Ø24 byte sectors and the first full sector of data encountered was Sector 6. In this case the last 3Ø72 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 Ø or 8Ø (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 Øs, 8Øs (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 Ø). 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 8Ø (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 ØØEØØØ (hex) from drive 2 and that a set DMA address command with this value has already been executed. Suppose also that there are 1Ø24 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 8Ø Ø2 F8 DF ØØ ØØ

The sector table address of ØØDFF8 (hex) has a value of eight less than ØØEØØØ (hex) since there are eight sectors on the track of the diskette. The last byte (indicated with a value of ØØ) is the overall status byte for the command. The status codes are the same as the READ SECTOR COMMAND where they are listed.

2.6.1Ø. 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 Øs as entries. Sectors that are not to be written (or rewritten) are marked with FFs (hex) while an 8Ø (hex) causes the command to terminate.

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 Ø or FF (hex). The command fills the status byte with a 4Ø (hex) when all eight data bits and two stop bits have been transmitted.

The speed of this serial port is 96ØØ 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:	Ø 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 ØØØØ3E (hex).

After loading a new character at this location, the controller writes 4Ø (hex) at location ØØØØ3F (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 Ø 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:	Ø 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 Ø or FF (hex). The controller fills this byte with a 4Ø (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 ØØØØ5Ø (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:	Ø 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:	Ø 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.

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 Ø 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 4Ø for 4Ø track drives, 77 for 77 track drives, and 8Ø for 8Ø 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:	AØ (hex)
Command length:	8 bytes
Command parameter list length:	7 bytes
Command status list length:	Ø 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 Ø and FFFF (hex). The last two bytes specify an address in the memory of the on-board Z-8ØA 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 4ØØ1 (hex), 8ØØ1 (hex), AØØØ (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:	Ø 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 Ø and FFFF (hex).

The last two bytes specify an address in the memory space of the on-board Z-8ØA microprocessor. This command transfers data from main memory to the memory of the controller. There are only 1Ø24 bytes of RAM on the controller board. This RAM starts at location 1ØØØ (hex). The only locations safe to write in are between 1Ø3Ø 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:	Ø+ bytes

The two bytes in the parameter list specify an address in the memory space of the on-board Z-8ØA 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 Ø+ 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.**

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 Øs into these first 38 (hex) bytes and places a short, 19 byte (decimal) handshake routine between ØØØØ38 and ØØØØ4A (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, 8Ø (hex) bytes are loaded between ØØØØ8Ø and ØØØØFF (hex) from the first sector on Track Ø of the disk. Finally, the controller writes a control byte to the handshake routine which causes the main CPU to branch to location ØØØØ8Ø (hex). A listing of the 19-byte handshake routine is given below.

Table 7-1. 19-Byte Handshake Routine

```
ØØØØ38    21 4A ØØ      START:   LXI   H,4A
ØØØØ3B    36 ØØ                  MVI   M,Ø
ØØØØ3D    7E             LOOP:   MOV   A,M
ØØØØ3E    B7                     ORA   A
ØØØØ3F    CA 3D ØØ               JZ    LOOP
ØØØØ42    FE 4Ø                  CPI   4ØH
ØØØØ44    C2 3D ØØ               JNZ   LOOP
ØØØØ47    C3 8Ø ØØ               JMP   8ØH
ØØØØ4A    FF                     DB    ØFFH
```

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 Ø.

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'          START:  LD      SP,ECODE+30H            ;initialize the stack pointer
0003' 21 1030                   LD      HL,1030H                ;initalize command addresss
0006' 22 0161'                  LD      (DOTCMD+1),HL
0009' 21 113A                   LD      HL,SDADVT
000C' 22 0167'                  LD      (ATCMD+1),HL
000F' 21 016F'                  LD      HL,SMESSG               ;start of program message
0012' CD 011E'                  CALL    OUTM                    ;send the message
0015' CD 012A'                  CALL    INPUT                   ;get response to drive number
0018' D2 0024'                  JP      NC,DATAOK               ;test for valid input
001B' 21 01BA'          DEXIT:  LD      HL,BMESSG               ;invalid input message
001E' CD 011E'                  CALL    OUTM                    ;send the message
0021' C3 0000'                  JP      START                   ;go back to start of program
0024' 32 045D'          DATAOK: LD      (SINGLE+1),A            ;store the drive number in code
0027' 21 01ED'                  LD      HL,DMESSG               ;type of density message
002A' CD 011E'                  CALL    OUTM                    ;send the message
002D' CD 012A'                  CALL    INPUT                   ;wait for response
0030' DA 001B'                  JP      C,DEXIT                 ;test for improper input
0033' E6 01                     AND     1                       ;density encoded in bit 0
0035' 32 032A'                  LD      (DENSTY),A              ;save for later use
0038' CA 0065'                  JP      Z,SIDE                  ;skip sector size if single density
003B' 21 0225'                  LD      HL,LMESSG               ;sector length message
003E' CD 011E'                  CALL    OUTM                    ;send the message
0041' CD 012A'                  CALL    INPUT                   ;wait for input
0044' DA 001B'                  JP      C,DEXIT                 ;test for improper input
0047' FE 03                     CP      3                       ;futher test for improper input
0049' CA 001B'                  JP      Z,DEXIT                 ;error exit
004C' 16 00                     LD      D,0                     ;form offset into sector table
004E' 5F                        LD      E,A
004F' 3C                        INC     A                       ;adjust for sector length code
0050' 32 03C5'                  LD      (DLCODE-DDFMT+DOUBLE),A ;store in format code
0053' 21 016C'                  LD      HL,STABLE
0056' 19                        ADD     HL,DE
0057' 7E                        LD      A,M                     ;fetch number of sectors
0058' 32 0407'                  LD      (DLAST-DDFMT+DOUBLE),A  ;store in format code
005B' 3E 20                     LD      A,20H                   ;sector length code is 80,100, or 0
005D' 87                DCNST:  ADD     A,A
005E' 1D                        DEC     E                       ;decrement the sector type
005F' F2 005D'                  JP      P,DCNST                 ;test for cycle done
0062' 32 03EF'                  LD      (DSIZE-DDFMT+DOUBLE),A  ;store 1/4 length in format code
0065' 21 0265'          SIDE:   LD      HL,HMESSG               ;double sided media message
0068' CD 011E'                  CALL    OUTM                    ;send the message
006B' CD 012A'                  CALL    INPUT                   ;wait for input
006E' DA 001B'                  JP      C,DEXIT                 ;test for improper input
0071' E6 01                     AND     1                       ;discard all but bit 0
0073' 32 041C'                  LD      (DDSBIT-DDFMT+DOUBLE),A ;store in format code double density
0076' 32 0532'                  LD      (SDSBIT-SDFMT+SINGLE),A ;store in format code single density
0079' 21 0151'          LOADC:  LD      HL,LSDCMD               ;load single density code command
007C' 06 0A                     LD      B,0AH                   ;command length
007E' CD 00FB'                  CALL    LCMD                    ;load the code
0081' 21 0160'                  LD      HL,DOTCMD               ;format track 0 command
0084' 06 06                     LD      B,6                     ;command length
0086' CD 00FB'                  CALL    LCMD                    ;execute the command
0089' CA 00A8'                  JP      Z,PROCED                ;zero => no error
008C' 21 029A'                  LD      HL,RMESSG               ;drive not ready message
008F' FE 82                     CP      82H                     ;drive not ready error code
0091' CA 0097'                  JP      Z,$+6                   ;test for drive not ready
0094' 21 02D6'                  LD      HL,WMESSG               ;drive must be write protected
```

```
0097'  CD 011E'              CALL    OUTM            ;send the message
009A'  CD 012A'              CALL    INPUT           ;wait for input
009D'  DA 001B'              JP      C,DEXIT         ;test for improper input
00A0'  E6 01                 AND     1               ;discard all but bit 0
00A2'  CA 0000'              JP      Z,START         ;zero => start the program over
00A5'  C3 0079'              JP      LOADC           ;go back and do the command over
00A8'  21 0327'      PROCED: LD      HL,CRLF         ;carriage return and line feed
00AB'  CD 011E'              CALL    OUTM            ;output the string
00AE'  21 1050               LD      HL,SDRDY        ;adjusted execution address of format
00B1'  3A 032A'              LD      A,(DENSTY)
00B4'  B7                    OR      A               ;test for double density
00B5'  CA 00C9'              JP      Z,CONTUE        ;make no adjustments for single density
00B8'  21 0147'              LD      HL,LDDCMD       ;load double density format command
00BB'  06 0A                 LD      B,0AH           ;command length
00BD'  CD 00FB'              CALL    LCMD            ;load the code into controller
00C0'  21 1159               LD      HL,DDADVT       ;advance track execute address
00C3'  22 0167'              LD      (ATCMD+1),HL    ;update the command execute address
00C6'  21 1030               LD      HL,1030H        ;format execute address
00C9'  22 0161'      CONTUE: LD      (DOTCMD+1),HL   ;update track format execute address
00CC'  3E 2A                 LD      A,"*"           ;send a star for a track done
00CE'  CD 0114'              CALL    OUTPUT
00D1'  21 0166'              LD      HL,ATCMD        ;advance track command
00D4'  06 06                 LD      B,6             ;command length
00D6'  CD 00FB'              CALL    LCMD            ;load the command and execute
00D9'  FE 4D                 CP      4DH             ;last track value (77 decimal)
00DB'  C2 00E7'              JP      NZ,FMTRCK       ;zero => formatting done
00DE'  21 0312'      ENDFMT: LD      HL,FMESSG       ;send final message
00E1'  CD 011E'              CALL    OUTM
00E4'  C3 0000'              JP      START           ;go format another disk
00E7'  21 0160'      FMTRCK: LD      HL,DOTCMD       ;format a track command
00EA'  06 06                 LD      B,6             ;command length
00EC'  CD 00FB'              CALL    LCMD            ;load and execute the command
00EF'  CA 00CC'              JP      Z,CONTUE+3      ;loop back for more tracks
00F2'  21 029A'              LD      HL,RMESSG       ;drive has become not ready
00F5'  CD 011E'              CALL    OUTM
00F8'  C3 00DE'              JP      ENDFMT          ;stop the formatting

