DRAFT

User's Manual

DISK JOCKEY 2D (tm)

revision 4

Table of Contents

ATTENTION USERS OF THE NORTH STAR ZPB-2A PROCESSOR BOARD

WRITE DATA RACE CONDITION

A race condition exists in the write data logic of the ZPB-2A CPU board which can interfere with the operation of other boards on the S-100 bus if these boards utilize an internal bi-directional data bus. The following modification will alleviate this problem without degrading the performance of the North Star CPU or any other known device sharing the bus.

Locate IC 7F. It is a 74LS132 in the upper left section of the ZPB circuit board. Remove this chip from its socket, bend out pin 10 and replace the IC in its socket in such a way that pin 10 sticks out without making contact with its assigned socket hole or with any other component. Make sure that the chip is oriented correctly when it is replaced. Pin 10 should be pointing toward the top of the board. This completes the modification.

User's Manual

DISK JOCKEY 2/D tm

INTRODUCTION

The Thinker Toys DISK JOCKEY 2/D (DJ) board features three distinct subsections:

1. A floppy disk controller, capable of reading and writing data in either single density FM code or double density MFM code with write precompensation, which can be connected to any floppy disk drive plug compatible with the Shugart 800/850.

2. A baud rate selectable hardware UART serial interface that allows communication with a terminal device at TTY 20ma current loop or RS-232 levels.

3. Automatic address generation upon reset or power-up which allows a "jump start" to the boot strap program in the ROM contained on the board.

The DJ plugs into an S-100 bus slot in a system with an 8080, 8085, or Z80 (1.7MHz - 5MHz) CPU. The controller has a cable connector for attaching a flat cable to the first floppy disk drive, and can control a chain of up to four drives daisy chained on this cable. A second connector on the DJ is provided for attatching a terminal device.

The DJ uses memory mapped I/O. Device registers used to input from and output to the floppy disk and the serial port are accessed from the CPU board of the S-100 system by references to memory addresses. Some registers differ in function depending on whether they are being read or written.

Most users will not wish to use the hardware level registers directly. Instead, they can call standard disk and serial I/O subroutines contained in 1016 bytes of PROM memory on the DJ board. This PROM occupies a 1024 byte block of S-100 bus memory address space. A 1024 byte RAM is also provided which is used by the PROM firmware for the storage of various disk related variables such as the current track number, the current drive number, etc. An exact map of these variables is included at the end of the PROM listings. The remainder of the RAM may be used as a disk data buffer or for general purpose memory.

The actual addresses where the I/O registers, PROM, and RAM

appear are controlled by another PROM, referred to as the address selection PROM. The PROM is supplied with standard addresses burned into it for these registers. If the standard addresses would conflict with some other device on the system bus, a PROM burned with non-standard addresses can be substituted.

The DISK JOCKEY 2/D uses 2048 bytes of memory starting at 340:000 or E000H (standard version). The first 1016 bytes are occupied by PROM, the next 8 bytes constitute the memory mapped I/O, and the last 1024 bytes contain the RAM buffer.

PROGRAMMING SPECIFICATIONS

ROM JUMP TABLE

Most users will wish to take advantage of the standard I/O subroutines supplied in PROM on the DJ.

The user should branch to the appropriate address in a jump table in the first few words of the system ROM. Since each subroutine ends with a RET instruction, a CALL instruction should be used to branch to the subroutine.

The jump table contains jump instructions to the true address of the utility routines within the ROM. Having a jump table allows the individual routines to be updated and moved around within the ROM without having to change software that calls the routines. Let A represent the address of word Ø of the onboard ROM. In boards with standard address decoding PROMS, A = 340:ØØØQ (EØØØH). The address to call for the utility routines are then:

ADDRESS	STANDARD VALUE		SYMBOLIC VALUE	FUNCTION
	Octal	Hex		
A	340:ØØØ	EØØØ	DBOOT	DOS bootstrap routine
A+3	340:ØØ3	EØØ3	TERMIN	Serial input
A+6	340:ØØ6	EØØ6	TRMOUT	Serial output
A+9	340:Ø11	EØØ9	TKZERO	Recalibrate (seek to TRKØ)
A+12	340:Ø14	EØØC	TRKSET	Seek
A+15	340:Ø17	EØØF	SETSEC	Select sector
A+18	340:Ø22	EØ12	SETDMA	Set DMA address
A+21	340:Ø25	EØ15	DREAD	Read a sector of disk data
A+24	340:Ø3Ø	EO18	DWRITE	Write a sector of disk data
A+27	340:Ø33	EØ1B	SELDRV	Select a disk drive
A+3Ø	340:Ø36	EØ1E	TPANIC	Test for panic character
A+33	340:Ø41	EØ21	TSTAT	Serial status input
A+36	340:Ø44	EO24	DMAST	Read current DMA address
A+39	340:Ø47	EØ27	STATUS	Disk status input
A+42	340:Ø52	EØ2A	DSKERR	Loop to strobe error LED
A+45	340:Ø55	EØ2D	SETDEN	Set density
A+48	340:Ø6Ø	EØ3Ø	SETSID	Set side for 2-headed drives

The specific function of each subroutine is described below.

The subroutine upon completion will execute a RET instruction. A disk subroutine that completes normally will return with the carry flag cleared to zero. A disk subroutine that detects an error condition will return with the carry flag set to 1. A program should always test the carry flag after a return from a disk utility subroutine and branch to an appropriate error handling routine if the carry flag is set.

SERIAL I/O

There is a hardware UART on the DJ board along with a crystal controlled baud rate generator. There are sixteen different baud rates available including 12 of the most common. The baud rate of the UART must match the baud rate of the terminal connected to the DJ board in order for the serial interface to function properly.

The UART (Universal Asynchronous Receiver-Transmitter) consists of two independent sections: a transmitter section and a receiver section. Each section has two registers. In the transmitter section one register is loaded by the system bus. The contents of this bus register are transferred to a shift register where start, stop, and (conditionally) parity bits are appended. The transmitted serial data originates from this shift register. Whenever the contents of the system bus register have been transferred to the second shift register the UART sets the TBRE (Transmitter Buffer Register Empty) bit in its status register.

In the receiver section there is a shift register which assembles a parallel data word from the input serial stream after start and stop bits have been removed. When a complete data word has been assembled in this register it is loaded into a second register that is accessible from the system bus. Whenever this bus register is loaded from the receiver shift register the UART sets the DR (Data Ready) bit in its status register.

The subroutine TERMIN can be called to wait for the UART to raise the DR bit of its status register. The character is then transferred to the A register and trimmed to seven bits. Reading the UART's data register automatically resets the DR bit. The TERMIN subroutine will not return until a character arrives.

The subroutine TRMOUT causes the UART to transmit the data in the C register of the CPU. The TBRE bit of the UART's status register is tested. When TBRE is high, the contents of the C register is transferred to the UART's system bus register. This automatically resets the TBRE bit. The TRMOUT subroutine will wait for the TBRE bit to be high before transferring data to the UART.

The subroutine TPANIC can be called to detect the presence of a panic character in the serial input stream. TPANIC tests the DR bit of the UART's status register. When this bit is high, TPANIC calls the TERMIN subroutine and then compares the data from the UART with the contents of the C register. The ZERO flag of the CPU's FLAGS register is set upon completion of the TPANIC subroutine if the character in the C register has been struck on the terminal keyboard.

The subroutine TSTAT can be called to test the condition of

the DR bit in the UART's status register. Upon completion, the ZERO flag of the CPU's FLAGS register is set if the DR bit is high. The subroutine does NOT reset the DR bit.

DISK I/O

To understand the significance of the disk utility subroutines, it is necessary to say a few words about how data is organized on the disk.

Information on the disk is organized into 77 concentric tracks. The disk read/write head can be moved to any track by a series of step in or step out commands. A step in command moves the read/write head one track towards the center of the disk. A step out command moves the head one track away from the center of the disk. The numbering of the tracks is arranged so that track zero is the farthest from the center of the disk. One of the responsibilities of the Western Digital 1791 controller is to know the current track number over which the read/write head is located and to calculate how many step in or step out commands are necessary to move the head to a desired new track.

Once the read/write head has been moved to the desired track, the rotation of the disk will move a circle of magnetic material beneath the head. Within this circle of material, data is recorded in distinct regions called sectors. The sector is the smallest amount of information that can be separately read from or written to the disk. There are three different sector formats that IBM currently supports. The table below details the relationship between the size of a sector and the number of sectors that can fit on a single track.

	bytes of data per sector	sectors per track
SINGLE DENSITY	128	26
	256	15
	512	8
DOUBLE DENSITY	256	26
	512	15
	1Ø24	8

In the header field which preceeds the data field of a sector, the track number, the side, the sector number and the sector length are recorded. During read or write commands, this header is read before data transfers take place. Whenever a seek

command is issued which causes the the read/write head to move to a new track the firmware on the DJ board performs a verify which reads this sector header to make sure the head is positioned correctly and to determine if there is any change in the sector length or the density of the recorded information. If there is an error as to the track number, the firmware automatically issues a seek to track zero command to position the head over a known track.

The disk drive has a sensor that reports when the read/write head is physically positioned at track zero. A series of step out commands must be issued by the 1791 controller until this status line becomes active. This operation will always position the head to the same physical track. The seek to track zero command is often called a recalibrate command and is a standard utility subroutine supplied with the disk firmware.

Transferring a sector of disk data between memory and the disk therefore involves the following steps, each corresponding to a subroutine call to the Disk Jockey firmware (with the exception of error checking):

Specify the track number the read/write head should be positioned over during subsequent data transfers between the disk and memory.

Check for error conditions.

Specify the sector number that will be involved in subsequent data transfers between the disk and memory.

Check for error conditions.

Specify the starting memory address of block of data that is to be transfered to or form the disk.

Check for error condition.

Actually perform the read or write operation.

Check for error conditions.

ROM SUBROUTINES

TRKSET - The value in the C register of the CPU specifies what track the read/write head will be positioned over when the next disk read or disk write operation is issued. A bounds check is made for a value greater than or equal to zero and less than or equal to 76. If the value in the C register is within these bounds, the contents of the C register is written into the RAM location. TRACK.

Otherwise no action is taken, the carry flag is set and the subroutine returns to the calling program.

SECTOR - The value in the C register of the CPU specifies what sector will be involved in the next disk read or write operation. A bounds check is made for a value greater than or equal to 1 and less than or equal to 26. If the value the C register is within these bounds, the data in C is transfered the the RAM location SECTOR and a normal return is made. Otherwise no action is taken, the carry flag is set and the subroutine returns to the calling program. Just prior to a disk transfer operation a comparison is made between the value in SECTOR and the maximum number of sectors on the track that transfer is to take place on. If the value in SECTOR exceeds the maximum number of sectors, the transfer operation is aborted and error information is reported.

SETDMA - During disk transfer operations blocks of data are moved to and from the disk. These blocks can be 128, 256, 512, or 1Ø24 bytes long. The starting address of a data block that will be involved in the next disk transfer operation is specified by the B-C register pair when the SETDMA subroutine is called. Since the disk registers are memory mapped, the firmware has been designed to try to protect them from being written into or read from during disk transfer operations. Accordingly, a bounds check is performed before the DMA address is recorded in the Disk Jockey RAM. If a 1Ø24 byte data transfer to or from the disk would cause memory references to the I/O resgisters of the disk controller, the carry flag is set and the routine returns with no action taken. If the value of the B-C pair is such that there could not be any memory references to the last eight locations of the Disk Jockey ROM during a subsequent disk operation, the contents of the B-C pair are written into the memory location of the Disk Jockey RAM specified by the label DMAADR. The carry flag is cleared and the routine ends.

SELDRV - The value of the C register determines which of 4 disk drives will be selected for the next disk transfer operation. A bounds check is performed on C. If the value in C is greater than 3, the carry flag is set and the routine returns with no action taken. If the value in C is between zero and three, this data is written in the Disk Jockey RAM at the location specified by the label DISK. The carry flag is cleared and the routine returns to the calling program.

SETSID - Double sided floppy disk drives have two read/write heads so that information can be stored and retrieved from both sides of the diskette. The two heads are

positioned so that they are both on the same track one directly below the other. They also share common read/write electronics. Therefore only one of these heads can be selected at a time. Bit Ø of the C register is used to select which of the two heads on a double sided drive will be used during the next disk transfer operation. A zero in bit Ø will select the bottom head and a 1 will select the top head. Selecting a side and selecting a disk are independent operations. If side zero is selected then regardless of the disk selected, side zero will always be accessed until SETSID is called. Finally, if the selected disk is single sided, side zero will always be selected regardless of the results of the SETSID routine.

SETDEN - The 1791 Floppy Disk Controller operates in two modes: single density FM (Frequency Modulation) mode or double density MFM (Modified Frequency Modulation) mode. Bit Ø of the C register determines what density the 1791 will be operating in when the next disk transfer operation is initiated. Care must be exercised in the use of this routine. Under certain circumstances, if the density is changed in between disk transfers on the same track, the micro-program that the 1791 controller executes could fall into an error loop that it could not recover from. In such a case the system would have to be reset before further disk operations could be performed. The density mode of the 1791 can safely be changed when a subsequent disk transfer operation will occur on a different track than the last. It should be noted that the firmware of the Disk Jockey has the ability to automatically set the density mode of the 1791. Whenever a new drive is to be selected or whenever the head is not loaded, the Disk Jockey firmware performs a "read header" operation just after positioning the read/write head (if necessary) and just before attempting to perform a disk transfer. This "read header" operation is used to establish the density of the (possibly new) track and to determine the length of the sectors on this track. If the density has not changed from the last "read header" operation or if the calling program has set the density correctly through the use of SETDEN, the process of reading the sector header is slightly faster (by approximately one and a half diskette revolutions) than it would be if the initial assumption concerning the density was wrong.

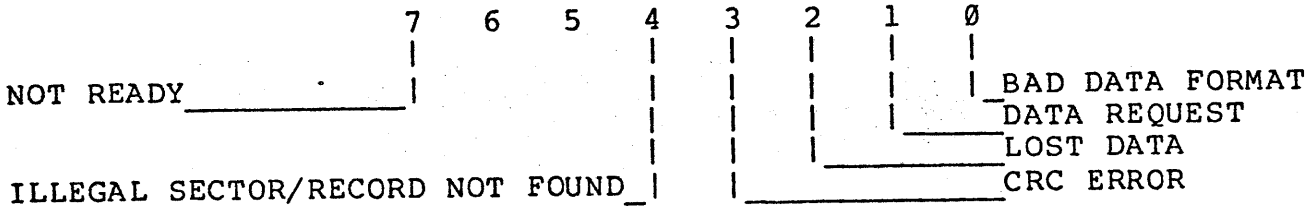
TKZERO - This subroutine positions the read/write head to the outer-most track of the diskette: track ØØ. The track zero sensor is used to determine this positioning and no "read header" verify operation is performed. There are several side effects of positioning the head at track zero: (1) a flag is set in the Disk Jockey RAM to force

a "read header" density/position verify operation prior to the next disk transfer operation and (2) the mode of the 1791 controller will be forced to single density as long as disk transfer operations occur on track zero. All IBM compatible diskettes have track zero formatted in single density and condition (2) above relieves the system software of the burden of conditionally changing density every time the head is moved to track zero. If the rest of the disk is recorded in double density, the Disk Jockey firmware will automatically switch back to double density when the head is moved away from track zero without the intervention of external software.

READ - This subroutine transfers information from the diskette to memory. The first task is to select the proper disk drive. If the new drive is not the same as the current drive, the load head time-out flag is set and the current drive is updated to be the new drive. Next, the "head loaded" flag is tested. If the head is not loaded or if the current drive was not the same as the new drive, the head load time-out flag is set. The firmware then merges the drive select bits with the head select bit and physically selects a drive, loads the head(s), and selects a side (if the drive is double sided). If the head load time-out bit is set, a 40 millesecond delay occurs to allow for the head to settle after loading. Next the "ready" line from the drive is tested. If the drive is not ready, the head is unloaded and the routine returns to the calling program with the carry bit set and an 80H in the A register. If the drive is ready, the head is positioned in accordance with the most recent seek operation. Head motion (including a head load) or a change of disk drive will cause the firmware to verify the track position by doing a "read header" operation. The correct density of the track is also determined during this operation and the density mode is changed if necessary. If the 1791 controller cannot read the header information in either density, the head is moved to track zero, the carry is set, and the read operation is terminated with an 11H in the A register. If the head is correctly positioned, the size of the sectors on the current track is encoded in the Disk Jockey RAM. The firmware uses this information to find the value of the highest addressable sector. This value is compared to the that specified by the most recent set sector operation. If the desired sector has a value too large for the present track, the head is unloaded, the carry flag is set and the routine returns with a 10H in the A register. If the value is acceptable, the data from this sector is transfered to memory starting at the address specified by the most recent set DMA operation. The length of this transfer is

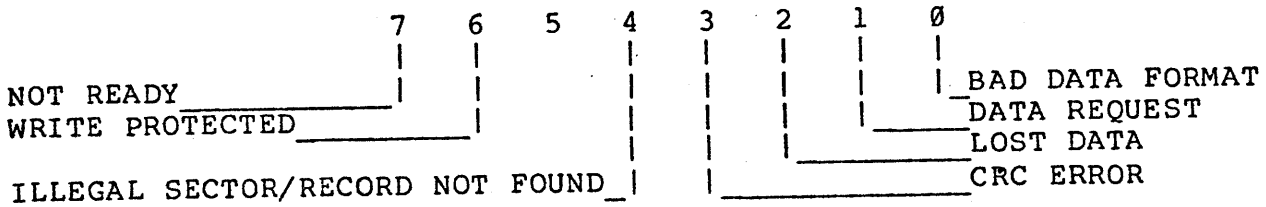
determined by the length of the sectors on the current track. The last two bytes of data on the sector are not read into memory. These are the CRC check sum bytes and are used to detect data transfer errors. The 1791 chip processes these bytes and then updates its status register. The last operation that the routine performs is to place the status information in the A register and conditionally set the carry flag. The details of these status bits are illustrated below.

"DREAD" REGISTER A ERROR BITS



DWRITE - The flow of logic for this routine is exactly the same as described above in the read data operation up to the point where the information transfer is to take place. If all the conditions for a data transfer as described above are satisfied, a write sector command is issued to the 1791 controller and information is transfered from memory to the disk drive starting at the memory address specified by the most recent DMA operation. This data is written on the sector specified by the most recent set sector operation and the head is positioned over the track specified by the most recent seek operation. As the controller writes data on the disk it is continually computing two CRC check sum bytes. After the last byte of data has been written on the diskette, the two check sum bytes are appended to the sector by the controller for later use when the sector is read back into memory. As with the read operation the controller updates its status register after the last CRC byte has been written on the diskette. These status bits are placed in the A register just before control is returned to the calling program. The carry flag is conditionally set from these bits. The details of this status information can be seen below.

"DWRITE" REGISTER A ERROR BITS

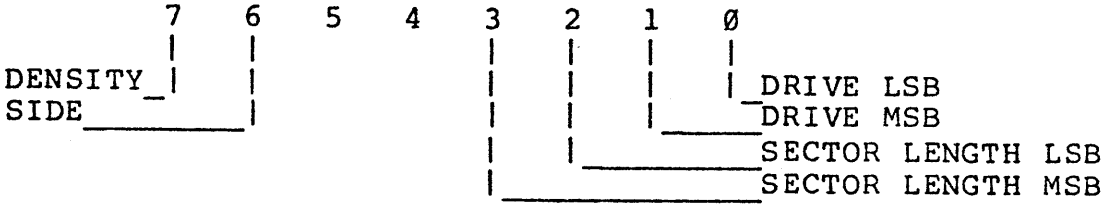


DBOOT - Branching to this routine will initiate a bootstrap load operation from the floppy disk. 128 bytes of data will be read (single density mode) into the first half of the 3rd page of the Disk Jockey RAM (normally 340:000Q or E000H). The bootstrap routine terminates with a branch to the first location of this block. Typically sector 1 of track zero will contain another bootstrap program whose job it is to load a Disk Operating System (DOS) such as Disk/ATE or CP/M. If the bootstrap read is not successful, control is passed to the DSKERR utility which is described below. Before sector one is read into memory, various memory locations of the Disk Jockey RAM are initialized. Also DBOOT goes through a several second delay the first time it is called after power-up. In order to effect an orderly start-up sequence, DBOOT does not require that the drive have a diskette in place when it is called. If the drive is not ready when DBOOT is called, it falls into a loop that turns on the LED at the top of the controller and slowly pulses the activity light at the front of the drive. This was done so that DBOOT could be started before a diskette was inserted in the drive. When a diskette has been inserted, the door should be closed just AFTER the activity light has been pulsed.

DMAST - This subroutine loads the B-C register pair with the current value of the DMA address recorded in the Disk Jockey RAM.

STATUS - This subroutine loads the B register with the sector number involved in the last disk transfer operation. It loads the C register with the track number the head is currently positioned over. Finally, it loads the A register with a bit pattern indicating the drive involved in the last disk transfer operation, the length of the sectors on the current track, the side specified by the last SETSID call, and whether or not data on the current track is written in single or double density format. The details of how this information is encoded in the A register is presented below.

A REGISTER BIT PATTERN



DRIVE MSB	DRIVE LSB	DRIVE NO.
0	0	DRIVE A
0	1	DRIVE B
1	0	DRIVE C
1	1	DRIVE D

SIDE BIT	SIDE SELECTED
0	SIDE 0
1	SIDE 1

SECTOR LENGTH MSB	SECTOR LENGTH LSB	SECTOR LENGTH	DENSITY
0	0	128	SINGLE
0	1	256	DOUBLE
1	0	512	DOUBLE
1	1	1024	DOUBLE

DSKERR - Calling this routine will put the CPU into a loop which will cause the LED (Light Emitting Diode) at the top left portion of the controller board to flash on and off at intervals of about a second. This routine takes no parameters and will not return-- its primary usefulness is to indicate when a hard error has occured during the bootstrap load operation.

DISKETTE INITIALIZATION

Before a new diskette can be successfully used, it must be initialized. Most diskettes are sold pre-initialized. However, it is sometimes necessary to reinitialize a diskette. The process of initializing a diskette involves writing the header field of every sector of every track onto the diskette. None of the subroutines described above can be used to write these header fields. This is a safety measure to ensure that an erroneous branch to the firmware PROM cannot re-initialize a diskette, destroying all the data recorded on it. The initialization function for diskettes is typically provided by a command included in the Disk Operating System. Disk/ATE diskettes furnished by Morrow's/Thinker Toys contain commands FMT128, FMT256, FMT512 and FMT1Ø24 to allow the user to format diskettes in any of the four IBM compatible formats. CP/M diskettes from Thinker Toys contain a command called FORMAT which allows the CP/M user to format diskettes in single or dual density.

RECAP OF REGISTER A ERROR BITS

"SETDMA"	7	6	5	4	3	2	1	0	BIT
DMA ADDRESS SET TO DJ I/O SPACE									

"DREAD"	7	6	5	4	3	2	1	0	BIT
NOT READY									
ILLEGAL DMA ADDRESS									
ILLEGAL SECTOR/RECORD NOT FOUND									
CRC ERROR									
LOST DATA									
DATA REQUEST									
BAD DATA FORMAT									

"DWRITE"	7	6	5	4	3	2	1	0	BIT
NOT READY									
WRITE PROTECTED									
ILLEGAL DMA ADDR									
ILLEGAL SECTOR/RECORD NOT FOUND									
CRC ERROR									
LOST DATA									
DATA REQUEST									
BAD DATA FORMAT									

"SELDRV"	7	6	5	4	3	2	1	0	BIT
INVALID DRIVE NUMBER									

UTILIZING DISK JOCKEY FIRMWARE

Data transfers to and from the disk must be preceeded by calls to certain Disk Jockey routines. The function of these routines is to set up parameters that will be used during the transfer. The following procedure is suggested:

1) Select the drive to be involved in the transfer. This is accomplished by calling the routine "SELDRV" with the proper drive number in register C. The drive need not be selected before every transfer. A drive once selected will remain selected until another drive is specified. For 2-headed drives, the side of a drive should be specified by calling the SETSID routine with the desired side number in the C register.

2) If the drive has not been accessed before, the read/write head of the drive is in an unknown position. To initialize the drive a call should be made to "TKZERO" in order to bring the head to track zero.

3) Set the DMA address. This involves calling the routine "SETDMA" with the correct value in the B-C register pair. It is not necessary to set the DMA address before every data transfer. If data is always being read into the same area of memory, then only one "SETDMA" call need be made.

4) Set the read/write head over the desired track. This involves a call to "TRKSET" with the desired track number in register C. It is only necessary to call the "TRKSET" routine when changing tracks. If the data transfer involves the same track as the previous transfer then no call to "TRKSET" should be performed.

5) Set the desired sector number. The sector can be set by calling "SETSEC" with the correct sector number in register C. If the sector has not changed since the previous "SETSEC" call, as with a read-modify-write sequence, then this routine may be skipped.

6) Read or write the desired sector. The controller can now be commanded to read or write to the disk by calling "DREAD" or "DWRITE".

The order in which these operations occur is not important with the exception that the "DREAD" or "DWRITE" routine must be called last.

Data Transfer Examples

READ:

Suppose sectors 5, 6, 7 and 8 of track 12, drive 1 are to be read into memory starting at location 7:ØØØQ (7ØØH). The following program will do this:

Utilizing Disk Jockey Firmware

Example of Disk Read

```
001:000 061 356 346      1    READ     LXI   SP,0E6EEH  set up the stack
001:003 257              2             XRA   A          select drive A
001:004 117              3             MOV   C,A
001:005 315 363 341      4             CALL  SELDRV
001:010 315 362 341      5             CALL  TKZERO     recalibrate the head
001:013 016 014          6             MVI   C,12       seek the head to
001:015 315 313 342      7             CALL  TRKSET     track 12
001:020 001 005 004      8             LXI   B,4:005Q   sector count&number
001:023 305              9             PUSH  B          save sector cnt&num
001:024 001 000 160      10            LXI   B,7000H    set up read address
001:027 315 011 342      11   LOOP     CALL  SETDMA
001:032 301              12            POP   B          restore sect to read
001:033 305              13            PUSH  B
001:034 315 166 342      14            CALL  SETSEC     set up sect to read
001:037 315 042 342      15            CALL  DREAD      read the sector
001:042 332 070 001      16            JC    ERROR      test for error
001:045 301              17            POP   B          restore sect cnt&num
001:046 005              18            DCR   B          update count
001:047 312 073 001      19            JZ    DONE
001:052 014              20            INR   C          update sector number
001:053 305              21            PUSH  B          save count&number
001:054 315 352 341      22            CALL  DMAST      dma address into B-C
001:057 041 000 001      23            LXI   H,100H     add sector size to
001:062 011              24            DAD   B          current address
001:063 345              25            PUSH  H          new address into B-C
001:064 301              26            POP   B
001:065 303 027 001      27            JMP   LOOP       continue reading
001:070 303 070 001      28   ERROR    JMP   ERROR      error stop
001:073 303 073 001      29   DONE     JMP   DONE
```