00FB'  11 0050       LCMD:   LD      DE,50H          ;start of command sequence
00FE'  7E                    LD      A,M             ;get command data
00FF'  12                    LD      (DE),A          ;load into command area
0100'  23                    INC     HL              ;advance the pointers
0101'  13                    INC     DE
0102'  05                    DEC     B
0103'  C2 00FE'              JP      NZ,LCMD+3       ;test for transfer done

0106'  D3 EF         ECMD:   OUT     (0EFH),A        ;start the controller
0108'  1B                    DEC     DE              ;pointer for status byte of halt cmd
0109'  1A                    LD      A,(DE)
010A'  B7                    OR      A               ;test for command string done
010B'  CA 0109'              JP      Z,ECMD+3
010E'  3A 0053               LD      A,(53H)         ;status byte for execute command
0111'  FE 40                 CP      40H             ;test for no error
0113'  C9                    RET

0114'  21 015C'      OUTPUT: LD      HL,SOCMD+1      ;data byte of serial output command
0117'  06 05                 LD      B,5             ;serial output command string length
0119'  77                    LD      M,A             ;store the data
011A'  2B                    DEC     HL              ;back up to pointer
011B'  C3 00FB'              JP      LCMD            ;load the command and execute
```

```
011E'  7E              OUTM:   LD      A,M             ;get current byte of message
011F'  B7                      OR      A               ;test for end of message
0120'  C8                      RET     Z               ;return at end of message
0121'  E5                      PUSH    HL              ;save the character pointer
0122'  CD 0114'                CALL    OUTPUT          ;output the character
0125'  E1                      POP     HL              ;recover the character pointer
0126'  23                      INC     HL              ;advance the character pointer
0127'  C3 011E'                JP      OUTM            ;go get the next character

012A'  21 003F          INPUT:  LD      HL,3FH          ;serial input status byte
012D'  3E 40                    LD      A,40H           ;test value for status
012F'  96                       SUB     M               ;test for character ready
0130'  C2 012D'                 JP      NZ,INPUT+3      ;zero => new character ready
0133'  77                       LD      M,A             ;zero out the status byte
0134'  2B                       DEC     HL              ;back up pointer to the character
0135'  7E                       LD      A,M             ;pickup the character
0136'  F5                       PUSH    AF              ;save the data
0137'  CD 0114'                 CALL    OUTPUT          ;echo the data
013A'  F1                       POP     AF
013B'  E6 7F                    AND     7FH             ;turn it into ASCII
013D'  FE 30                    CP      30H             ;test for smaller than zero
013F'  D8                       RET     C
0140'  FE 34                    CP      34H             ;test for larger than three
0142'  3F                       CCF
0143'  D8                       RET     C
0144'  E6 03                    AND     3               ;change ASCII to binary
0146'  C9                       RET
                                PAGE
```

```
0147' A1
0148' 032B'
014A' 00
014B' 0131
014D' 1030
014F' 25
0150' 00

0151' A1
0152' 045C'
0154' 00
0155' 0112
0157' 1030
0159' 25
015A' 00

015B' 2B
015C' 00
015D' 00
015E' 25
015F' 00

0160' A2
0161' 1030
0163' 00
0164' 25
0165' 00

0166' A2
0167' 113A
0169' 00
016A' 25
016B' 00

016C' 1B
016D' 10
016E' 09
```

```
LDDCMD: DB      0A1H
        DW      DOUBLE
        DB      0
        DW      SINGLE-DOUBLE
        DW      1030H
        DB      25H
        DB      0

LSDCMD: DB      0A1H
        DW      SINGLE
        DB      0
        DW      ECODE-SINGLE
        DW      1030H
        DB      25H
        DB      0

SOCMD:  DB      2BH
        DB      0
        DB      0
        DB      25H
        DB      0

DOTCMD: DB      0A2H
        DW      1030H
        DB      0
        DB      25H
        DB      0

ATCMD:  DB      0A2H
        DW      SDADVT
        DB      0
        DB      25H
        DB      0

STABLE: DB      1BH
        DB      10H
        DB      9
        PAGE
```

```
;write controller memory command
;main memory address pointer

;byte count
;controller memory address pointer
;controller halt command
;halt command status byte

;output character to controller cmd
;output data
;output character command status
;controller halt command
;halt command status byte

;execute controller routine command
;format a track address
;execute command status
;halt command
;status byte

;advance the track value address

;26 sectors per track (256 bytes)
;15 sectors per track (512 bytes)
;8 sectors per track (1024 bytes)
```

```
016F'  0D0A                 SMESSG: DW      CRLFS
0171'  49 42 4D 20                  DB      "IBM Compatable 8 inch Format Program"
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                         DW      CRLFS
0197'  53 65 6C 65                  DB      "Select a Drive ( 0, 1, 2, or 3 ): "
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                           DB      0
01BA'  0D0A                 BMESSG: DW      CRLFS
01BC'  49 6D 70 72                  DB      "Improper input - returning to start of program"
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                         DW      CRLFS
01EC'  00                           DB      0
01ED'  0D0A                 DMESSG: DW      CRLFS
01EF'  53 65 6C 65                  DB      "Select double density ( 1 ) or single density ( 0 ): "
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                           DB      0
0225'  0D0A                 LMESSG: DW      CRLFS
0227'  53 65 6C 65                  DB      "Select the byte length of a sector ( 0=256, 1=512, 2=1024 ): "
022B'  63 74 20 74
022F'  68 65 20 62
0233'  79 74 65 20
```

```
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                      DB      0
0265'   0D0A            HMESSG: DW      CRLFS
0267'   53 65 6C 65             DB      "Select single ( 0 ) or double ( 1 ) sided media : "
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                      DB      0
029A'   0D0A            RMESSG: DW      CRLFS
029C'   44 72 69 76             DB      "Drive not ready - restart program? ( 0 ) or cycle ( 1 ): "
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                      DB      0
02D6'   0D0A            WMESSG: DW      CRLFS
02D8'   57 72 69 74             DB      "Write protected - restart program? ( 0 ) or cycle ( 1 ): "
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
```

```
030C'   31 20 29 3A
0310'   20
0311'   00                          DB      0
0312'   0D0A                FMESSG: DW      CRLFS
0314'   46 6F 72 6D                 DB      "Formatting finished"
0318'   61 74 74 69
031C'   6E 67 20 66
0320'   69 6E 69 73
0324'   68 65 64
0327'   0D0A                CRLF:   DW      CRLFS
0329'   00                          DB      0
032A'   00                  DENSTY: DB      0
                                    PAGE
```

```
032B'

1030    21 4003          DOUBLE  EQU     $
                                 .PHASE  1030H
1033    CB 7E            DDFMT:  LD      HL,STATUS
1035    3E 82                    BIT     7,M             ;check that the drive is ready
1037    C8               NREXIT: LD      A,82H           ;drive not ready error code
1038    CB 76                    RET     Z               ;error exit
103A    3E 90                    BIT     6,M             ;test for write protected
103C    C0                       LD      A,90H           ;write protected error code
103D    DD 36 0B 00              RET     NZ              ;error exit
1041    3A 10C4                  LD      (IX+0BH),0      ;reset index counter
1044    FD BE 01                 LD      A,(DTRCK)       ;get the new track value
1047    F5                       CP      (IY+1)          ;compare with current track
1048    C4 00A3                  PUSH    AF              ;save the track
104B    21 4001                  CALL    NZ,SEEK         ;move the head(s) if needed
104E    11 4007                  LD      HL,DISKD        ;pointer to disk shift register
1051    F1                       LD      DE,CONTRL       ;pointer to control port
1052    FE 2B                    POP     AF              ;recover the tack
1054    3E 04                    CP      2BH             ;compare with track 43
1056    38 02                    LD      A,4             ;no write precompensation
1058    3E 14                    JR      C,LOADPC        ;carry => track is less than 43
105A    32 1081          LOADPC: LD      A,14H           ;write precompensation bit set
105D    9F                       LD      (PRECMP),A      ;setup the write precompensation byte
105E    F6 FE                    SBC     A,A             ;push carry bit throughout accumulator
1060    FD A6 02                 OR      0FEH            ;low current bit now set
1063    F6 02                    AND     (IY+2)          ;merge with drive pattern
1065    FD 77 02                 OR      2               ;select side 0
1068    F6 0C                    LD      (IY+2),A        ;restore drive pattern
106A    32 4005                  OR      0CH             ;turn off step command
106D    06 50                    LD      (4005H),A       ;update the drive register
106F    3A 4003          DDLBL1: LD      B,50H           ;preamble length
1072    E6 10                    LD      A,(STATUS)
1074    20 F9                    AND     INDEX           ;look for index pulse
1076    3A 4003                  JR      NZ,DDLBL1       ;wait for no index pulse present
1079    E6 10            DDLBL2: LD      A,(STATUS)
107B    28 F9                    AND     INDEX
107D    3E 90                    JR      Z,DDLBL2        ;wait for leading edge of new indes pulse
107F    12                       LD      A,90H           ;control byte - normal write/no CRC
1080    3E 00                    LD      (DE),A          ;initialize control port
1081                             LD      A,0
1082    32 4006          PRECMP  EQU     $-1             ;write precompensation & controller start
1085    36 4E                    LD      (4006H),A       ;start the controller
1087    10 FC            DDLBL3: LD      M,4EH
1089    06 0C                    DJNZ    DDLBL3          ;write the preamble
108B    36 00                    LD      B,0CH           ;zero preamble length
108D    10 FC            DDLBL4: LD      M,0
108F    3E 80                    DJNZ    DDLBL4          ;write the zero preamble
1091    12                       LD      A,80H           ;control byte for 16 bit write
1092    36 52                    LD      (DE),A          ;change mode
1094    36 24                    LD      M,52H           ;first half of C2
1096    36 52                    LD      M,24H           ;second half of C2
1098    36 24                    LD      M,52H           ;another C2
109A    36 52                    LD      M,24H
109C    3E 90                    LD      M,52H           ;the third C2
109E    12                       LD      A,90H           ;control byte 8 bit write
109F    36 24                    LD      (DE),A          ;change mode
10A1    36 FC                    LD      M,24H           ;finish the sync bytes
                                 LD      M,0FCH          ;index mark
```