Example of Disk Read

```
0100    31 EE E6        1   READ    LXI   SP,0E6EEH  set up the stack
0103    AF              2           XRA   A          select drive A
0104    4F              3           MOV   C,A
0105    CD F3 E1        4           CALL  SELDRV
0108    CD F2 E1        5           CALL  TKZERO     recalibrate the head
010B    0E 0C           6           MVI   C,12       seek the head to
010D    CD CB E2        7           CALL  TRKSET     track 12
0110    01 05 04        8           LXI   B,4:005Q   sector count&number
0113    C5              9           PUSH  B          save sector cnt&num
0114    01 00 70        10          LXI   B,7000H    set up read address
0117    CD 09 E2        11  LOOP    CALL  SETDMA
011A    C1              12          POP   B          restore sect to read
011B    C5              13          PUSH  B
011C    CD 76 E2        14          CALL  SETSEC     set up sect to read
011F    CD 22 E2        15          CALL  DREAD      read the sector
0122    DA 38 01        16          JC    ERROR      test for error
0125    C1              17          POP   B          restore sect cnt&num
0126    05              18          DCR   B          update count
0127    CA 3B 01        19          JZ    DONE
012A    0C              20          INR   C          update sector number
012B    C5              21          PUSH  B          save count&number
012C    CD EA E1        22          CALL  DMAST      dma address into B-C
012F    21 00 01        23          LXI   H,100H     add sector size to
0132    09              24          DAD   B          current address
0133    E5              25          PUSH  H          new address into B-C
0134    C1              26          POP   B
0135    C3 17 01        27          JMP   LOOP       continue reading
0138    C3 38 01        28  ERROR   JMP   ERROR      error stop
013B    C3 3B 01        29  DONE    JMP   DONE
```

Utilizing Disk Jockey Firmware

WRITE:

The following program writes from memory starting at 200:000Q (8000H) onto tracks 4,5, and 6 of disk drive 1.

```
001:000 061 356 346
001:003 257
001:004 117
001:005 315 363 341
001:010 315 362 341
001:013 001 000 177
001:016 315 011 342
001:021 076 004
001:023 062 112 001
001:026 117
001:027 315 313 342
001:032 001 001 032
001:035 305
001:036 315 352 341
001:041 041 000 001
001:044 011
001:045 345
001:046 301
001:047 315 011 342
001:052 301
001:053 305
001:054 315 166 342
001:057 315 123 342
001:062 332 107 001
001:065 301
001:066 014
001:067 005
001:070 302 035 001
001:073 072 112 001
001:076 074
001:077 376 007
001:101 302 023 001
001:104 303 104 001
001:107 303 107 001
001:112 000
```

```
1   WRITE   LXI   SP,0E6EEH  set up the stack
2           XRA   A          select drive A
3           MOV   C,A
4           CALL  SELDRV
5           CALL  TKZERO     recalibrate the he
6           LXI   B,8000H-100H set initial adrs
7           CALL  SETDMA
8           MVI   A,4        initial track numb
9   TLOOP   STA   TEMP       save track number
10          MOV   C,A        seek to correct tr
11          CALL  TRKSET
12          LXI   B,32:001Q  sector count&numbe
13  SLOOP   PUSH  B          save sect and coun
14          CALL  DMAST      get current address
15          LXI   H,100H     update to next sect
16          DAD   B
17          PUSH  H          move address to B-C
18          POP   B
19          CALL  SETDMA     set up new address
20          POP   B          restore sect cnt&nu
21          PUSH  B
22          CALL  SETSEC     set up next sector
23          CALL  DWRITE     write the data
24          JC    ERROR      test for error
25          POP   B          recover sect cnt&nu
26          INR   C          update sector
27          DCR   B          update count
28          JNZ   SLOOP
29          LDA   TEMP       get current track
30          INR   A          update track
31          CPI   7          check if all done
32          JNZ   TLOOP      continue to next tr
33  DONE    JMP   DONE
34  ERROR   JMP   ERROR      error exit
35  TEMP    DB    0          track storage
36
```

WRITE:

The following program writes from memory starting at 200:000Q (8000H) to tracks 4,5, and 6 of disk drive 1.

```
0100    31 EE E6        1   WRITE   LXI   SP,0E6EEH  set up the stack
0103    AF              2           XRA   A          select drive A
0104    4F              3           MOV   C,A
0105    CD F3 E1        4           CALL  SELDRV
0108    CD F2 E1        5           CALL  TKZERO     recalibrate the hea
010B    01 00 7F        6           LXI   B,8000H-100H set initial adrs.
010E    CD 09 E2        7           CALL  SETDMA
0111    3E 04           8           MVI   A,4        initial track numbe
0113    32 4A 01        9   TLOOP   STA   TEMP       save track number
0116    4F              10          MOV   C,A        seek to correct trk
0117    CD CB E2        11          CALL  TRKSET
011A    01 01 1A        12          LXI   B,32:001Q  sector count&number
011D    C5              13  SLOOP   PUSH  B          save sect and count
011E    CD EA E1        14          CALL  DMAST      get current address
0121    21 00 01        15          LXI   H,100H     update to next sect
0124    09              16          DAD   B
0125    E5              17          PUSH  H          move address to B-C
0126    C1              18          POP   B
0127    CD 09 E2        19          CALL  SETDMA     set up new address
012A    C1              20          POP   B          restore sect cnt&nu
012B    C5              21          PUSH  B
012C    CD 76 E2        22          CALL  SETSEC     set up next sector
012F    CD 53 E2        23          CALL  DWRITE     write the data
0132    DA 47 01        24          JC    ERROR      test for error
0135    C1              25          POP   B          recover sect cnt&nu
0136    0C              26          INR   C          update sector
0137    05              27          DCR   B          update count
0138    C2 1D 01        28          JNZ   SLOOP
013B    3A 4A 01        29          LDA   TEMP       get current track
013E    3C              30          INR   A          update track
013F    FE 07           31          CPI   7          check if all done
0141    C2 13 01        32          JNZ   TLOOP      continue to next tr
0144    C3 44 01        33  DONE    JMP   DONE
0147    C3 47 01        34  ERROR   JMP   ERROR      error exit
014A    00              35  TEMP    DB    0          track storage
                        36
```

An assembled Disk Jockey 2D is part of a DISCUS 2 system and is also accompanied by a copy of Disk/ATE (tm). Both Disk/ATE and the Disk Jockey 2D CP/M are tailored to the I/O of the Disk Jockey 2D controller. Both expect that a serial TTY/RS-232 terminal is connected to J2 (serial port) of the Disk Jockey. Both are supplied on a write protected diskette (notch open) which should be kept that way. DO NOT COVER THE NOTCH ON THE DISKETTE. Finally, both systems are designed to self load when the disk is placed in drive A and a branch is made to 340:000Q (E000H). For CP/M users, the CP/M diskette is accompanied by a series of manuals describing how to back-up a CP/M diskette. The only precaution is that when drive B is to be used for the back-up, it must be "logged in" (e.g., DIR B:) before the back-up process begins.

Backing Up Disk/ATE

To make a back-up copy of Disk/ATE, load Disk/ATE and have a blank diskette which is not write protected (the notch should be covered). Follow the steps outlined below:

If You Have a Dual Drive System:

1) Perform steps 6 and 7 below, inserting the blank disk in drive B.

2) Type: TD A B
   TD is the transfer disk command with the source drive on the left and the destination drive on the right. This command will copy all the files on drive A to drive B.

If You Have a Single Drive System:

1) Type: B16
   This command forces ATE to express numbers and addresses in hexidecimal radix.

2) Type: L IO2DTBL <T>
   This command loads the I/O driver symbol table from the disk. After the symbol table is loaded, ATE will be able to search the table for variable values. Some variables will be necessary to accomplish the back-up.

3) Type: ? SYSIO.IOEND
   This is the standard way of interrogating ATE to find the value of variables. SYSIO is the beginning of the I/O driver, and IOEND is the end of the driver. Make a note of these two values because we will need them later.

4) Type: L ATETBL <T>
This loads a different symbol table from the disk and overwrites the previously known symbols.

5) Type: ? BEGIN.END
The two parameters typed back represent the extent of ATE except for disk buffers. Make a note of these two values also.

6) Type:
GO FMT128 to format single density
GO FMT256 to format double density with 256 byte sectors
GO FMT512 to format double density / 512 byte sectors
GO FMT1Ø24 to format double density / 1Ø24 byte sectors
This loads and executes the desired format program. The purpose of this routine is to write the IBM standard sector header and data marks out on the disk, and to put a bootstrap on track zero.

7) The selected format program prompts the user through the necessary steps to format a diskette and automatically returns to Disk/ATE when the operation is complete.

8) IO2D and ATE must now be saved on the new diskette. IO2D must be the first file on the disk, and ATE must be the second.

9) Using the values for SYSIO and IOEND obtained from step 3 above, Type: S IO2D (SYSIO value here)H.(IOEND value here)H The "H" suffix is necessary to force ATE to interpret the preceding number as a hexidecimal number.

1Ø) Using the values for BEGIN and END obtained in step 5, Type: S ATE (BEGIN value here)H.(END value here)H

11) Disk/ATE has now been copied on the fresh diskette. Files may now be transferred from the original diskette as required.

Backing up other files

Once IO and ATE have been backed up on a diskette, some of the other files on the original diskette from Thinker Toys may need to be moved onto the backup media. There are two types of files presently supported by Disk/ATE: Source and binary. The source files always load into the source area of Disk/ATE and may also be saved on another diskette from the same source area. The Disk/ATE user's manual describes this procedure in detail. When a binary file is first saved on a diskette by Disk/ATE, the starting address is recorded in the directory entry along with the length. The STAT command will display the starting address of a binary file along with its length. The length is given in

increments of 1024 bytes (k). Hence there are 2560 bytes in a file that is 2.5k long. When the file is loaded into memory, unless otherwise specified (see the Disk/ATE user's manual) it will be loaded starting at the address displayed by the STAT command and ending at the address which is the sum of the starting address and the file length. This file can be saved on another diskette (which has been previously formatted) by simply placing the diskette in the drive and typing a S(ave) command followed by the proper beginning and ending address. As an example, the binary file FMT1024 is 3.0k long and and has a starting address of 65:000Q (3500H). To save this file on a back-up diskette, the following steps need to be performed:

1) Put the disk that has FMT1024 on it in the drive.

2) Type: L FMT1024

3) Place the back-up diskette in the drive.

4) Type: S FMT1024 65:000Q..100:377Q

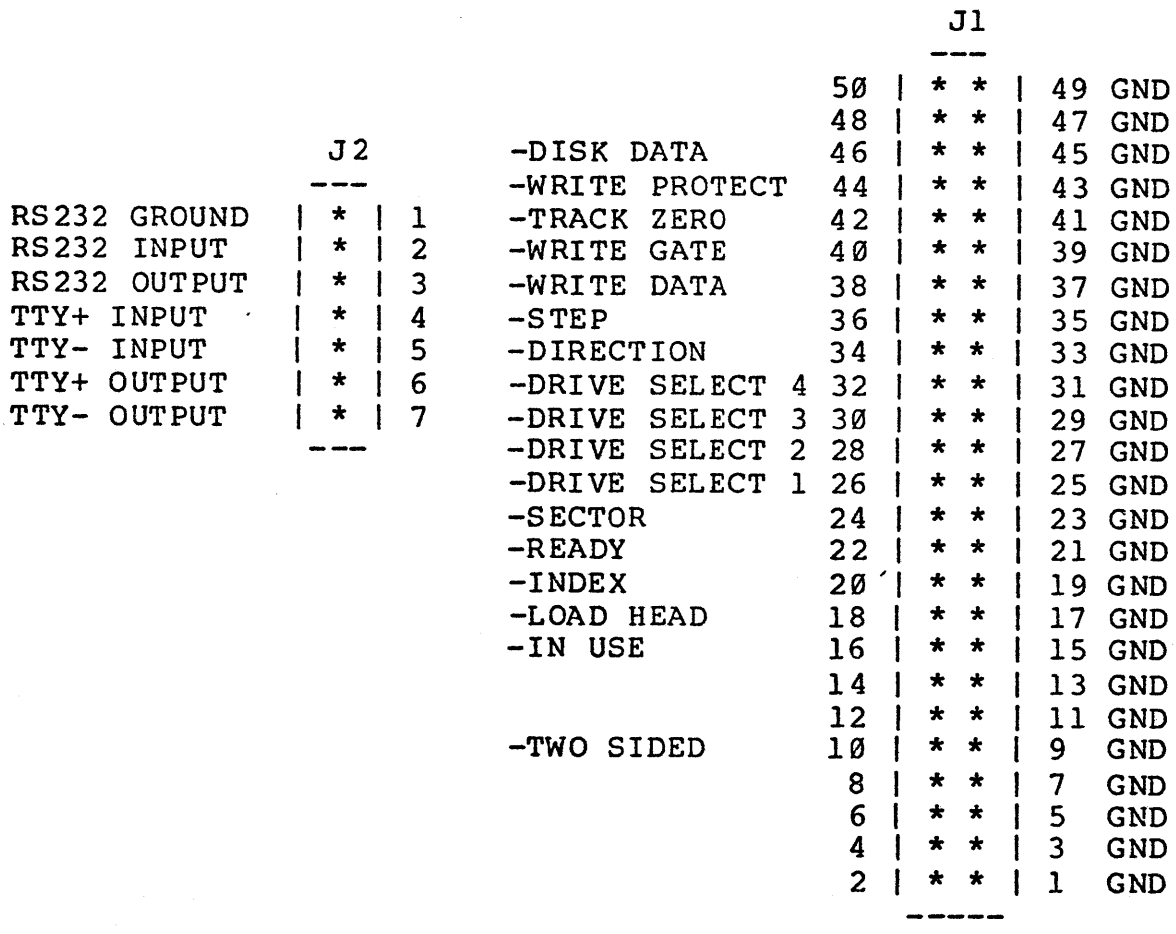
This completes the operation. If the write protect notch of the diskette is not covered on the back-up diskette, Disk/ATE will report a disk error and the operation will have to be done over with the notch covered.

The Bootstrap loader

Both Disk/ATE and copies of CP/M which are purchased through Thinker Toys are supplied on diskettes which load into the system through the use of the bootstrap loader DBOOT. To use DBOOT the system should be turned on and the CPU's program counter should be initialized to 340:000Q (E000H) either from the front panel of the computer or through jump-start logic either on the controller or on some other board in the system. A 2-3 second delay occurs the first time DBOOT is called after power-up so that the system has time to stabilize before the disk is accessed. Power should be applied to the drive(s) that are connected to the Disk Jockey controller at approximately the same time it is supplied to the CPU. However the system should be given time to stabilize before a diskette is inserted a drive. DBOOT always loads from drive A. If a diskette is not in place when DBOOT is started, the activity light at the front of drive A is slowly pulsed to indicate that the bootstrap loader is waiting for a diskette to be inserted in the drive and the door to be closed. The proper time to close the door is just AFTER the activity light has flashed. Shortly after the door is closed the drive signals the controller that it is ready and a loader program on sector one of track zero is read into the Disk Jockey RAM. When DBOOT is finished, it transfers control to this secondary loader.

I/O CONNECTORS J1 AND J2

Illustrated below are the details of the pin connections of J1 and J2. In both illustrations, the top of the circuit board is to the right of the drawing. The end pins of both connectors are numbered on the silk screen legend of the PC board. Note that all disk interface signals are active low.



PATCHES FOR CP/M*

General

This section is included for those users of the Disk Jockey 2D who have purchased a copy of CP/M Vers. 1.4 from a source OTHER than Thinker Toys. Copies of CP/M sold through Thinker Toys have the necessary I/O routines to interface CP/M to the Disk Jockey controller and to the DJ2D's serial I/O facility. These patches will help create a SINGLE DENSITY CP/M diskette--NOT a double density one. Though this may seem of marginal interest at first glance, we would point out that this section, combined with the software listings provided in the back of this manual, constitutes an excellent example of interfacing the Discus 2D to a significant disk operating system.

At the end of this section are two listings which are designed to allow the Disk Jockey to be interfaced with the Digital Research CP/M operating system. This can be done with a minimum of effort.

The first listing is the so called "cold start loader" which is used to bring CP/M in from the disk. It also has code which will allow the user easily to write a modified version of CP/M out on the disk. There is even a small routine which writes the "cold start loader" itself on sector 1 of track Ø.

The second listing is CBIOS software (Custom Basic Input-Output System) which is the interface between CP/M and the Disk Jockey controller. The general idea is to key in the cold start loader, use the loader to bring CP/M in from a diskette, enter the CBIOS code and, finally, use the cold start loader to save everything out on a clean diskette.

The "Cold Start Loader"

There are three parts to the cold start loader. LOAD is at address 347:ØØØQ (ØE7ØØH) and is designed to read CP/M into memory from location 51:ØØØQ (29ØØH) to 77:377Q (3FFFH). After loading CP/M, the LOAD routine branches to location 76:ØØØQ (3EØØH) which is a routine that initializes several memory locations, prints a sign-on message, and then branches to CP/M proper.

SAVE is at location 347:111Q (ØE749H) and is the reverse of LOAD. SAVE writes out on the disk starting at track Ø sector 2 all memory locations between 51:ØØØQ (29ØØH) and 77:377Q (3FFFH). After performing this operation, SAVE comes to a dynamic halt at STALL 347:133Q (ØE75BH).

*CP/M is a trademark of Digital Research

Patches for CP/M

INTLZ is a short routine which writes locations 347:000Q (0E700H) through 347:177Q (0E77FH) on sector 1 of track 0. Thus, once the cold start loader is keyed into memory, it can save itself at the right location on the disk.

CBIOS

The standard version of CP/M is designed to run with the Intel MDS development system and floppy disk interface. Most of the CP/M system software is completely independent of the particular 8080 hardware environment in which it happens to be running. However, there is a certain part which must be tailored to the hardware of the host system. This hardware dependent software is completely contained on pages 76 and 77 of CP/M memory (assuming the standard 16K version). CP/M can be made to run on different hardware by changing the software on pages 76 (3E00H) and 77 (3F00H). The CBIOS software which is supplied with the Disk Jockey is designed to let CP/M run when an eight inch full sized floppy disk is attached to the Disk Jockey controller that is plugged into an S-100 main frame.

Patching CP/M

Before actually performing any of the steps below, the Disk Jockey should be plugged into an S-100 bus mainframe, and an 8" disk drive should be connected to the controller. Be sure to observe correct cable orientation. You should have on hand two diskettes: one with CP/M and a blank one that has been formatted. A copy of CP/M which will run on the Disk Jockey will be constructed on the blank disk before any changes are attempted on the original CP/M disk. As a precaution, the diskette with the CP/M binary should have a write protect notch and this notch should NEVER be covered during the following steps.

Step I:

Plug in the controller. Connect the disk to the controller and turn on the the CPU and the disk drive. Do NOT put a diskette in the drive at this time.

Step II:

Be sure the drive is on and the door is OPEN. Initialize the CPU's program counter to 340:000Q and start the machine. After a several second delay, the LED at the top of the controller should turn on and the activity light (if one is present) on the front of the drive should flash briefly every several seconds. Various memory locations in the Disk Jockey RAM are now initialized and the firmware is ready to perform disk transfer operations. Stop the CPU.

Step III:

Enter the "cold start loader" into memory starting at location 347:000Q (0E700H). The instructions will extend from 347:000Q (E700H) to 347:177Q (0E77FH), filling most of the first half of the last page of RAM on the controller.

Step IV:

Set the program counter of the CPU to location 347:142Q (0E762H), but do NOT start the CPU yet.

Step V:

Insert the BLANK diskette into the drive and close the door. Be sure that the diskette is NOT write protected. (An 8" write protected diskette has a notch near the corner of the diskette diagonally oppoiste the labled corner.) If this notch is missing or covered, the diskette is not write protected. Be sure the diskette is inserted right side up. On a Disk Jockey system, the label will be on the top . The diskette is inserted in the drive with the label held bewteen the thumb and forefinger.

Step VI:

Start the computer. The drive activity light (if one is present) will come on, the head will load and step out to track 0 unless it is there already. After sixteen revolutions of the diskette, the head will unload and the activity light will go off.

Step VII:

Stop the CPU. It should be in the tight loop JMP DONE -- 303 171 347 octal (C3 79 E7 hex). The cold start loader has been written on sector 1 of track 0.

Step VIII:

Remove the diskette from the drive.

Step IX:

Change location 347:001Q (0E701H) from 000Q (00H) to 133Q (5BH) and change location 347:002 (0E702H) from 76Q (3EH) to 347Q (0E7H).

Step X:

Initialize the program counter of the CPU to 347:000Q (E700H) but do NOT start the machine.

Step XI:

Insert the CP/M diskette and be sure that the write protect notch is not covered. Close the door securely

Step XII:

Start the CPU. The head will load and after a second or two the head will step to track 1. Wait for the head to unload and the activity light to go off. CP/M has been loaded into memory between 51:000Q (2900H) and 77:377Q (3EFFH).

Step XIII:

Enter the CBIOS code starting at 76:000Q (3E00H). Be sure to check that the code has been entered correctly.

Step XIV:

Initialize the program counter of the CPU to 347:111Q (E749H) but do NOT start the CPU.

Step XV:

Take the diskette which has the cold start loader on track 0 sector 1 and place it in the drive. Be sure that this diskette is still write enabled (the notch should be covered).

Step XVI:

Start the CPU. The head should load, return to track 0 and write the better part of tracks 0 and 1 before it unloads. After the head unloads, remove the diskette and remove the write enable tab from the diskette. Stop the CPU. The CPU should be executing the JMP STALL instruction -- 303 133 347 octal (C3 5B E7 hex).

Step XVII:

Connect a terminal to the serial port of the Disk Jockey and adjust the baud rate, parity, stop bits, and word length of the terminal and controller so that they match.

Step XVIII:

Inspect the diskette which was removed in step XVI. Be sure that the write protect notch is NOT covered. Insert the diskette in the drive once again. Initialize the CPU's program counter to 340:000Q (E000H) and start the machine. After a few seconds the terminal should print:

16K CP/M VERS/1.4

After a few more seconds the prompt should appear:

A>

A Disk Jockey version of CP/M is now up and running. After this new version of CP/M has been tested (as documentated in the CP/M manual), Steps I through XVII can be used to alter the original CP/M diskette if desired.

HARDWARE LEVEL REGISTERS

Users desiring a greater level of control over the floppy disk or serial interface may wish to refer directly to the I/O device registers on the DJ from their 8080 or Z80 program. There are thirteen one-byte registers-- five of them read only, five write only and three read/write. The registers have eight memory addresses on the S-100 bus with a different register being selected during a read operation and a write operation when the addressed register is read only or write only.

The 1791 controller comprises one of the read only registers (status register), one write only register (command register), and all three of the read-write registers (track, sector, and data registers). The uses of these registers will be touched on only briefly here as there is included in the documentation a detailed data sheet describing the way in which the 1791 controller functions.

The 1602 UART comprises two of the read only registers (input data and status registers) and one of the write only registers (output data). As with the 1791, we do not describe these registers in great detail since a data sheet for the 1602 is also included in the documentation.

The 1791 controller has a negative logic data bus. For this reason the internal bidirectional data bus of the DJ board is also negative logic. However, the bus of the 1602 UART is positive logic. This means that when references are made to the UART registers, the signal levels are opposite to what one would normally expect. In practice then, one should always invert data just before it is written into the UART output register; likewise, data read from the UART should be inverted before it is interpreted.

READABLE REGISTERS

Register 0 - The inverted UART data output register
Location 343:370 (E3F8 hex) standard Disk Jockey:

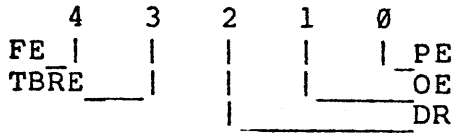
Date is stored in this register by the UART after it has been assembled from the serial data input stream. When a new character is assembled and transferred to this register, the UART sets the DR (Data Ready) flag. When this register is read by the CPU, the DR flag is reset by the UART hardware.

Register 1 - The inverted UART status register
Location 343:371 (E3F9 hex) standard Disk Jockey

Only the low order five bits of this register have any significance. The meaning of these bits is presented below. The 1602 data sheet should be referred to for a more detailed discussion of these bits. We shall list these signals using

their positive logic mnemonics with the understanding that the actual signals read will be the negation of these mnemonics.

INVERTED UART STATUS BITS

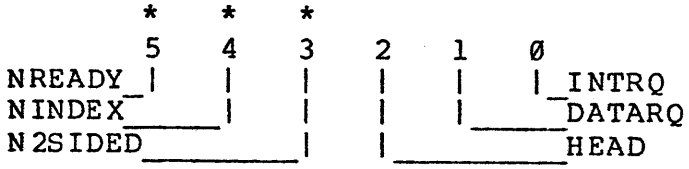


FE = Framing Error
TBRE = Transmitter Buffer Register Empty
DR = Data Ready
OE = Overrun Error
PE = Parity Error

REGISTER 2 - Disk Jockey status register
Location 343:372 (E3FA hex) standard Disk Jockey

This register contains bits that identify the current status of the Disk Jockey and the currently selected drive. Only the six low order bits have any significance in this register. The meanings of these bits are presented below:

DISK JOCKEY STATUS REGISTER



Bits marked with an asterisk reflect the current state of the status lines from the currently selected floppy disk drive. For a detailed specification of these signals see the documentation that accompanys the floppy disk drive. If no drive is currently selected or if the head is not loaded these bits are all high.

NREADY - This bit is a Ø when the currently selected drive is powered up with a diskette in place and the door closed.

NINDEX - This line reflects the status of the INDEX line from the floppy disk drive. It goes to a Ø once per revolution of the diskette.

N2SIDED- This line is a Ø when a double sided drive is connected to the controller AND there is a double sided diskette in place in the drive with the door closed.

Hardware level registers

HEAD - When this line is a 1 the head of the currently selected floppy disk drive is loaded.

DATARQ - When this line is a 1 the data request line from the 1791 controller is high and the controller is requesting that its data register be read from or written to. When the data register is referenced, this line will change to a Ø.

INTRQ - The 1791 controller sets this line to a one whenever it has completed a command and is no longer busy. This line is reset by a reference to the command register or the status register of the 1791 controller.

Register 3 - Not currently used
Location 343:373 (E3FB hex) standard Disk Jockey

Register 4 - 1791 controller status register
Location 343:374 (E3FC hex) standard Disk Jockey

This is the status register of the 1791 controller. The meaning of the bit patterns of this register varies depending upon the command that the controller is executing or has executed. See the 1791 data document for a detailed discussion of this register.

WRITE ONLY REGISTERS

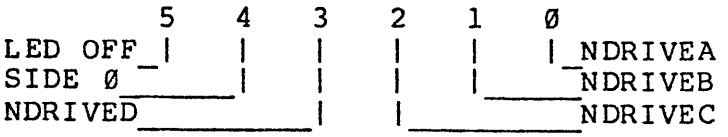
Register Ø - The inverted UART data input register
location 343:37Ø (E3F8 hex) standard Disk Jockey

Inverted data is stored is this register by the CPU for serial output by the UART. The UART transfers the data from this register to an internal parallel load serial output register where the start bit optional parity bit and the stop bits are appended to the data. Whenever the UART empties register Ø, the TBRE status bit is raised to inform the CPU that it is possible to output more data to the UART.

Register 1 - Disk Jockey drive control register
location 343:371 (E3F9 hex) standard Disk Jockey

This is a six bit register that is used by the Disk Jockey to select one of four drives, select side one or two for double sided drives, and to turn on and off the error flag LED built into the board near the serial connector J2. Only the low order six bits of this register have any significance. The meanings of these bits are presented below.