```
10A3  06 32          LD      B,32H           ;postamble length
10A5  36 4E          LD      M,4EH
10A7  10 FC          DJNZ    DDLBL5          ;write the postamble

10A9  06 0C          LD      B,0CH           ;zero preamble length
10AB  36 00          LD      M,0
10AD  10 FC          DJNZ    DDLBL6          ;write the preamble
10AF  3E 81          LD      A,81H           ;16 bit write mode w/CRC
10B1  12             LD      (DE),A          ;change mode
10B2  36 44          LD      M,44H           ;first half of A1
10B4  36 89          LD      M,89H           ;second half of A1
10B6  36 44          LD      M,44H           ;second A1
10B8  36 89          LD      M,89H
10BA  36 44          LD      M,44H           ;third A1
10BC  3E 91          LD      A,91H           ;8 bit write mode w/CRC
10BE  12             LD      (DE),A          ;change mode
10BF  36 89          LD      M,89H           ;finish sync bytes
10C1  36 FE          LD      M,0FEH          ;sector header ID byte
10C3  36 00          LD      M,0             ;write the track number
10C4                 DTRCK   EQU     $-1
10C5  36 00          LD      M,0             ;write the side
10C6                 DSIDE   EQU     $-1
10C7  36 01          LD      M,1             ;write the sector number
10C8                 DSECT   EQU     $-1
10C9  36 01          LD      M,1             ;sector length code
10CA                 DLCODE  EQU     $-1
10CB  3E A1          LD      A,0A1H          ;mode to write CRC bytes
10CD  12             LD      (DE),A          ;change mode
10CE  77             LD      M,A
10CF  77             LD      M,A             ;write the CRC bytes
10D0  3E 90          LD      A,90H           ;reset CRC generator
10D2  12             LD      (DE),A          ;change mode
10D3  06 16          LD      B,16H           ;4E postamble length
10D5  36 4E          LD      M,4EH
10D7  10 FC          DJNZ    DDLBL7          ;write the postamble
10D9  06 0C          LD      B,0CH           ;data field preamble
10DB  36 00          LD      M,0
10DD  10 FC          DJNZ    DDLBL8          ;write the preamble
10DF  3E 81          LD      A,81H           ;16 bit write w/CRC
10E1  12             LD      (DE),A          ;change mode
10E2  36 44          LD      M,44H           ;first half of A1
10E4  36 89          LD      M,89H           ;second half of A1
10E6  36 44          LD      M,44H           ;second A1
10E8  36 89          LD      M,89H
10EA  36 44          LD      M,44H           ;third A1
10EC  3E 91          LD      A,91H           ;8 bit write w/CRC
10EE  12             LD      (DE),A          ;change mode
10EF  36 89          LD      M,89H           ;finish the 3 sync bytes
10F1  36 FB          LD      M,0FBH          ;data header ID byte
10F3  06 40          LD      B,40H           ;sector length divided by four
10F4                 DSIZE   EQU     $-1
10F5  36 E5          DDLBL9: LD      M,0E5H          ;empty sector data byte
10F7  36 E5          LD      M,0E5H
10F9  36 E5          LD      M,0E5H
10FB  36 E5          LD      M,0E5H          ;write four fill bytes
10FD  10 F6          DJNZ    DDLBL9          ;test for data field write done
10FF  3E A1          LD      A,0A1H          ;CRC control byte
1101  12             LD      (DE),A          ;change mode
1102  77             LD      M,A             ;write the CRC bytes
```

```
1103    77
1104    3E 90
1106    12
1107    3A 10C8
110A    3C
110B    FE 1B
110C
110D    36 4E
110F    20 02
1111    3E 01
1113    32 10C8
1116    06 35
1118    36 4E
111A    10 FC
111C    20 8B
111E    36 4E
1120    06 00
1121
1122    3A 10C6
1125    A8
1126    32 10C6
1129    36 4E
112B    06 4F
112D    08
112E    36 4E
1130    3A 4003
1133    E6 10
1135    28 F7
1137    08
1138    28 0F
113A    FD 7E 02
113D    F6 0C
113F    E6 FD
1141    32 4005
1144    36 4E
1146    C3 1085
1149    36 4E
114B    36 4E
114D    36 4E
114F    AF
1150    12
1151    3E 06
1153    32 4006
1156    3E 40
1158    C9
1159    3A 10C4
115C    3C
115D    32 10C4
1160    C9
```

```
                LD      M,A
                LD      A,90H           ;turn off the CRC generator
                LD      (DE),A          ;change mode
                LD      A,(DSECT)       ;get the sector number
                INC     A
                CP      1BH             ;test for last sector +1
DLAST           EQU     $-1
                LD      M,4EH           ;first byte of postamble
                JR      NZ,$+4          ;zero => all sectors written
                LD      A,1
                LD      (DSECT),A       ;update the sector number
                LD      B,35H           ;postamble length less one
DDLBLA:         LD      M,4EH
                DJNZ    DDLBLA          ;write the postamble
                JR      NZ,DMLOOP
                LD      M,4EH           ;first fill byte
                LD      B,0             ;double sided bit test
DDSBIT          EQU     $-1
                LD      A,(DSIDE)
                XOR     B               ;conditionally switch the side byte
                LD      (DSIDE),A       ;update the side byte
                LD      M,4EH           ;second fill byte
                LD      B,4FH           ;preamble length less one
                EX      AF,AF'          ;save the double sided status
DLBLB:          LD      M,4EH           ;write a fill byte
                LD      A,(STATUS)
                AND     INDEX           ;wait for the index pulse
                JR      Z,DLBLB
                EX      AF,AF'          ;recover the double sided status
                JR      Z,DDLBLC        ;zero => track write is done
                LD      A,(IY+2)        ;drive pattern
                OR      0CH             ;turn off the step command
                AND     0FDH            ;change read/write heads
                LD      (4005H),A       ;update the command register
                LD      M,4EH           ;first preamble byte
                JP      DDLBL3          ;format the other side
DDLBLC:         LD      M,4EH           ;trailing fill byte
                LD      M,4EH           ;trailing fill byte
                LD      M,4EH           ;trailing fill byte
                XOR     A
                LD      (DE),A          ;turn off the write gate
                LD      A,6
                LD      (4006H),A       ;turn off the controller
                LD      A,40H           ;status code
                RET
DDADVT:         LD      A,(DTRCK)       ;get the current track value
                INC     A               ;increment
                LD      (DTRCK),A       ;restore the new value
                RET                     ;return with current track value
                .DEPHASE
                PAGE
```

```
045C'