DRIVE CONTROL REGISTER



LED OFF - When a zero is stored in this bit the LED at the top of the board near J2 is turned on. A one stored in this bit turns off the LED.

SIDE Ø - When a double-sided drive is connected to the Disk Jockey a one stored in this bit selects head Ø while a zero selects head 1. When a single-sided drive is connected to the Disk Jockey, this bit has no effect on the drive.

NDRIVED - When this bit is a zero and the head is loaded the fourth or last drive is selected. A one written in this bit will deselect the last drive.

NDRIVEC - This is the drive select bit for the third drive connected to the Disk Jockey. A zero selects the third drive when the head is loaded while a one deselects the third drive.

NDRIVEB - The drive select bit for the second drive connected to the Disk Jockey. When the head is loaded, a zero in this bit will select the second drive while a one will deselect it.

NDRIVEA - The drive select bit for the first drive connected to the Disk Jockey. A zero in this bit will select the first drive when the head is loaded and a zero will deselect it.

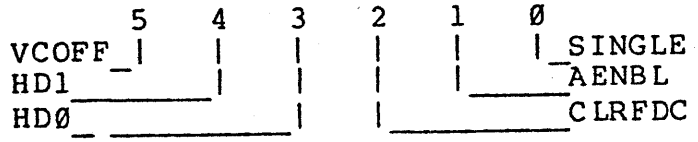
Only one of the four low order bits of this register should ever be a zero. If more than one of these bits are zero, loading the head will select more than one drive and cause data errors during reads and possible head position errors on seeks.

Register 2 - The Disk Jockey function register
Location 343:372 (E3FA hex) standard Disk Jockey

Only the low order six bits of this register have any significance. Two bits load and unload the read/write head of the drive, one determines the density mode that the 1791 controller operates at, another turns on and off the VCO of the phase-lock loop, and yet another controls the master reset of the 1791 controller. The final bit controls the way that the CPU will access the data register of the 1791. During power-up, this register is initialized so that it is as if ones had been written

in all six bits. The specific funciton of the various bits is detailed below.

DISK JOCKEY FUNCTION REGISTER



VCOFF - This bit controls the voltage controlled oscillator (VCO). A one written in this bit will turn the VCO off while a zero will turn the VCO on. The VCO must be on to read data from the disk.

CLRFDC - A one written in this bit will reset the 1791 controller. The chip will remain in the reset state until this bit is changed to a zero. When the reset signal is removed the 1791 executes a restore (seek to track zero).

AENBL - During data transfers, when the CPU references the 1791's data register the PREADY line (S-1ØØ bus line 72) is brought low which puts the processor in a wait state. The CPU remains in this state until the 1791 raises its DATA REQUEST line. This mode of operation dispenses with the usual status test during data transfers and makes it possible for the Disk Jockey to run at double density speeds without having to use a DMA channel. However there are times when the CPU needs access to this register when the DATA REQUEST is low (before a seek command is issued for example). When the AENBL bit is a one the stall logic that usually governs accesses to the 1791's data register is disabled. This allows the CPU to have access to this register as if it were a normal memory location. However, before the Disk Jockey can correctly transfer data to or from the floppy disk drive, this bit must be a zero so that the CPU can synchronize its data transfers to the 1791 controller.

SINGLE - When this bit is a one, the DJ board will read and write data to and from the disk in single density. When this bit is a zero, reads and writes are performed in double density.

Hardware level registers

HDØ,HD1 - These two bits control the loading of the read/write head. Their functional character is detailed in the table below.

HD1	HDØ	Read/write head function
Ø	Ø	not allowed
Ø	1	head is loaded
1	Ø	head is unloaded
1	1	1791 may unload head

Register 3 - Not currently used
Location 343:373 (E3FB hex) standard Disk Jockey

Register 4 - 1791 controller command register
Location 343:374 (E3FC hex) standard Disk Jockey

This is the command register of the 1791 controller. There are four different classes of commands and within each class there are a number of separate commands that the controller can execute. See the 1791 data document for a detailed discussion of this register and its use.

READ-WRITE REGISTERS

Register 5 - 1791 track register
Location 343:375 (E3FD hex) standard Disk Jockey

The 1791 controller uses this register as a reference to where the read/write head of the disk drive is positioned. Extreme care should be exercised when writing in this register. If care is not exercised, seek errors may likely occur. See the 1791 data document for a more detailed discussion.

Register 6 - 1791 sector register
Location 343:376 (E3FE hex) standard Disk Jockey

This is the sector register of the 1791 controller. Only one of the commands will cause the 1791 to write in this register. Generally the 1791 uses this register to determine which sector is to be read or written. See the 1791 data document for a more detailed discussion.

Register 7 - 1791 data register
Location 343:377 (E3FF hex) standard Disk Jockey

This is the data register of the 1791 controller. Data is written into this register when the controller is writing to the disk. Data is read from this register when the controller is

reading from the disk. The desired track number is also written in this register when seek commands are issued to the controller. As before the 1791 data document should be refered to for a more complete discussion

FINAL NOTE

The Disk Jockey firmware contains numerous examples illustrating the use of the hardware registers listed above. A comprehensive study of the two Western Digital data documents along with a careful examination of the Disk Jockey firmware will equip the interested user with enough knowledge to control the disk drive at the hardware level.

DJ2D REVISION 4 PARTS LIST

[ ]	1	5" x 10" printed circuit board
[ ]	1	240 Ohm 1/4 watt 5% resistor red-yellow-brown
[ ]	3	330 Ohm 1/4 watt 5% resistors orange-orange-brown
[ ]	2	470 Ohm 1/4 watt 5% resistors yellow-purple-brown
[ ]	2	750 Ohm 1/2 watt 5% resistors purple-green-brown
[ ]	12	1k Ohm 1/4 watt 5% resistors brown-black-red
[ ]	1	1.5k Ohm 1/4 watt 5% resistor brown-green-red
[ ]	3	3.3k Ohm 1/4 watt 5% resistors orange-orange-red
[ ]	3	4.7k Ohm 1/4 watt 5% resistors yellow-purple-red
[ ]	2	5.36k Ohm 1/8 watt 1% resistors - These two parts replace the 6.19k 1% resistors which appear on the parts legend of the circuit board.
[ ]	2	10k Ohm 1/4 watt 5% resistors brown-black-orange
[ ]	1	11.0k Ohm 1/8 watt 1% resistor - This part replaces 13.0k 1% resistor which appears on the parts legend of the circuit board.
[ ]	2	27k Ohm 1/4 watt 5% resistors red-purple-orange
[ ]	1	47k Ohm 1/4 watt 5% resistors yellow-purple-orange
[ ]	4	1 Megohm 1/4 watt 5% resistors brown-black-green
[ ]	1	180 Ohm 1/8 watt 5% 9 resistor SIP array
[ ]	2	3.3k Ohm 1/8 watt 5% 9 resistor SIP array
[ ]	3	33 picofarad 5% silver mica capacitors - Two of these parts replace the 100 picofarad capacitors which appear on the parts legend of the circuit board.
[ ]	2	47 picofarad 2% or 1% silver mica capacitors
[ ]	1	112 picofarad 2% or 1% silver mica capacitor
[ ]	1	470 picofarad 5% silver mica capacitor
[ ]	1	.001 microfarad disk capacitor
[ ]	1	.01 microfarad mylar capacitor

DJ2D Revision 4 Parts List

[ ]	1	1.5 microfarad dipped tantalum capacitor	
[ ]	5	1.8 microfarad axial lead tantalum capacitors	
[ ]	2	39 microfarad axial lead tantalum capacitors	
[ ]	19	Disk by-pass capacitors - may vary in value from .01 to .1 microfarads depending on current supplies	
[ ]	1	Dual-in-line 50 conductor right angle header	
[ ]	1	Single-in-line 7 conductor right angle header	
[ ]	2	Heat sinks for 5 volt regulators	
[ ]	4	6-32 5/16 flat head machine screws	
[ ]	4	6-32 1/4 hex machine nuts	
[ ]	1	5.0688 MHz HU/18 Crystal	
[ ]	1	10.0000 MHz HU/18 Crystal	
[ ]	2	8 position DIP switch arrays	4D,13D
[ ]	4	1N914/4820-0201 signal diodes	
[ ]	1	1N751A 5.1 volt 5% Zener diode	
[ ]	1	RL209 light emitting diode	
[ ]	2	2N3904 transistors	
[ ]	2	2N3906 transistors	
[ ]	1	8 pin low-profile socket	
[ ]	15	14 pin low-profile sockets	
[ ]	16	16 pin low-profile sockets	
[ ]	5	18 pin low-profile sockets	
[ ]	4	20 pin low-profile sockets	
[ ]	2	40 pin low-profile sockets	
[ ]	2	74LS00 quad 2-input NAND gate	3D
[ ]	1	74LS02 quad 2-input NOR gate	4C
[ ]	1	7404 hex inverter	2B

DJ2D Revision 4 Parts List

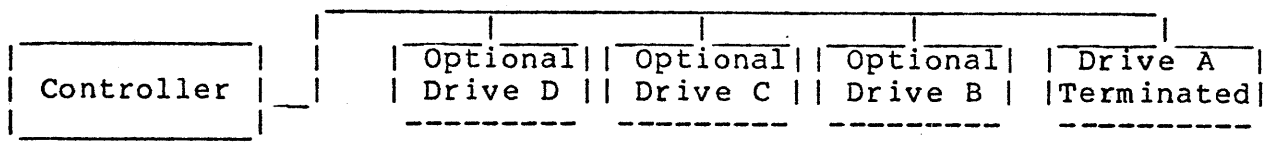
[ ]	2	74LS04/LS14 hex inverter	7B,10B
[ ]	1	74LS08 quad 2-input AND gate	1D
[ ]	1	74LS13/20 dual 4-input NAND gate	8B
[ ]	1	74LS30 8-input NAND gate	7C
[ ]	2	74LS32 quad 2-input OR gate	7A,6C
[ ]	3	74LS74 dual D type flip-flop	3B,9B,2C
[ ]	1	74LS132 quad 2-input NAND Schmitt Trigger	3B
[ ]	1	74LS155 dual 1 of 4 decoder	9A
[ ]	1	74LS161 hexidecimal counter	1B
[ ]	1	74165/74LS165 8 bit parallel load shift register	2D
[ ]	2	74LS174 hex register with clear	12C,13C
[ ]	2	74LS221 dual monostable - These two parts replace the 74221 IC which appears on the silk screened legend of the circuit board.	4B,6B
[ ]	2	74LS240 octal tri-state buffer	9D,10D
[ ]	2	74LS240/244 tri-state buffer	5D,6D
[ ]	3	74LS365/74LS367 hex tri-state buffer	10A,7D,8D
[ ]	1	74LS366/74LS368 hex tri-state inverter buffer	12B
[ ]	2	74366/74368 hex tri-state inverter buffer	11B,13B
[ ]	1	74390/74LS390 dual decade counter	1C
[ ]	1	MM6300/6301/82S129/74S287 4 x 256 PROM	5C
[ ]	1	MMI6331/74LS288 8 x 32 PROM	8A
[ ]	2	MM6353/82S137/PB426 4 X 1024 PROM	8C,11C
[ ]	2	2114-3L 4 X 1024 low power 300NS static RAM	9C,10C
[ ]	1	BR1941/2941/COM5016 dual baud rate generator	12D
[ ]	1	TR1602 UART	13D
[ ]	1	FD1791 dual density floppy disk controller	13A

DJ2D Revision 4 Parts List

[ ]	1	1458/4558 dual op-amp	4A
[ ]	2	78Ø5 monolithic 5 volt regulator	
[ ]	1	7812 monolithic 12 volt regulator	
[ ]	1	7912 monolithic -12 volt regulator	

CABLE CONNECTIONS

Drives on Discus systems are connected in daisy chain fashion to the controller board, as illustrated below.



As can be seen from the above figure, Drive A is located at one end of the cable and is the only terminated drive on the cable. The location of any additional drives on the cable is not important as long as they are not at the end of the cable. Again, extra drives are not terminated.

Aside from termination, the only physical difference between an "A" and a "B" drive, or between any two differently addressed drives, is the jumper strapping on the PC board of the drives. Strapping a drive for termination and drive selection is documented in the Shugart OEM manual.

Four different daisy chain cables are available for one, two, three or four drive systems. A daisy chain cable is simply a parallel cable. Not all available connectors on a multiple drive cable need be filled for the system to function. Also, a dual system with drives addressed, say, as "A" and "C" would work fine as long as the operator remembered to refer to the second drive as "C" rather than "B". In other words, the absence of a "B" drive in no way "locks out" the "C" and "D" drives.

The following rule applies to all cable configurations supplied by Thinker Toys:

> The 50 pin flat ribbon cable provided with the Discus system should be connected to the Disk Jockey controller board so that the cable extends out over the solder side of the PC board-- not the component side.
>
> Whichever end of the 50 pin flat ribbon cable is chosen to plug into the controller board, that side of the cable which is on the LEFT (closer to the heat sink) as it connects to the controller should be UP as it connects to each and every drive on the system. Thus, J1 pin 50 on the DJ controller board should come in to each disk drive via the top part of the male 50 pin connector attached to the cabinet of each drive. If the LED on the front of the drive comes on upon power up, the cable is on backwards and should be reversed. The LED on the front of the drive should light up only when a command has been issued to load the head.
>
> Any visual "key" such as an arrow or triangle on a connector should be used solely as an aid in implementing the connection scheme described above.

BAUD RATE SELECTION

Paddles 1 to 4 of Switch 2 in the lower right corner of the DJ control the baud rate for the 1602 UART. Sixteen separate baud rates, ranging from 50 to 19,200, are available. The following table lists all possible switch settings for baud rate selection.

BAUD RATE SWITCH SETTINGS

SW2-1	SW2-2	SW2-3	SW2-4	BAUD RATE
on	on	on	on	50
on	on	on	off	75
on	on	off	on	110
on	on	off	off	134.5
on	off	on	on	150
on	off	on	off	300
on	off	off	on	600
on	off	off	off	1200
off	on	on	on	1800
off	on	on	off	2000
off	on	off	on	2400
off	on	off	off	3600
off	off	on	on	4800
off	off	on	off	7200
off	off	off	on	9600
off	off	off	off	19200

WORD LENGTH

Paddle 7 of Switch 2 controls data word length selection for the 1602 UART. Placing paddle 7 in the "on" position sets the word length to 7 bits, while "off" fixes the word length to 8 bits. The table below gives the word length selection settings for the DJ.

WORD LENGTH SELECTION

SW2-7	WORD LENGTH
"on"	7 BITS
"off"	8 BITS

STOP BIT COUNT

SW2-5 controls the number of stop bits, either one or two, which the UART sends after each data word. The "off" position will set the device to two stop bits, and the "on" position to one.

Most devices are extremely tolerant concerning stop bit setting. As a general rule, if a device fails to communicate with the Disk Jockey, it is not because the stop bit setting is incorrect.

STOP BIT COUNT SELECTION

SW2-5	STOP BIT COUNT
"on"	1 STOP BIT
"off"	2 STOP BITS

PARITY

If paddle 6 of switch 2 is in the "off" position, the UART will not generate any parity bits at the end of the serial data word. If the paddle is in the "on" position, refer to the table below for the proper parity setting via paddle 8.

PARITY SWITCH SETTING

SW2-8	PARITY
"on"	ODD PARITY
"off"	EVEN PARITY

POWER-ON JUMP TABLE WITH 74LS240'S AT 5D AND 6D

(REV 3 BOARDS SHOULD USE 244'S ONLY)

SET PADDLE 6 OF SW1 TO "off" FOR 74LS240'S

SET PADDLE 7 OF SW1 TO "on" TO ENABLE POWER-ON JUMP

(SW1 is the switch to the LEFT)

JUMP ADDRESS		SWITCH SETTING				
Octal	Hex	SW1-1 (A15)	SW1-2 (A14)	SW1-3 (A13)	SW1-4 (A12)	SW1-5 (A11)
000:000	0000	off	off	off	off	off
010:000	0800	off	off	off	off	on
020:000	1000	off	off	off	on	off
030:000	1800	off	off	off	on	on
040:000	2000	off	off	on	off	off
050:000	2800	off	off	on	off	on
060:000	3000	off	off	on	on	off
070:000	3800	off	off	on	on	on
100:000	4000	off	on	off	off	off
110:000	4800	off	on	off	off	on
120:000	5000	off	on	off	on	off
130:000	5800	off	on	off	on	on
140:000	6000	off	on	on	off	off
150:000	6800	off	on	on	off	on
160:000	7000	off	on	on	on	off
170:000	7800	off	on	on	on	on
200:000	8000	on	off	off	off	off
210:000	8800	on	off	off	off	on
220:000	9000	on	off	off	on	off
230:000	9800	on	off	off	on	on
240:000	A000	on	on	on	off	off
250:000	A800	on	off	on	off	on
260:000	B000	on	off	on	on	off
270:000	B800	on	off	on	on	on
300:000	C000	on	on	off	off	off
310:000	C800	on	on	off	off	on
320:000	D000	on	on	off	on	off
330:000	D800	on	on	off	on	on
340:000	E000	on	on	on	off	off
350:000	E800	on	on	on	off	on
360:000	F000	on	on	on	on	off
370:000	F800	on	on	on	on	on

POWER-ON JUMP TABLE WITH 74LS244'S AT 5D AND 6D

(REV 3 BOARDS SHOULD USE 244's ONLY)

SET PADDLE 6 OF SW1 TO "on" FOR 74LS244'S

SET PADDLE 7 OF SW1 TO "on" TO ENABLE POWER-ON JUMP

(SW1 is the switch on the LEFT)

JUMP ADDRESS		SWITCH SETTING				
Octal	Hex	SW1-1 (A15)	SW1-2 (A14)	SW1-3 (A13)	SW1-4 (A12)	SW1-5 (A11)
000:000	0000	on	on	on	on	on
010:000	0800	on	on	on	on	off
020:000	1000	on	on	on	off	on
030:000	1800	on	on	on	off	off
040:000	2000	on	on	off	on	on
050:000	2800	on	on	off	on	off
060:000	3000	on	on	off	off	on
070:000	3800	on	on	off	off	off
100:000	4000	on	off	on	on	on
110:000	4800	on	off	on	on	off
120:000	5000	on	off	on	off	on
130:000	5800	on	off	on	off	off
140:000	6000	on	off	off	on	on
150:000	6800	on	off	off	on	off
160:000	7000	on	off	off	off	on
170:000	7800	on	off	off	off	off
200:000	8000	off	on	on	on	on
210:000	8800	off	on	on	on	off
220:000	9000	off	on	on	off	on
230:000	9800	off	on	on	off	off
240:000	A000	off	on	off	on	on
250:000	A800	off	on	off	on	off
260:000	B000	off	on	off	off	on
270:000	B800	off	on	off	off	off
300:000	C000	off	off	on	on	on
310:000	C800	off	off	on	on	off
320:000	D000	off	off	on	off	on
330:000	D800	off	off	on	off	off
340:000	E000	off	off	off	on	on
350:000	E800	off	off	off	on	off
360:000	F000	off	off	off	off	on
370:000	F800	off	off	off	off	off

BOOT LED

Near the upper left corner of the DJ2D board, just to the right of terminal connector J2, is the boot LED. This LED will flash on and off if the DBOOT routine reports an error. Since the boot routine is not affected by terminal I/O, this LED can help in determining whether a no-go attempt at bringing up an operating system is due to faulty I/O hardware and/or drivers or due to some other cause-- memory, media, controller, CPU etc.

PHANTOM ENABLE

The DJ2D will respond to the PHANTOM line-- S-100 pin 67-- if paddle 8 of SW4 is placed in the 'on' position. This paddle is the lowest paddle of the LEFT switch, at location 4D. The DJ2D will become de-selected when the PHANTOM line goes active if this paddle is 'on'. If this paddle is placed in the 'off' position, the DJ2D will ignore the PHANTOM line. In order for the Power-on Jump feature of the DJ2D to work on a SOL computer, the PHANTOM Enable Switch must be 'on'.

POWER STABILIZATION

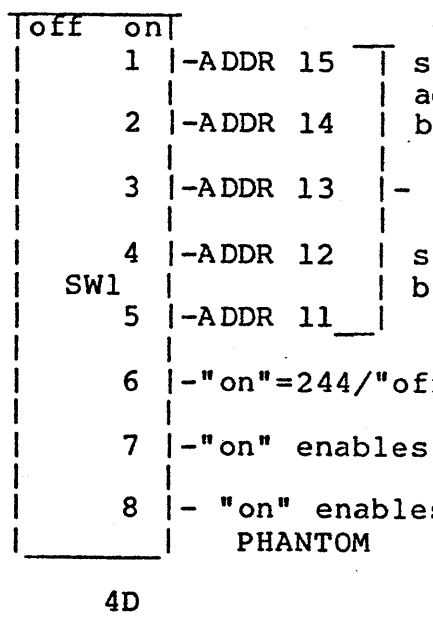
When booting a disk for the first time after powering up, the head on Drive A will not load (as evidenced by the LED on the drive door release) for a second or two. After this initial boot, all subsequent boots should load the head immediately until power is turned off and on (erasing memory). During a boot, the firmware on the DJ2D searches its internal RAM for a bit pattern to indicate that at least one boot has taken place since power up. If no such bit pattern is present, a short delay will be inserted to allow all components in the system to stabilize.

BOOTING WITHOUT A DISKETTE

If no diskette has been placed in Drive A and a boot is attempted (as is often the case during a power-on-jump when a system is first powered up), the LED on Drive A will flash on briefly about once every second. It is possible to execute a boot in this mode. Insert the system diskette into Drive A. Do not lower the drive door, but push the diskette into the drive far enough so that it locks into place (the higher the drive door, the easier for the diskette to lock into place). Wait for the red LED on the diskette release button to flash on and off and, when it goes off, close the drive door. The diskette will boot the next time the LED goes on.

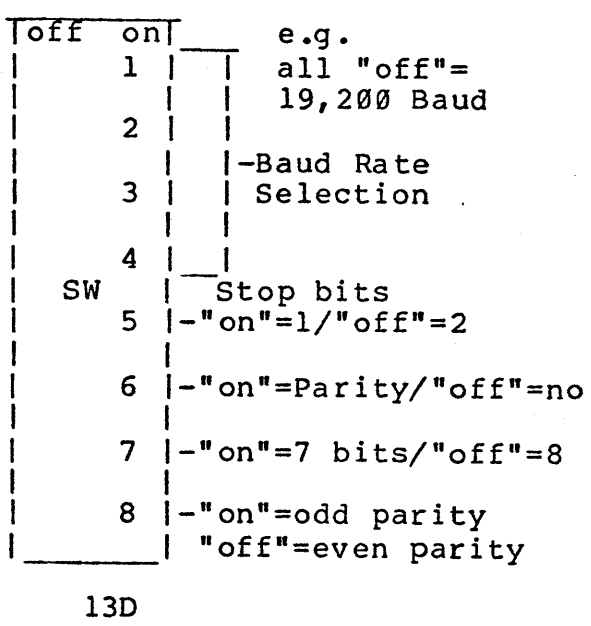
FAST REFERENCE FOR DJ2D DIP SWITCHES

Power-on-jump Switch



4D

UART Switch



13D

(Setting for some paddles on SW1 at 4D depend upon whether 74LS240's or 74LS244's are used in locations 5D and 6D.)

ALL OFF

	OFF	ON
1		X
2	X	
3		X
4	X	
5		X
6	X	
7	X	
8	X	

ASSEMBLY INSTRUCTIONS

WARNING! IMPROPER ASSEMBLY OF THIS KIT WILL VOID THE WARRANTY. READ THESE INSTRUCTIONS CAREFULLY BEFORE ATTEMPTING TO CONSTRUCT THIS KIT

INVENTORY

Make sure that all parts listed in the Parts List have been included. Notify Thinker Toys immediately if any are missing. Also, quickly return all extra parts.

USE BENDING BOARD

With the exception of the axial tantalum capacitors and the 1/2 watt 750 Ohm resistors, all the resistor and diode leads should be bent to .5 inches. The leads of the 750 Ohm resistors should have a spacing of .6 inches. The axial lead tantalum capacitors should be bent to .7 inches. Use of a bending block will give your finished kit a more professional look.

USE SOCKETS

Sockets are provided for every IC on the Disk Jockey.

NO REPAIR WORK WILL BE ATTEMPTED ON ANY RETURNED BOARD WITH ANY IC SOLDERED DIRECTLY TO THE CARD

ORIENTATION

When this manual refers to the bottom of the circuit board it means the side with the gold S-100 edge connectors. Right and left assume a view from the component side of the board which has the silk screen legend.

All IC sockets will either have their pins numbered or have a 45 degree angle across the corner of pin one. On the Disk Jockey, all sockets and all IC's have pin 1 closest to the bottom right corner of the board.

The tantalum capacitors are polarized. The dipped tantalum cap has a red dot at its positive lead. This lead should be inserted at the bottom of the oval legend where the "+" sign is located. The 1.8 microfarad capacitor's positive lead is identified by a circular "tit" where it enters the body of the housing. The positive end of the 39 microfarad capacitors is identified by a red band. The silk screen identifies the positive lead of these axial parts with a "+" sign. The by-pass caps, identified on the silk screened legend by an asterisk "*" enclosed by an oval, are not polarized. The .Ø1 mylar cap and the .ØØ1 disk cap are not polarized.

The two DIP switch arrays are to be positioned so that switch paddle number 1 is toward the top of the board.

The SIP resistor packs, historically prone to being inserted backwards, should have their white dot nearest the white dot on their respective legends. This turns out to be down for the two 3.3k Ohm packs at the bottom of the board and to the right for the 18Ø Ohm pack just below the J1 connector at the top right of the board.

The crystals included in this kit have a piece of foam pad attached to their PC board side. When these parts are installed, the protective paper on the back of the pad should be peeled off just before the leads are inserted through the circuit board at the position indicated on the parts legend. The foam pad has an adhesive on it which will hold the crystal to the circuit board. The pad and the adhesive are insulators so that no short circuit can occur when the crystal is installed.

The orientation of the transistors is indicated on the silk screen legend of the circuit board, as is their type number. A very common cause of smoke on power-up is a 2N39Ø6 correctly oriented in the place of a 2N39Ø4 and vice versa.

The black band at one end of the diodes marks the cathode and should correspond to the white arrow point on the legend of the circuit board.

Placing the 5Ø pin flat cable connector, J1, upside down is a disaster. The angled pins should go through the circuit board. Only the longer straight pins are long enough to accept the ribbon cable to the disk drive. The I/O connector, J2, should be positioned so that the longer angled pins point toward the top of the board while the shorter straight pins go through the circuit board.

EXAMINE THE BOARD

Visually examine the circuit board for any trace opens or shorts. A concentrated five minute scrutiny will uncover most trace defects. Several hours of scattered, unconcentrated scrutiny generally won't reveal anything. Take special care that no shorts or opens exist on those areas of the circuit board that will be covered by IC sockets. Ohm out any suspicious looking traces for either shorts or discontinuity as appropriate. Return immediately any bare board found to be flawed. Such boards will be replaced under warranty.

SOLDERING AND SOLDER IRONS

The most desirable soldering tool for complex electronic kits is a constant temperature iron with an element regulated at 650 degrees F. The tip should be fine so that it can be brought into close contact with the pads of the circuit board. Such irons are available from Weller and Unger and should be part of any electronics shop.