1030    3E 00       SINGLE  EQU     $
                            .PHASE  1030H
1030    3E 00       SDFMT:  LD      A,0             ;second byte filled with proper drive number
1032    CD 00A6             CALL    SDRIVE          ;select the new drive
1035    C0                  RET     NZ              ;return if wrong value
1036    FD 7E 02            LD      A,(IY+2)        ;get the drive pattern
1039    F6 0F               OR      0FH             ;side 0 and no step command
103B    32 4005             LD      (4005H),A       ;update drive control register
103E    21 0000             LD      HL,0            ;delay for the head load
1041    2B          SDWAIT: DEC     HL
1042    7C                  LD      A,H
1043    B5                  OR      L
1044    20 FB               JR      NZ,SDWAIT
1046    DD 77 0B            LD      (IX+0BH),A      ;reset the index counter
1049    CD 00A0     SDTRK0: CALL    HOME            ;calibrate the head(s)
104C    CB 6E               BIT     5,M             ;test for track zero
104E    28 05               JR      Z,SNREXT
1050    21 4003     SDRDY:  LD      HL,STATUS
1053    CB 7E               BIT     7,M             ;test for the drive ready
1055    3E 82       SNREXT: LD      A,82H           ;drive not ready code
1057    C8                  RET     Z               ;error exit
1058    CB 76               BIT     6,M             ;write protect bit
105A    3E 90               LD      A,90H           ;write protect error code
105C    C0                  RET     NZ
105D    DD 36 0B 00         LD      (IX+0BH),0      ;reset the index counter
1061    3A 10B6             LD      A,(STRCK)       ;get the new track
1064    FD BE 01            CP      (IY+1)          ;compare with current track
1067    C4 00A3             CALL    NZ,SEEK         ;do track seek if necessary
106A    21 4001             LD      HL,DISKD        ;controller data register
106D    11 4007             LD      DE,CONTRL       ;control register
1070    06 28               LD      B,28H           ;preamble length
1072    3A 4003     SDLBL1: LD      A,(STATUS)
1075    E6 10               AND     INDEX
1077    20 F9               JR      NZ,SDLBL1       ;wait for no index pulse
1079    3A 4003     SDLBL2: LD      A,(STATUS)
107C    E6 10               AND     INDEX
107E    28 F9               JR      Z,SDLBL2        ;wait for leading edge of new index pulse
1080    3E 90               LD      A,90H           ;clear the CRC register & turn on write gate
1082    12                  LD      (DE),A          ;change modes
1083    3E 44               LD      A,44H           ;single density & start bit
1085    32 4006             LD      (4006H),A       ;start the controller
1088    36 FF       SDLBL3: LD      M,0FFH
108A    10 FC               DJNZ    SDLBL3          ;write the preamble
108C    3E 80               LD      A,80H           ;16 bit write mode
108E    12                  LD      (DE),A          ;change modes
108F    06 0C               LD      B,0CH           ;zero preamble length
1091    36 AA       SDLBL4: LD      M,0AAH          ;half a zero cell
1093    10 FC               DJNZ    SDLBL4          ;write the zero preamble
1095    36 F7               LD      M,0F7H          ;first half of FC
1097    3E 90               LD      A,90H           ;8 bit write mode
1099    12                  LD      (DE),A          ;change modes
109A    36 7A               LD      M,7AH           ;second half of FC
109C    06 1A               LD      B,1AH           ;postamble length
109E    36 FF       SDLBL5: LD      M,0FFH
10A0    10 FC               DJNZ    SDLBL5          ;write the postamble

10A2    3E 80       SMLOOP: LD      A,80H           ;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
10D6  12
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
```

```
                LD      (DE),A          ;change modes
                LD      B,0CH           ;sector header preamble length
SDLBL6:         LD      M,0AAH          ;half a zero cell
                DJNZ    SDLBL6          ;write the preamble
                LD      A,81H           ;enable CRC & 16 bit write
                LD      (DE),A          ;change modes
                LD      M,0F5H          ;first half of FE
                LD      A,91H           ;enable CRC & 8 bit write
                LD      (DE),A          ;change modes
                LD      M,7EH           ;second half of FE
                LD      M,0             ;write the track
STRCK           EQU     $-1
                LD      M,0             ;write the side byte
SSIDE           EQU     $-1
                LD      M,1             ;write the sector number
SSECT           EQU     $-1
                LD      M,0             ;write the sector length code
                LD      A,0A1H
                LD      (DE),A          ;change modes
                LD      M,A
                LD      M,A             ;write the CRC bytes
                LD      A,90H           ;reset the CRC
                LD      (DE),A          ;change modes
                LD      B,0BH           ;sector header postamble length
SDLBL7:         LD      M,0FFH
                DJNZ    SDLBL7          ;write the postamble
                LD      A,80H           ;16 bit write mode
                LD      (DE),A          ;change modes
                LD      B,0CH           ;data field preamble length
SDLBL8:         LD      M,0AAH          ;half a zero cell
                DJNZ    SDLBL8          ;write the preamble
                LD      A,81H           ;enable CRC & 16 bit write
                LD      (DE),A          ;change modes
                LD      M,0F5H          ;first half of FB
                LD      A,91H           ;8 bit write
                LD      (DE),A          ;change modes
                LD      M,6FH           ;second half of FB
                LD      B,80H           ;sector data field length
SDLBL9:         LD      M,0E5H
                DJNZ    SDLBL9          ;write the data field
                LD      A,0A1H
                LD      (DE),A          ;change modes
                LD      M,A
                LD      M,A             ;write the CRC bytes
                LD      A,90H           ;reset the CRC
                LD      (DE),A          ;change modes
                LD      A,(SSECT)       ;get the current sector
                INC     A               ;advance
                CP      1BH             ;compare with 27
                LD      M,0FFH          ;first postamble byte
                JR      NZ,$+4          ;zero => all sectors written
                LD      A,1
                LD      (SSECT),A       ;update the sector
                LD      B,1AH           ;postamble length less one
SDLBLA:         LD      M,0FFH
                DJNZ    SDLBLA          ;write the postamble
                JR      NZ,SMLOOP       ;test for more sectors to format
                LD      M,0FFH          ;first fill byte
                LD      B,0             ;side bit
```

```
1106
1107    3A 10B8
110A    A8
110B    32 10B8
110E    36 FF
1110    06 19
1112    08
1113    36 FF
1115    3A 4003
1118    E6 10
111A    28 F7
111C    08
111D    28 0F
111F    FD 7E 02
1122    F6 0C
1124    E6 FD
1126    32 4005
1129    36 FF
112B    C3 1088
112E    36 FF
1130    AF
1131    12
1132    3E 06
1134    32 4006
1137    3E 40
1139    C9
113A    3A 10B6
113D    3C
113E    32 10B6
1141    C9

056E'
```

```
SDSBIT  EQU     $-1
        LD      A,(SSIDE)       ;get the current side
        XOR     B               ;conditionally switch side bits
        LD      (SSIDE),A       ;update the side byte
        LD      M,0FFH          ;write second fill byte
        LD      B,19H           ;preamble length less one
        EX      AF,AF'          ;save the double sided status
SDLBLB: LD      M,0FFH          ;write a fill byte
        LD      A,(STATUS)
        AND     INDEX
        JR      Z,SDLBLB        ;wait for the index hole
        EX      AF,AF'          ;recover the double sided status
        JR      Z,SDLBLC        ;zero => single sided
        LD      A,(IY+2)        ;get the drive pattern
        OR      0CH             ;turn off the step command
        AND     0FDH            ;turn on head one
        LD      (4005H),A       ;update drive control register
        LD      M,0FFH          ;write first preamble byte
        JP      SDLBL3          ;go format the other side
SDLBLC: LD      M,0FFH          ;trailing byte
        XOR     A
        LD      (DE),A          ;turn off write gate
        LD      A,6
        LD      (4006H),A       ;turn off the controller
        LD      A,40H           ;status code
        RET
SDADVT: LD      A,(STRCK)       ;get the current track
        INC     A               ;advance track value
        LD      (STRCK),A       ;update the track value
        RET                     ;return with track value
        .DEPHASE
ECODE   EQU     $
        END
```

```
0000'  31 04A2'      START:  LD      SP,ECODE+30H
0003'  21 1030               LD      HL,1030H
0006'  22 0177'              LD      (DOTCMD+1),HL
0009'  3E 20                 LD      A,20H
000B'  32 0404'              LD      (DATA-NSFMT+FORMAT),A
000E'  32 0406'              LD      (CPDATA-NSFMT+FORMAT),A
0011'  AF                    XOR     A
0012'  32 0470'              LD      (TRACK-NSFMT+FORMAT),A
0015'  21 018C'              LD      HL,SMESSG
0018'  CD 013E'              CALL    OUTM
001B'  CD 014A'              CALL    INPUT
001E'  D2 002A'              JP      NC,DATAOK
0021'  21 01E2'      DEXIT:  LD      HL,BMESSG
0024'  CD 013E'              CALL    OUTM
0027'  C3 0000'              JP      START
002A'  32 0385'      DATAOK: LD      (FORMAT+1),A
002D'  21 024D'              LD      HL,LMESSG
0030'  CD 013E'              CALL    OUTM
0033'  CD 014A'              CALL    INPUT
0036'  DA 0021'              JP      C,DEXIT
0039'  FE 03                 CP      3
003B'  CA 0021'              JP      Z,DEXIT
003E'  16 00                 LD      D,0
0040'  5F                   LD      E,A
0041'  21 0182'              LD      HL,STABLE
0044'  19                    ADD     HL,DE
0045'  7E                   LD      A,M
0046'  32 03DE'              LD      (STRACK-NSFMT+FORMAT),A
0049'  D5                   PUSH    DE
004A'  21 0215'              LD      HL,DMESSG
004D'  CD 013E'              CALL    OUTM
0050'  CD 014A'              CALL    INPUT
0053'  D1                    POP     DE
0054'  DA 0021'              JP      C,DEXIT
0057'  E6 01                 AND     1
0059'  06 51                 LD      B,051H
005B'  CA 0065'              JP      Z,STOREO
005E'  F5                    PUSH    AF
005F'  0F                    RRCA
0060'  83                    ADD     A,E
0061'  5F                   LD      E,A
0062'  F1                   POP     AF
0063'  06 D1                LD      B,0D1H
0065'  32 03D7'      STOREO: LD      (DEN1-NSFMT+FORMAT),A
0068'  78                    LD      A,B
0069'  32 0410'              LD      (DEN2-NSFMT+FORMAT),A
006C'  D5                    PUSH    DE
006D'  21 02BF'              LD      HL,HMESSG
0070'  CD 013E'              CALL    OUTM
0073'  CD 014A'              CALL    INPUT
0076'  D1                    POP     DE
0077'  DA 0021'              JP      C,DEXIT
007A'  E6 01                 AND     1
007C'  32 0450'              LD      (DFLAG-NSFMT+FORMAT),A
007F'  CA 0086'              JP      Z,DATAC
0082'  07                    RLCA
0083'  07                    RLCA
```