There are three important soldering requirements for building this kit:

1. Do not use an iron that is too cold (less than 600 degrees F) or too hot (more than 750 degrees F).
2. Do not hold the iron against a pad for more than about six seconds.
3. Do not apply excessive amounts of solder.

The recommended procedure for soldering components to the circuit board is as follows:

1. Bring the iron in contact with BOTH the component lead AND the pad.
2. Apply a SMALL amount of solder at the point where the iron, component lead, and pad ALL make contact.
3. After the initial application of solder has been accomplished with the solder flowing to the pad and component lead, the heat of the iron will have transferred to BOTH the pad AND lead. Apply a small amount of additional solder to cover the joint between the pad and the lead.

DO NOT PILE SOLDER ON THE JOINT! EXCESSIVE HEAT AND SOLDER CAUSE PADS AND LEADS TO LIFT FROM THE CIRCUIT BOARD. EXCESSIVE SOLDER IS THE PRIMARY CAUSE FOR BOARD SHORTS AND BRIDGED CONNECTIONS.

[ ] Install and solder the four signal diodes (1N914 or equivalent) and clip the excess leads from the parts. Be sure that the black bands of the diodes are positioned to match the arrow points of the white legend of the circuit board

PROTECT YOUR EYES WHEN YOU CLIP COMPONENT LEADS AFTER SOLDERING

[ ] Install and solder all the 1/4 watt resistors in place. Do this in sections so that the leads can be conveniently clipped.

[ ] Install, solder, and trim the leads of the 1% precision resistors.

[ ] Install, solder, and trim the lead of the 1N751A Zener diode. Be sure that the black band of the diode is to the left as indicated by the white arrow point.

[ ] Next, install, solder and trim the leads of the 750 Ohm 1/2 watt resistors.

[ ] Install and solder the 40 pin sockets first, then the 20, 18, 16, and 14 pin sockets in that order. Finally install and solder the 8 pin socket. By installing the sockets in this order, a smaller sized socket will never be placed in a larger sized position.

[ ] Install and solder the SIP resistor pack arrays. The top pack should have its white dot to the right while the bottom packs will have their white orientation dots to the bottom of the circuit board.

[ ] Install and solder the 5 axial lead 1.8 microfarad capicators. The top two have their "+" leads to the right while the bottom three have their "+" leads to the left. Clip the excess leads from the parts.

[ ] Install, solder, and clip the leads of the two 39 microfarad caps. The red band of these parts must point to the right.

[ ] Bend the leads of the 7812 and 7912 regulators, skipping the 7805's for now. Placing a nut on top of the regulator, insert a screw from the bottom of the circuit board through the hole of the board and through the hole of the regulator. Hand tighten the nut. Solder the leads. Tighten the screws firmly.

[ ] After bending the leads 90 degrees, install and solder the two crystals in place. Clip the excess leads. Fix them to the circuit board by peeling the protective paper off their foam pad and pressing the pad against the board. Be sure to solder the crystals into place so that their padded side will fall into the area outlined on the silk screened legend.

[ ] Install and solder the two connectors J1 and J2. Be sure to reread the orientation section before installing these parts.

[ ] Install and solder the light emitting diode at the top of the board just to the right of J2. One of the leads of this diode is longer than the other. The longer lead is the anode and must be to the left when the part is inserted. Clip the excess leads after soldering.

[ ] Install, solder and clip the leads of the 1.5 dipped tantalum cap just below J2. Be sure that the lead with the red dot is pointed toward the bottom of the circuit board.

[ ] Install, solder and clip the 33 picofarad silver mica cap just to the left of the 10 Meg crystal in the upper left corner of the board.

[ ] Install, solder and clip the two 47 picofarad silver mica caps above and below the 74221 IC at location 4B.

[ ] Install, solder and clip the two 33 picofarad silver mica caps-- one below the 74LS165 IC at location 2D and the other between the 74LS244 IC (labeled 74LS240 on the silk screened legend) at 6D and the 74LS367 at 7D.

[ ] Install, solder and clip the 112 picofarad silver mica cap beneath the 74221 IC at location 6B.

[ ] Install, solder and clip the 470 picofarad mica cap beneath the 7404 IC at location 6B.

[ ] Install, solder and clip the .001 microfarad disk cap to the left of the 74LS74 IC at location 6A.

[ ] Install, solder and clip the .01 microfarad mylar cap to the left of the 1458/4558 IC at location 4A.

[ ] Install, solder and clip the leads of the three transistors near J2, and of the 3904 transistor below DIP switch 4D, carefully observing the placement and orientation information silk screened on the circuit board.

[ ] Install and solder the two DIP switch arrays. Switch 1 of each DIP should be positioned toward the top of the board.

Parts Installation

[ ] Install, solder, and clip the leads of the 19 by-pass capacitors whose positions are identified by an oval with an asterisk "*" in the middle.

[ ] Bend the leads of the two 7805 regulators and insert them in the circuit board. Place a separate, finned heat sink between the regulator and the board, work a screw from the back of the board through the board, heat sink, and regulator and hand tighten into the nut on top of the regulator. Solder the leads and adjust the wings of the separate heat sink and, finally, tighten the screw.

CLEAN AND EXAMINE THE BOARD

Use flux cleaner to remove solder rosin residue. Examine the circuit board carefully for shorts, solder bridges, or missed pins.

HOW TO FIND WHERE TO PLACE PARTS

For parts placement, please see the silk screened legend on the printed circuit board.

When placing IC's in their sockets (which you should NOT do at this time!), be aware of the following deviations from or options to the IC numbers marked on the silk screened legend:

--Where the silk screened legend calls for a "6.19" K resistor, use a 5.36K precision resistor.

--Where the silk screened legend calls for a "13.0" K resistor, use an 11.0K precesion resistor.

--Though the silk screened legend says "100P" for the lower two silver mica caps, 33 picofarad caps should be used at these two locations.

--Though the silk screened legend says "LS240" at 5D and 6D, 74LS244's may also be used here. Only 240's should be used at 9D and 10D.

--Only a 7404 should be placed in location B3, as indicated in the legend. An "LS" part, either a 74LS04 or a 74LS14, must not be substituted.

--Location 3C, which which is marked "LS00", should have a 74LS132-- NOT a 74LS00.

With the exception of those parts listed above, IC's may vary from those marked on the silk screened legend if they are listed as alternate IC's (following a slash) in the Parts List on pages 29-31.

DO NOT INSERT ANY IC'S IN THEIR SOCKETS AT THIS TIME

INITIAL CHECK-OUT AND POWER-UP

Before inserting any IC's in their sockets perform the following check-out procedure:

1. Re-check the back of the board for solder shorts and bridged connections and for pins of IC sockets that have not been soldered. These unsoldered pins can cause aggravating intermittant probems during check-out.

2. Re-check components for orientation and make sure all components to be soldered have been soldered.

3. With an ohm meter, check for shorts between all regulated voltages (+5V,-5V,+12V,-12V) and ground and between any two regulator outputs (all regulator output pins are on the right side of the regulator, towards the bottom of the circuit board in this case). Check for shorts between S-100 supply voltages (+8V,+16V, -16V) and ground. S-100 pins 1 and 51 hold 8 volts, pin 2 holds +16 volts, and pin 52 -16 volts. Ground is on S-100 pins 50 and 100. Check these voltages for shorts amoung each other.

4. Place the board WITHOUT IC's into an empty system bus slot and power up. In case of smoke, power down immediately and investigate.

5. With a VOM or scope, check the regulators for +5V (both of the 7805's), +12V, and -12V. The bottom pin of all four regulators is the output. Check for Vcc and ground on all IC's. Check for +12V on the 1791 controller, the 2941 baud rate generator, and the 1458/4558 op amp. Check for -12V on the 1602 UART and the 1458/4558 op amp. Finally, check for -5V on the 2941 baud rate generator. If everything is OK, power down and proceed to the next step.

IC INSERTION

If an IC insertion tool is not available, IC leads should be straightened a ROW at a time, not by the individual PIN. The edge of a straight sided table is an excellent device for this operation. Hold the IC by the plastic case, place one row of legs against a flat surface and push very slightly. Repeat with the opposite row. Continue this procedure until the legs of the IC can be inserted with minimum effort into its socket.

When inserting an IC into its socket, take care that you DO NOT BEND THE IC'S LEGS UNDERNEATH ITS PLASTIC PACK. This is an extremely common error and can escape even a fairly careful visual inspection.

Parts Installation

If IC pins become bent under during insertion, use a long nose pliers to straighten them and try again. When removing an IC from its socket, use an IC remover, an IC test clip (another must for any electronics shop) or a miniature screw driver. DO NOT ATTEMPT TO REMOVE AN IC WITH YOUR FINGERS. You will bleed on severely bent pins.

Once all IC's have been inserted, re-check for bent pins. Then check twice for proper orientation. Upside down IC's are generally destroyed upon power up.

IF FOR ANY REASON IT BECOMES NECESSARY TO REMOVE A COMPONENT WHICH HAS BEEN SOLDERED TO THE CIRCUIT BOARD, CLIP ALL LEADS BEFORE REMOVING. THIS WILL REDUCE THE CHANCE OF LIFTING PADS OFF TRACES.

POWER UP

If all previous checks have been performed, you are ready to put power to your fully populated board. In an empty system with power off, insert the Disk Jockey and power up. If the board smokes, power down and investigate. If not, measure the regulated voltages again.

If any voltages have been lost since powering up the bare board, power down and check for upside down IC's. Isolate the possible faulty chip or chips by powering down, removing a section of IC's, and powering up again. Continue this sequence until the faulty IC or IC's are found.

BE SURE NEVER TO INSERT OR REMOVE A BOARD WITH POWER ON! THIS MAY DAMAGE THE BOARD

This completes the initial check-out of your Disk Jockey.

DJ2D REV4 MEMORY MAP

HEX ADDRESS	FUNCTION		OCTAL ADDRESS
E000-E3F7	ROM FIRMWARE		340:000-343:367
	I/O REGISTERS		
	WHEN READ	WHEN WRITTEN	
E3F8	UART INVERTED DATA INPUT	UART INVERTED DATA OUTPUT	343:370
E3F9	UART INVERTED STATUS	DISK JOCKEY FUNCTION	343:371
E3FA	DISK JOCKEY STATUS	DRIVE CONTROL REGISTER	343:372
E3FB	NOT USED		343:373
E3FC	1791 CONTROLLER STATUS	1791 CONTROLLER COMMAND	343:374
E3FD	1791 TRACK REGISTER		343:375
E3FE	1791 SECTOR REGISTER		343:376
E3FF	1791 DATA REGISTER		343:377
E400-E7FF	RAM		344:000-347:377