```
0084'   83
0085'   5F
0086'   D5
0087'   21 0282'
008A'   CD 013E'
008D'   CD 014A'
0090'   D1
0091'   DA 0021'
0094'   E6 01
0096'   CA 00AE'
0099'   7B
009A'   E6 80
009C'   3E 10
009E'   CA 00A6'
00A1'   21 0105'
00A4'   19
00A5'   7E
00A6'   32 0406'
00A9'   3E E5
00AB'   32 0404'
00AE'   21 0167'
00B1'   06 0A
00B3'   CD 011B'
00B6'   21 0176'
00B9'   06 06
00BB'   CD 011B'
00BE'   CA 00DD'
00C1'   21 02F4'
00C4'   FE 82
00C6'   CA 00CC'
00C9'   21 0330'
00CC'   CD 013E'
00CF'   CD 014A'
00D2'   DA 0021'
00D5'   E6 01
00D7'   CA 0000'
00DA'   C3 00AE'
00DD'   21 0381'
00E0'   CD 013E'
00E3'   21 104F
00E6'   22 0177'
00E9'   3E 2A
00EB'   CD 0134'
00EE'   21 017C'
00F1'   06 06
00F3'   CD 011B'
00F6'   47
00F7'   3A 03DE'
00FA'   B8
00FB'   C2 0107'
00FE'   21 036C'
0101'   CD 013E'
0104'   C3 0000'
0107'   21 0176'
010A'   06 06
010C'   CD 011B'
010F'   CA 00E9'
0112'   21 02F4'
0115'   CD 013E'

                ADD     A,E
                LD      E,A
DATAC:          PUSH    DE
                LD      HL,NMESSG
                CALL    OUTM
                CALL    INPUT
                POP     DE
                JP      C,DEXIT
                AND     1
                JP      Z,LOADC
                LD      A,E
                AND     80H
                LD      A,10H
                JP      Z,STORED
                LD      HL,TYPE-80H
                ADD     HL,DE
                LD      A,M
STORED:         LD      (CPDATA-NSFMT+FORMAT),A
                LD      A,0E5H
                LD      (DATA-NSFMT+FORMAT),A
LOADC:          LD      HL,LFDCMD
                LD      B,0AH
                CALL    LCMD
                LD      HL,DOTCMD
                LD      B,6
                CALL    LCMD
                JP      Z,PROCED
                LD      HL,RMESSG
                CP      82H
                JP      Z,$+6
                LD      HL,WMESSG
                CALL    OUTM
                CALL    INPUT
                JP      C,DEXIT
                AND     1
                JP      Z,START
                JP      LOADC
PROCED:         LD      HL,CRLF
                CALL    OUTM
                LD      HL,ENTRY
                LD      (DOTCMD+1),HL
CONTUE:         LD      A,"*"
                CALL    OUTPUT
                LD      HL,ATCMD
                LD      B,6
                CALL    LCMD
                LD      B,A
                LD      A,(STRACK-NSFMT+FORMAT)
                CP      B
                JP      NZ,FMTRCK
ENDFMT:         LD      HL,FMESSG
                CALL    OUTM
                JP      START
FMTRCK:         LD      HL,DOTCMD
                LD      B,6
                CALL    LCMD
                JP      Z,CONTUE
                LD      HL,RMESSG
                CALL    OUTM
```

```
0118'  C3 00FE'          JP      ENDFMT

011B'  11 0050           LCMD:   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             ECMD:   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'          OUTPUT: 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                 OUTM:   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           INPUT:  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
                                 PAGE
```

```
0167'  A1
0168'  0384'
016A'  00
016B'  00EE
016D'  1030
016F'  25
0170'  00

0171'  2B
0172'  00
0173'  00
0174'  25
0175'  00

0176'  A2
0177'  1030
0179'  00
017A'  25
017B'  00

017C'  A2
017D'  1114
017F'  00
0180'  25
0181'  00

0182'  23
0183'  28
0184'  50

0185'  90
0186'  A0
0187'  C0
0188'  00
0189'  F0
018A'  D0
018B'  E0

LFDCMD: DB      0A1H
        DW      FORMAT
        DB      0
        DW      ECODE-FORMAT
        DW      1030H
        DB      25H
        DB      0

SOCMD:  DB      2BH
        DB      0
        DB      0
        DB      25H
        DB      0

DOTCMD: DB      0A2H
        DW      1030H
        DB      0
        DB      25H
        DB      0

ATCMD:  DB      0A2H
        DW      ADVTRK
        DB      0
        DB      25H
        DB      0

STABLE: DB      35
        DB      40
        DB      80

TYPE:   DB      90H
        DB      0A0H
        DB      0C0H
        DB      0
        DB      0F0H
        DB      0D0H
        DB      0E0H
        PAGE
```

```
018C'  0D0A                    SMESSG: DW      CRLFS
018E'  4E 6F 72 74                     DB      "North Star Compatable 5 1/4 inch Format Program"
0192'  68 20 53 74
0196'  61 72 20 43
019A'  6F 6D 70 61
019E'  74 61 62 6C
01A2'  65 20 35 20
01A6'  31 2F 34 20
01AA'  69 6E 63 68
01AE'  20 46 6F 72
01B2'  6D 61 74 20
01B6'  50 72 6F 67
01BA'  72 61 6D
01BD'  0D0A                            DW      CRLFS
01BF'  53 65 6C 65                     DB      "Select a Drive ( 0, 1, 2, or 3 ): "
01C3'  63 74 20 61
01C7'  20 44 72 69
01CB'  76 65 20 28
01CF'  20 30 2C 20
01D3'  31 2C 20 32
01D7'  2C 20 6F 72
01DB'  20 33 20 29
01DF'  3A 20
01E1'  00                              DB      0
01E2'  0D0A                    BMESSG: DW      CRLFS
01E4'  49 6D 70 72                     DB      "Improper input - returning to start of program"
01E8'  6F 70 65 72
01EC'  20 69 6E 70
01F0'  75 74 20 2D
01F4'  20 72 65 74
01F8'  75 72 6E 69
01FC'  6E 67 20 74
0200'  6F 20 73 74
0204'  61 72 74 20
0208'  6F 66 20 70
020C'  72 6F 67 72
0210'  61 6D
0212'  0D0A                            DW      CRLFS
0214'  00                              DB      0
0215'  0D0A                    DMESSG: DW      CRLFS
0217'  53 65 6C 65                     DB      "Select double density ( 1 ) or single density ( 0 ): "
021B'  63 74 20 64
021F'  6F 75 62 6C
0223'  65 20 64 65
0227'  6E 73 69 74
022B'  79 20 28 20
022F'  31 20 29 20
0233'  6F 72 20 73
0237'  69 6E 67 6C
023B'  65 20 64 65
023F'  6E 73 69 74
0243'  79 20 28 20
0247'  30 20 29 3A
024B'  20
024C'  00                              DB      0
024D'  0D0A                    LMESSG: DW      CRLFS
024F'  53 65 6C 65                     DB      "Select the number of tracks ( 0=35, 1=40, 2=80 ): "
```

```
0253'   63 74 20 74
0257'   68 65 20 6E
025B'   75 6D 62 65
025F'   72 20 6F 66
0263'   20 74 72 61
0267'   63 6B 73 20
026B'   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'   0D0A
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'   0D0A
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'   0D0A
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      0
NMESSG:         DW      CRLFS
                DB      "Select North Star ( 0 ) or CP/M ( 1 ) data compatibility: "

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

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

```
032A'  31 20 29 3A
032E'  20
032F'  00
0330'  0D0A                     DB      0
0332'  57 72 69 74              DW      CRLFS
0336'  65 20 70 72              DB      "Write protected - restart program? ( 0 ) or cycle ( 1 ): "
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                       DB      0
036C'  0D0A                     DW      CRLFS
036E'  46 6F 72 6D              DB      "Formatting finished"
0372'  61 74 74 69
0376'  6E 67 20 66
037A'  69 6E 69 73
037E'  68 65 64
0381'  0D0A                     DW      CRLFS
0383'  00                       DB      0
                                PAGE
```

```
0384'

1030    3E 00
1032    CD 00A6
1035    C0
1036    DD 36 0B 00
103A    FD 7E 02
103D    F6 0E
103F    32 4004
1042    CD 00A9
1045    3E 82
1047    C8
1048    CD 00A0
104B    CB 6E
104D    28 F6
104F    DD 36 0B 00
1053    3A 111C
1056    FD BE 01
1059    C4 00A3
105C    3A 4003
105F    E6 40
1061    3E 90
1063    C0
1064    DD 36 0A 80
1068    CD 00A9
106B    28 D8
106D    AF
106E    DD BE 0A
1071    20 F5
1073    3E 90
1075    32 4007
1078    21 4001
107B    0E 00
107D    DD 71 09
1080    06 11
1082    3E 00
1083
1084    1F
1085    3E 64
1087    30 0F
1089    3E 18
108A
108B    1F
108C    C6 05
108E    FD BE 01
1091    9F
1092    E6 10
1094    F6 24
1096    06 20
1098    32 4006

109B    36 00
109D    E3
109E    E3
109F    10 FA
10A1    3A 1083
10A4    B7
```

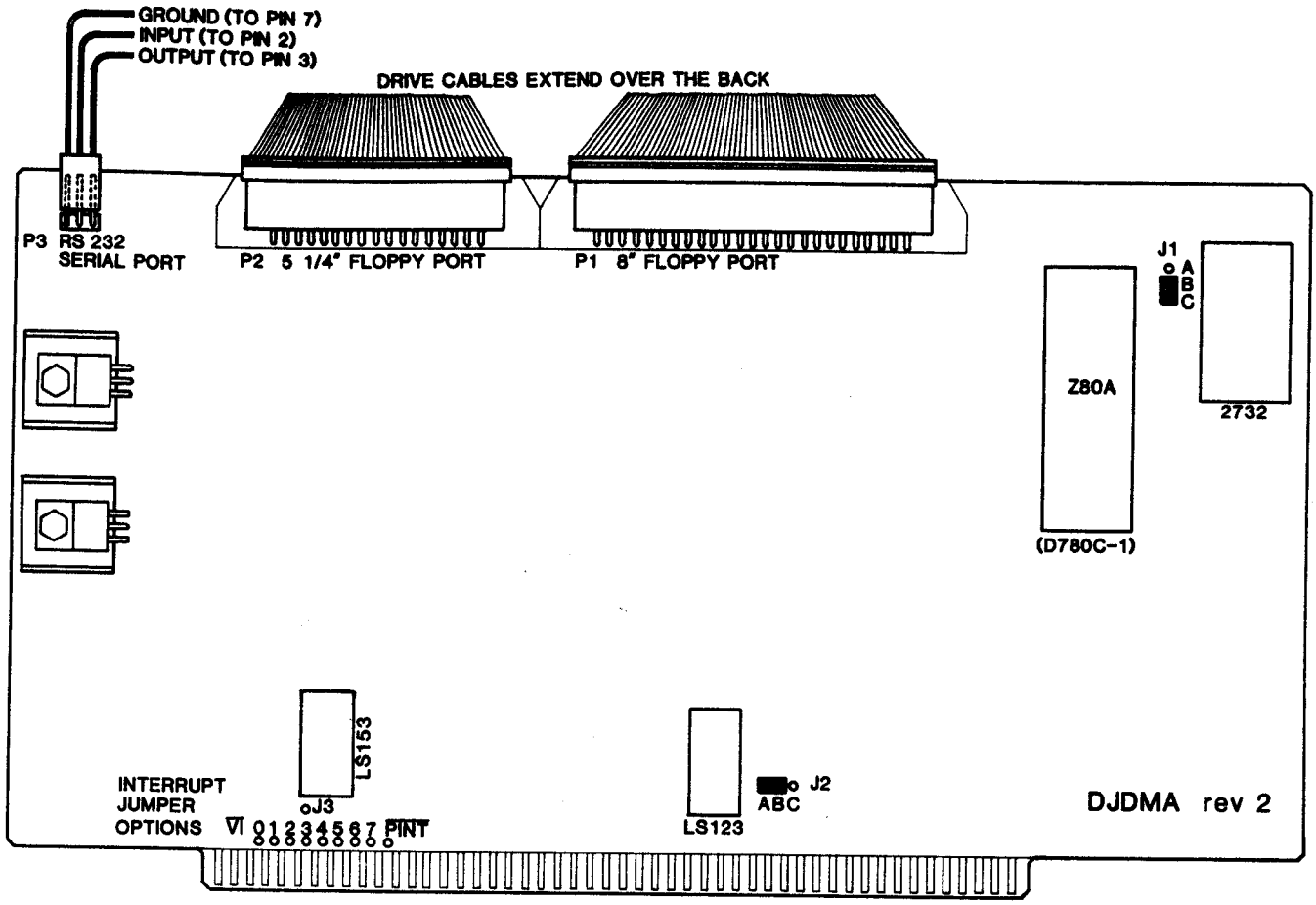
```
FORMAT  EQU     $
        .PHASE  1030H
NSFMT:  LD      A,0
        CALL    SDRIVE
        RET     NZ
        LD      (IX+0BH),0
        LD      A,(IY+2)
        OR      0EH
        LD      (4004H),A
        CALL    HSYNC
NREXIT: LD      A,82H
        RET     Z
TRACK0: CALL    HOME
        BIT     5,M
        JR      Z,NREXIT
ENTRY:  LD      (IX+0BH),0
        LD      A,(TRACK)
        CP      (IY+1)
        CALL    NZ,SEEK
        LD      A,(4003H)
        AND     40H
        LD      A,90H
        RET     NZ
        LD      (IX+0AH),80H
WSECT0: CALL    HSYNC
        JR      Z,NREXIT
        XOR     A
        CP      (IX+0AH)
        JR      NZ,WSECT0
        LD      A,90H
        LD      (CONTRL),A
        LD      HL,DISKD
        LD      C,0
        LD      (IX+9),C
        LD      B,11H
        LD      A,0
DEN1    EQU     $-1
        RRA
        LD      A,64H
        JR      NC,CSTART
        LD      A,18H
STRACK  EQU     $-1
        RRA
        ADD     A,5
        CP      (IY+1)
        SBC     A,A
        AND     10H
        OR      24H
        LD      B,20H
CSTART: LD      (4006H),A

ZEROW:  LD      M,0
        EX      (SP),HL
        EX      (SP),HL
        DJNZ    ZEROW
        LD      A,(DEN1)
        OR      A
```

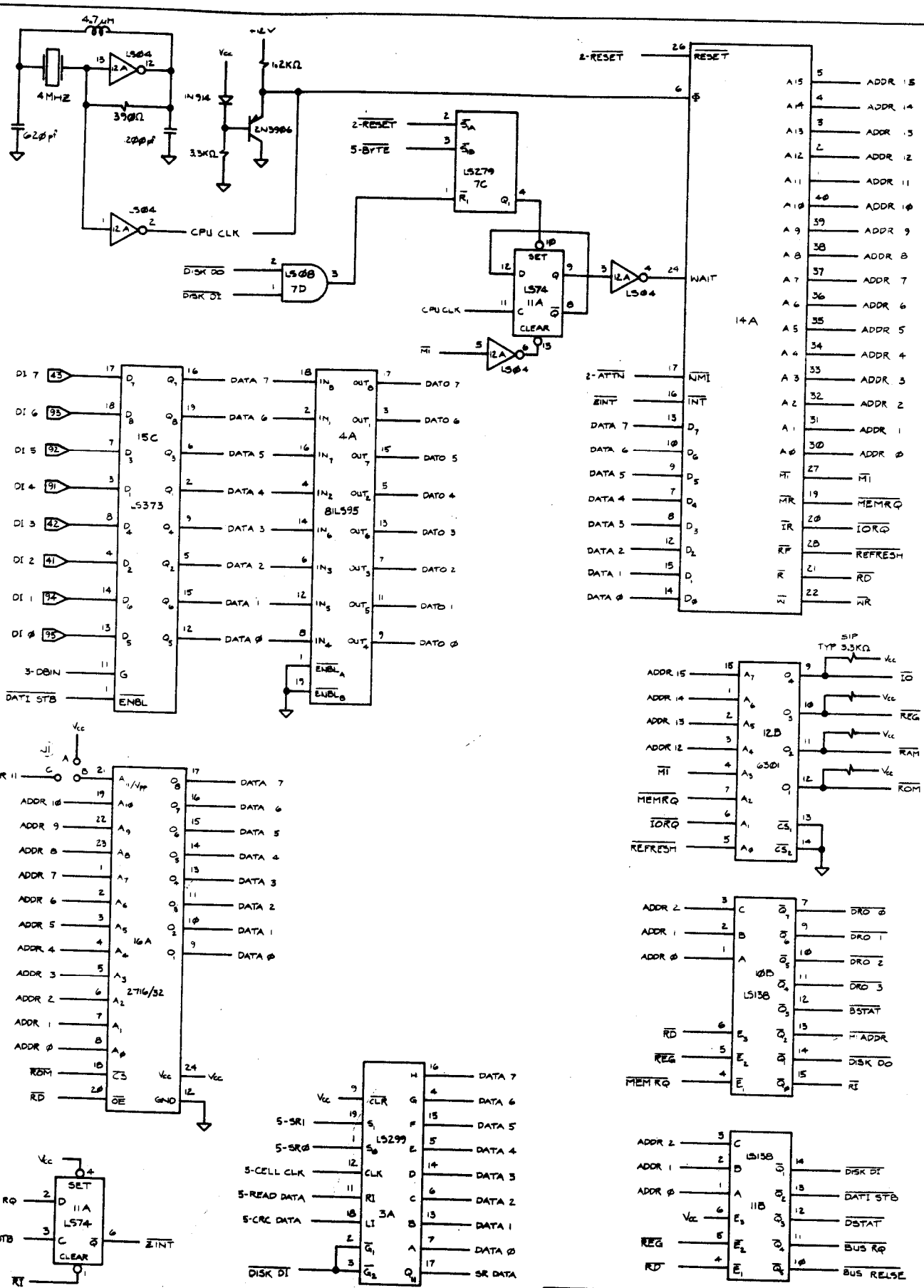
```
10A5    28 04           JR      Z,LASTS
10A7    36 FB           LD      M,0FBH
10A9    E3              EX      (SP),HL
10AA    E3              EX      (SP),HL
10AB    36 FB     LASTS:LD      M,0FBH
10AD    06 5C           LD      B,5CH
10AF    1E 20           LD      E,20H
10B0                    DATA    EQU     $-1
10B1    16 20           LD      D,20H
10B2                    CPDATA  EQU     $-1
10B3    AF              XOR     A
10B4    E3        D1LOOP:EX     (SP),HL
10B5    E3              EX      (SP),HL
10B6    73              LD      M,E
10B7    AB              XOR     E
10B8    07              RLCA
10B9    10 F9           DJNZ    D1LOOP
10BB    06 51           LD      B,51H
10BC                    DEN2    EQU     $-1
10BD    E3              EX      (SP),HL
10BE    E3              EX      (SP),HL
10BF    72              LD      M,D
10C0    AA              XOR     D
10C1    07              RLCA
10C2    08              EX      AF,AF'
10C3    7B              LD      A,E
10C4    32 10B2         LD      (CPDATA),A
10C7    08              EX      AF,AF'
10C8    E3              EX      (SP),HL
10C9    E3              EX      (SP),HL
10CA    73              LD      M,E
10CB    AB              XOR     E
10CC    07              RLCA
10CD    E3        D2LOOP:EX     (SP),HL
10CE    E3              EX      (SP),HL
10CF    73              LD      M,E
10D0    AB              XOR     E
10D1    07              RLCA
10D2    E3              EX      (SP),HL
10D3    E3              EX      (SP),HL
10D4    73              LD      M,E
10D5    AB              XOR     E
10D6    07              RLCA
10D7    10 F4           DJNZ    D2LOOP
10D9    E3              EX      (SP),HL
10DA    E3              EX      (SP),HL
10DB    77              LD      M,A
10DC    3A 1083         LD      A,(DEN1)
10DF    B7              OR      A
10E0    06 11           LD      B,11H
10E2    28 02           JR      Z,$+4
10E4    06 20           LD      B,20H
10E6    E3        ILOOP:EX      (SP),HL
10E7    E3              EX      (SP),HL
10E8    73              LD      M,E
10E9    3A 4003         LD      A,(STATUS)
10EC    E6 10           AND     INDEX
10EE    28 F6           JR      Z,ILOOP
10F0    0C              INC     C
```

```
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              DFLAG   EQU     $-1
10FD  32 111D     LD      (DSIDE),A
1100  28 0C       JR      Z,FTDONE
1102  FD 7E 02    LD      A,(IY+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     FTDONE: LD      (CONTRL),A      ;turn off write gate
1111  3E 40       LD      A,40H
1113  C9          RET
1114  3A 111C     ADVTRK: LD      A,(TRACK)       ;get the current track
1117  3C          INC     A                       ;advance track value
1118  32 111C     LD      (TRACK),A               ;update the track value
111B  C9          RET                             ;return with track value
111C  00          TRACK:  0
111D  00          DSIDE:  0

0472'             .DEPHASE
                  ECODE   EQU     $
                          END
```

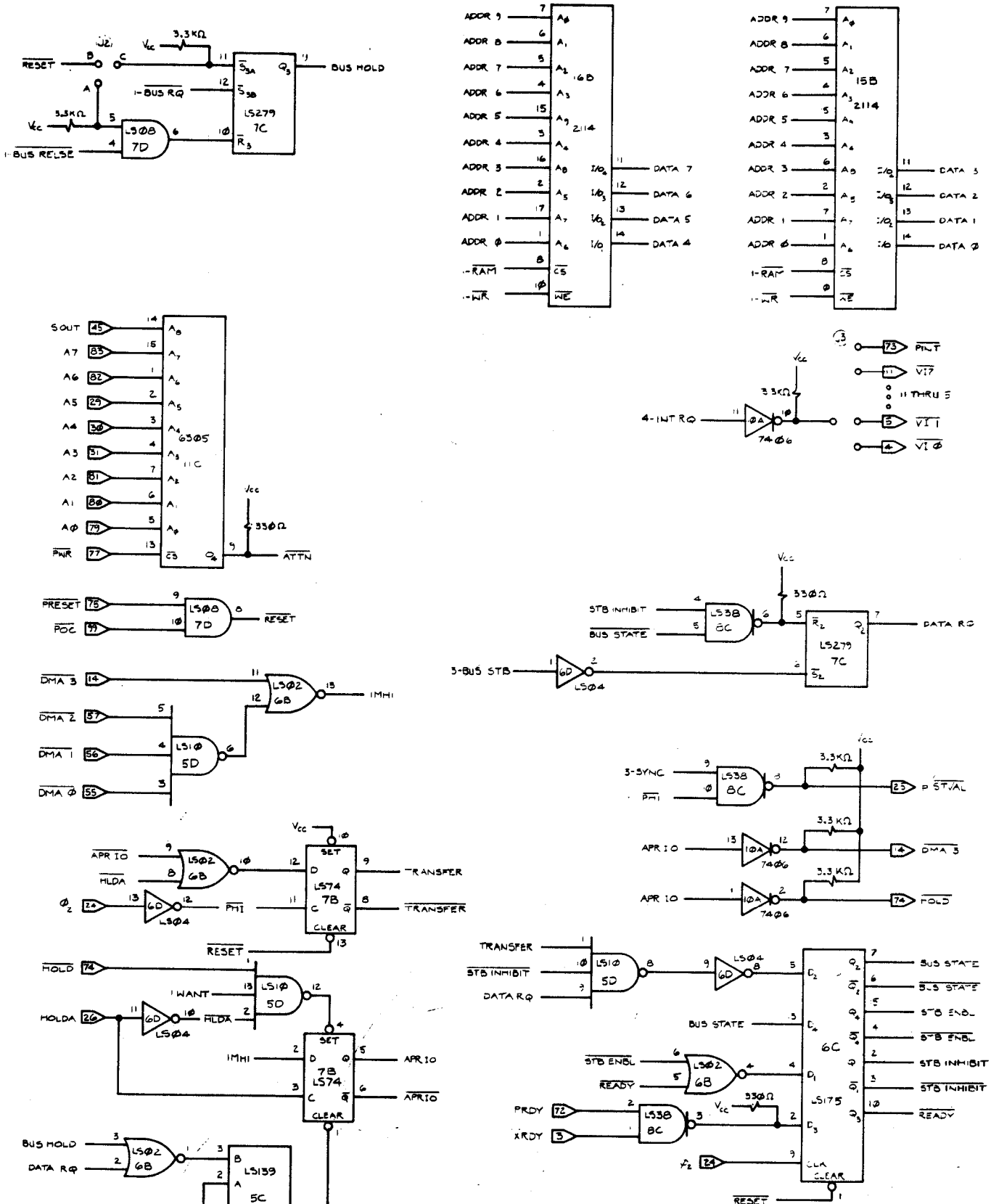


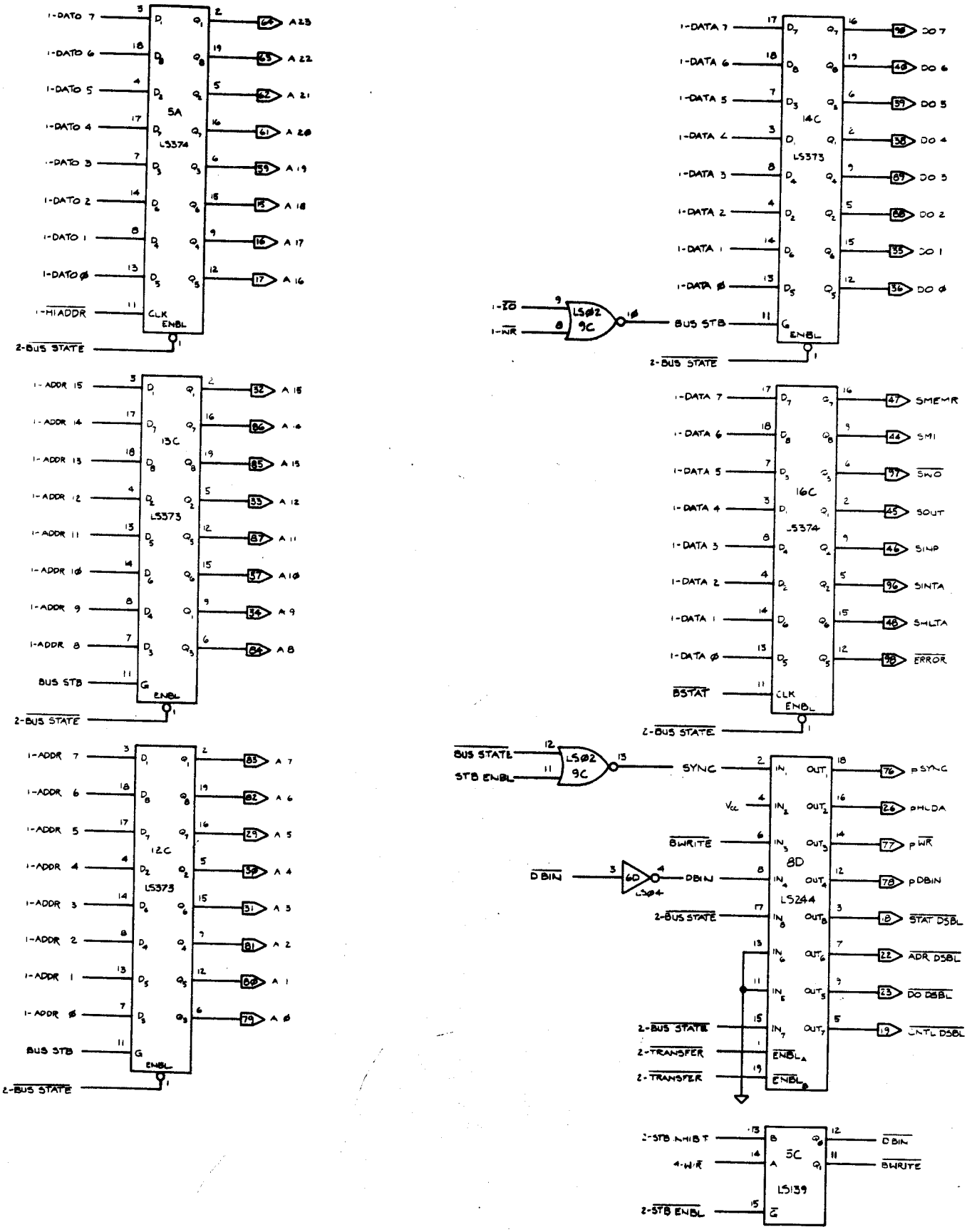
Disk Jockey / DMA Component Layout

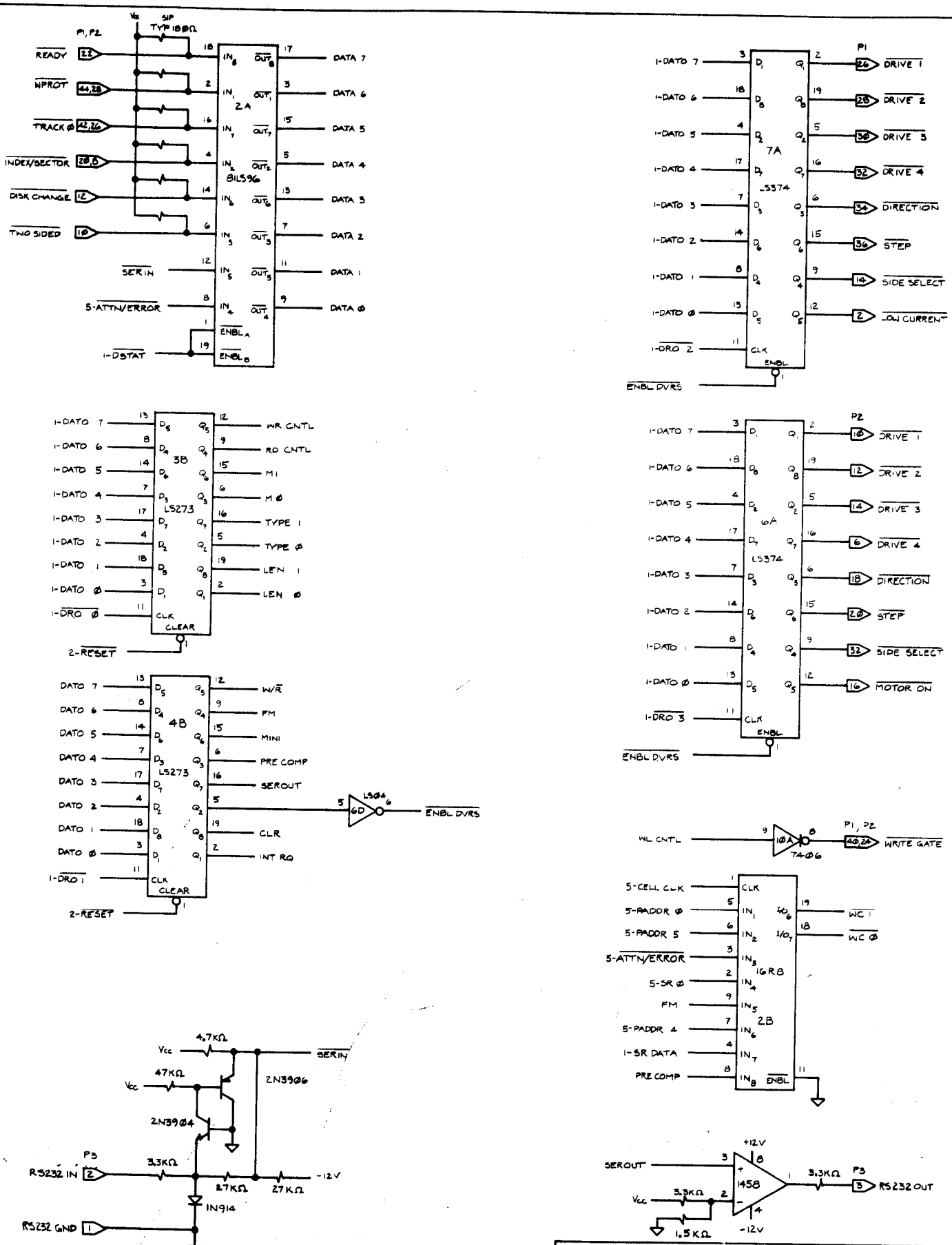


DISK JOCKEY / DMA **PAGE 1 of 5**

CPU, BUFFERS, ROM & DECODE LOGIC

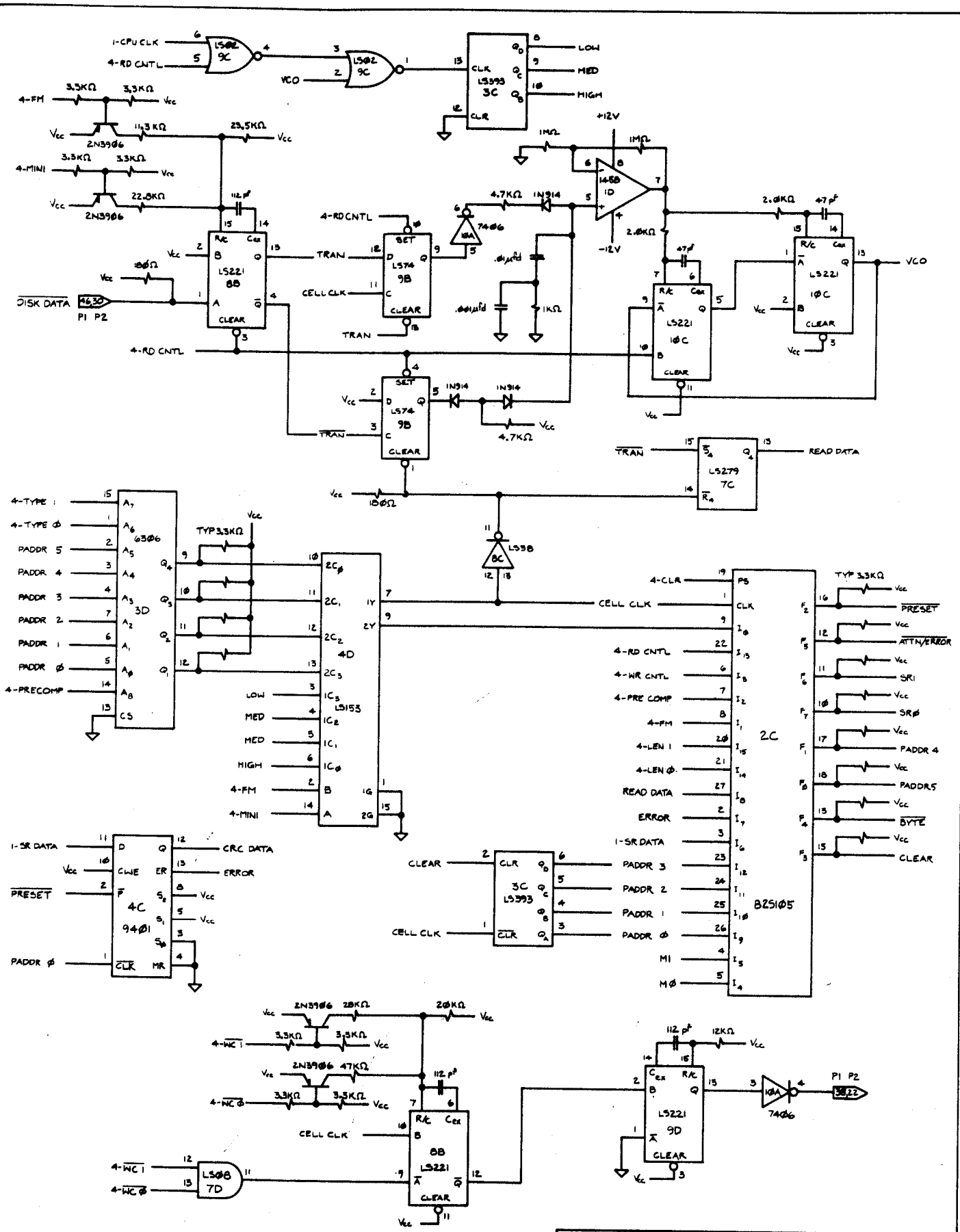






DISK JOCKEY / DMA **PAGE 4 of 5**

DRIVE & CONTROL REGISTERS



DISK JOCKEY / DMA **PAGE 5 of 5**

DISK READ/WRITE LOGIC

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	.luf 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	luf 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, 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 Index