SOFTWARE LISTINGS

```
                ;*****************************************************
                ;*                                                   *
                ;* CBIOS DRIVERS FOR CPM                             *
                ;*                                                   *
                ;* Currently the cbios is set up for a 16K cpm, to make a *
                ;* larger system, change the value of CPM.           *
                ;*                                                   *
                ;*****************************************************

2900 =          CPM     EQU     2900H   ;cp/m beginning load address
3106 =          ENTRY   EQU     CPM+806H        ;cp/m entrance point
0004 =          CDISK   EQU     4       ;current disk storage location
0003 =          IOBYTE  EQU     3H      ;iobyte storage location

                ;*****************************************************
                ;*                                                   *
                ;* Iobyte allows selection of different I/O devices. It *
                ;* can be initialized in any way by changing the equate *
                ;* bellow.                                           *
                ;* Initial iobyte is currently defined as :          *
                ;* console = tty                                     *
                ;* reader = tty                                      *
                ;* punch = tty                                       *
                ;* list = tty                                        *
                ;*                                                   *
                ;*****************************************************

0000 =          INTIOBY EQU     0       ;initial iobyte,

                ;*****************************************************
                ;*                                                   *
                ;* The following equates reference the disk jockey/2d *
                ;* controller board. If your controller is non-standard *
                ;* then all the equates can be changed by re-assigning the *
                ;* value of ORIGIN to be the starting address of your *
                ;* controller.                                       *
                ;*                                                   *
                ;*****************************************************

E000 =          ORIGIN  EQU     0E000H  ;disk jockey/2d beginning address
E003 =          INPUT   EQU     ORIGIN+3        ;serial input routine
E006 =          OUTPUT  EQU     ORIGIN+6        ;serial output routine
E009 =          TKZERO  EQU     ORIGIN+9H       ;track zero seek routine
E00C =          SEEK    EQU     ORIGIN+0CH      ;regular track seek routine
E00F =          SECTOR  EQU     ORIGIN+0FH      ;set sector routine
E012 =          DMA     EQU     ORIGIN+12H      ;read/write beginning address set
E015 =          DISKR   EQU     ORIGIN+15H      ;disk read routine
E018 =          DISKW   EQU     ORIGIN+18H      ;disk write routine
E01B =          SELECT  EQU     ORIGIN+1BH      ;disk selection routine
E021 =          TSTAT   EQU     ORIGIN+21H      ;serial device status routine
E6EE =          STACK   EQU     ORIGIN+6EEH     ;disk jockey/2d ram area for boot only
0099 =          SEKERR  EQU     99H     ;seek error bit mask
00FF =          RWERR   EQU     0FFH    ;read/write error bit mask
000D =          ACR     EQU     0DH     ;carriage return
000A =          ALF     EQU     0AH     ;line feed
E006 =          COTTY   EQU     OUTPUT  ;default character output
E003 =          CITTY   EQU     INPUT   ;default character input

                ;*****************************************************
                ;*                                                   *
                ;* The jump table immediately below must not be altered. *
                ;* It is ok to make the jumps to other address, but the *
                ;* function performed must be the same.              *
                ;*                                                   *
                ;*****************************************************

3E00                    ORG     CPM+1500H

3E00 C32D3E     START   JMP     BOOT    ;cold boot
3E03 C3603E             JMP     WBOOT   ;warm boot
3E06 C3C03E             JMP     CONST   ;console status
3E09 C3CC3E             JMP     CONIN   ;console input
3E0C C3DE3E     CPOUT   JMP     CONOUT  ;console output
3E0F C3F93E             JMP     LIST    ;list output
3E12 C3EE3E             JMP     PUNCH   ;punch output
3E15 C3E43E             JMP     READER  ;reader input
3E18 C3713E             JMP     HOME    ;track zero home
3E1B C3783E             JMP     SELECT  ;disk selection
3E1E C3893E             JMP     SETTRK  ;track seek
3E21 C3[illegible]      JMP     SECTOR  ;sector select
3E24 C3[illegible]      JMP     DMA     ;read/write address select
3E27 C3A13E             JMP     READ    ;disk read
3E2A C3B43E             JMP     WRITE   ;disk write
```

```
84:
85:  E749 2118E0    SAVE    LXI     H,DWRITE        ;change load to write instead of read
86:  E74C 221DE7            SHLD    RDLOOP+2
87:  E74F 215EE7            LXI     H,ERROR ;change error return address
88:  E752 2228E7            SHLD    EXIT+1
89:  E755 215BE7            LXI     H,STALL ;get return address
90:  E758 C303E7            JMP     LOAD+3  ;go and do the write
91:  E75B C35BE7    STALL   JMP     STALL   ;stop here if everything ok !
92:  E75E F5        ERROR   PUSH    PSW     ;save status and flags
93:  E75F C35FE7    ERROR1  JMP     ERROR1  ;stop here on error.
94:                 ********************************************************
95:                 *                                                      *
96:                 * intlz: write this cold boot loader program out to the *
97:                 *        disk.                                         *
98:                 *                                                      *
99:                 ********************************************************
100:
101:
102:  E762 31EEE6    INTLZ   LXI     SP,STACK        ;set up stack
103:  E765 CD09E0            CALL    TKZERO  ;home the drive
104:  E768 0100E7            LXI     B,RAM+300H      ;get starting address of this program
105:  E76B CD12E0            CALL    SETDMA  ;set the write address
106:  E76E 0E01              MVI     C,1     ;set the sector to write
107:  E770 CD0FE0            CALL    SETSEC
108:  E773 CD18E0            CALL    DWRITE  ;write this program out
109:  E776 DA5EE7            JC      ERROR
110:  E779 C379E7    DONE    JMP     DONE    ;stop here
```

```
                ;*************************************************************
                ;*                                                           *
                ;* Boot loader program for cp/m. The following code is       *
                ;* loaded by the boot program on the Disk Jockey 2D. The     *
                ;* 2D loads sector one of track zero into memory at          *
                ;* ORIGIN+300H (the last page of ram on the controller)      *
                ;* then jumps there. It is the responsibility of this code   *
                ;* to load in the rest of cp/m.                              *
                ;*                                                           *
                ;*************************************************************

2900 =          CPMORG  EQU     2900H   ;CPM STARTING ADDRESS
E000 =          ORIGIN  EQU     0E000H  ;Disk Jockey starting address
E400 =          RAM     EQU     ORIGIN+400H     ;ram starting address (of 2D)
E6EE =          STACK   EQU     RAM+2EEH        ;stack pointer starting address within ram
E009 =          TKZERO  EQU     ORIGIN+11Q      ;track zero seek entry point
E00C =          TRKSET  EQU     ORIGIN+14Q      ;entry for track seek
E00F =          SETSEC  EQU     ORIGIN+17Q      ;entry point for sector set
E012 =          SETDMA  EQU     ORIGIN+22Q      ;enrty address for read/write beginning address
E015 =          DREAD   EQU     ORIGIN+25Q      ;disk read entry point
E018 =          DWRITE. EQU     ORIGIN+30Q      ;disk write routine address
E024 =          DMAST   EQU     ORIGIN+44Q      ;disk read/write status routine

E700                    ORG     ORIGIN+700H

                ;*************************************************************
                ;*                                                           *
                ;* load: load in all the rest of cp/m and the cbios. There   *
                ;*       are only two ways to exit this code: 1) If an       *
                ;*       error occurs, a jump is made to the loader on the   *
                ;*       Disk Jockey 2D. 2) If everything works, a jump is   *
                ;*       made to the starting location of the cold boot in   *
                ;*       the cbios.                                          *
                ;*                                                           *
                ;*************************************************************

E700 21003E     LOAD    LXI     H,CPMORG+1500H  ;starting location for cbios
E703 31EEE6             LXI     SP,STACK        ;initialize the stack
E706 E5                 PUSH    H       ;save jump address for return later
E707 01022E     STADDR  LXI     B,2E02H ;reg B=sector count, reg C=starting sector
E70A C5                 PUSH    B       ;save sector and count
E70B CD0FE0             CALL    SETSEC  ;set the sector to read
E70E CD09E0             CALL    TKZERO  ;home the drive
E711 210029             LXI     H,CPMORG        ;starting location for load
E714 44         LDLOOP  MOV     B,H     ;put starting address in B&C
E715 4D                 MOV     C,L
E716 CD12E0             CALL    SETDMA  ;set up starting load address
E719 060A               MVI     B,10    ;retry counter
E71B C5         RDLOOP  PUSH    B       ;save retry count
E71C CD15E0             CALL    DREAD   ;read in the sector
E71F C1                 POP     B       ;fetch retry count
E720 D22AE7             JNC     RDGOOD  ;take jump if read is ok.
E723 05                 DCR     B       ;update retry counter
E724 C21BE7             JNZ     RDLOOP  ;try again if not ten errors
E727 C300E0     EXIT    JMP     ORIGIN  ;start all over from the beginning
E72A C1         RDGOOD  POP     B       ;retetch sector count and f
E72B 05                 DCR     B       ;update the count
E72C C8                 RZ              ;GO TO CPM IF DONE
E72D 0C                 INR     C       ;COMPUTE NEW SECTOR (MOD 26)
E72E 3E1B               MVI     A,27    ;test if over 26
E730 B9                 CMP     C
E731 C236E7             JNZ     OK      ;take jump if sector < 27
E734 0E01               MVI     C,1     ;start with sector 1 of next track
E736 C5         OK      PUSH    B       ;save count and sector
E737 CC0CE0             CZ      TRKSET  ;conditionally set new track
E73A C1                 POP     B       ;restore count and sector #
E73B C5                 PUSH    B       ;save it again
E73C CD0FE0             CALL    SETSEC  ;set new sector
E73F CD24E0             CALL    DMAST   ;get load address
E742 218000             LXI     H,200Q  ;update te load address
E745 09                 DAD     B
E746 C314E7             JMP     LDLOOP  ;read next sector

                ;*************************************************************
                ;*                                                           *
                ;* save: write all of cpm and the cbios onto the disk.       *
                ;*       If an error occurs, the status returned by the      *
                ;*       2D controller will be in location STACK-1.          *
                ;*                                                           *
                ;*************************************************************
```

```
                ;*****************************************
                ;*                                       *
                ;* boot: load in all of cpm and then     *
                ;*       jump there. Initialize iobyte.  *
                ;*                                       *
                ;*****************************************

3E2D 31EEE6     BOOT    LXI     SP,STACK        ;initial stack
3E30 3E00               MVI     A,INTIOBY       ;initialize iobyte
3E32 320300             STA     IOBYTE
3E35 21543F             LXI     H,PROMPT        ;print signon message
3E38 CD8E3E             CALL    MESSG
3E3B AF                 XRA     A       ;select disk A
3E3C 320400             STA     CDISK
3E3F 018000     GOCPM   LXI     B,80H   ;set up default disk buffer
3E42 CD1E3D             CALL    DMA
3E45 3EC3               MVI     A,0C3H  ;put jump instruction to warm boot at 0
3E47 320000             STA     0
3E4A 21033E             LXI     H,START+3
3E4D 220100             SHLD    1
3E50 320500             STA     5       ;put jump to cpm entry at 5
3E53 210631             LXI     H,ENTRY
3E56 220600             SHLD    6
3E59 3A0400             LDA     CDISK   ;jump to cpm with current disk in C
3E5C 4F                 MOV     C,A
3E5D C30029             JMP     CPM

                ;*************************************************************
                ;*                                                           *
                ;* warm boot: load in all of cpm except the cbios. Then      *
                ;* enter cpm.                                                *
                ;*                                                           *
                ;*************************************************************

3E60 31EEE6     WBOOT   LXI     SP,STACK        ;initialize the stack
3E63 AF                 XRA     A       ;select drive A
3E64 4F                 MOV     C,A
3E65 CD1BE0             CALL    SELECT
3E68 01022A             LXI     B,2A02H ;sector count and beginning sector
3E6B CD0AE7             CALL    ORIGIN+70AH     ;call the cold start loader
3E6E C33F3E             JMP     GOCPM   ;now enter cpm

                ;*************************************************************
                ;*                                                           *
                ;* Home: move the head to track zero.                        *
                ;*                                                           *
                ;*************************************************************

3E71 CD09E0     HOME    CALL    TKZERO  ;call the disk jockey/2d
3E74 0E99       SEEK1   MVI     C,SEKERR        ;non relevent error mask

                ;*************************************************************
                ;*                                                           *
                ;* doerrs: returns if no error. Otherwise prints an appro-   *
                ;* priate error messgae, and returns to cpm with an error    *
                ;* indication.                                               *
                ;*                                                           *
                ;*************************************************************

3E76 DA7B3E     DOERRS  JC      DOERR1  ;test if errror
3E79 AF         RWOK    XRA     A       ;return if ok
3E7A C9                 RET
3E7B A1         DOERR1  ANA     C       ;strip off unwanted errors
3E7C 0E08               MVI     C,8     ;error counter
3E7E 217A3F             LXI     H,MSGTBL        ;beginning address of messages
3E81 5E         DOLOOP  MOV     E,M     ;get error address in D&E
3E82 23                 INX     H
3E83 56                 MOV     D,M
3E84 23                 INX     H
3E85 1F                 RAR             ;check if this bit is the error
3E86 DA8D3E             JC      MESSGA  ;yes, exit after printig error
3E89 0D                 DCR     C       ;no error, update the count down
3E8A F2813E             JP      DOLOOP  ;continue if not found

                ;
                ; if fall through then unknown error
                ;

3E8D EB         MESSGA  XCHG            ;put message address into H&L

                ;*************************************************************
                ;*                                                           *
                ;* messg: print the messgae pointed to by H&L and termin-    *
                ;* ated by a 0FFH byte.                                      *
                ;*                                                           *
                ;*************************************************************

3E8E 7E         MESSG   MOV     A,M     ;get character
3E8F A7                 ANA     A       ;test for end
3E90 F8                 RM
3E91 E5                 PUSH    H       ;save address
3E92 4F                 MOV     C,A     ;prep for console output
3E93 CD033E             CALL    CPOUT   ;output it
```

```
3E95 E1             POP     H       ;restore pointer
3E97 23             INX     H       ;bump to next character
3E98 C33E3E         JMP     MESSG   ;continue until end

                    ;*****************************************************
                    ;                                                    *
                    ; settrk: call the disk jockey/2d to seek then exit by *
                    ;         testing for errors.                        *
                    ;                                                    *
                    ;*****************************************************

3E9B CD0CE0 SETTRK  CALL    SEEK
3E9E C3743E         JMP     SEEK1

                    ;*****************************************************
                    ;                                                    *
                    ; read: read one sector from the disk. Try ten times on *
                    ;       errors, before returning an error condition. *
                    ;                                                    *
                    ;*****************************************************

3EA1 2115E0 READ    LXI     H,DISKR ;put disk read address into repeat loop
3EA4 22AB3E RDWR    SHLD    RW+1
3EA7 060A           MVI     B,10    ;retry counter
3EA9 C5     RDWRL   PUSH    B
3EAA CD0000 RW      CALL    0       ;actually call disk read/write
3EAD C1             POP     B
3EAE D2793E         JNC     RWOK    ;exit if succesful
3EB1 05             DCR     B       ;test error count
3EB2 C2A93E         JNZ     RDWRL   ;continue if not zero
3EB5 0EFF           MVI     C,RWERR ;read/write error bit mask
3EB7 C3753E         JMP     DOERRS  ;print the appropriate error message

                    ;*****************************************************
                    ;                                                    *
                    ; write: write data onto the disk, also try ten times *
                    ;        before reporting an error.                  *
                    ;                                                    *
                    ;*****************************************************

3EBA 2118E0 WRITE   LXI     H,DISKW
3EBD C3A43E         JMP     RDWR

                    ;*****************************************************
                    ;                                                    *
                    ; const: get the status for the currently assigned console *
                    ;        device. The console device can be gotten from *
                    ;        iobyte, then a jump to the correct console status *
                    ;        routine is performed.                       *
                    ;                                                    *
                    ;*****************************************************

3EC0 213C3F CONST   LXI     H,CSTBLE        ;beginning of jump table
3EC3 C3CF3E         JMP     CONIN1  ;select correct jump

                    ;*****************************************************
                    ;                                                    *
                    ; csreader: if the console is assigned to the reader then *
                    ;           a jump will be made here, where another jump *
                    ;           will occur to the correct reader status. *
                    ;                                                    *
                    ;*****************************************************

3EC6 21343F CSREADR LXI     H,CSRTBLE       ;beginning of reader status table
3EC9 C3E73E         JMP     READERA

                    ;*****************************************************
                    ;                                                    *
                    ; conin: take the correct jump for the console input  *
                    ;        routine. The jump is based on the two least sig- *
                    ;        nificant bits of iobyte.                    *
                    ;                                                    *
                    ;*****************************************************

3ECC 21043F CONIN   LXI     H,CITBLE        ;beginning of character input table

                    ;
                    ; entry at conin1 will decode the two least significant bits
                    ; of iobyte. This is used by conin,conout, and const.
                    ;
```

```
3ECF 3A0300    CONIN1  LDA     IOBYTE
3ED2 17                RAL

                       *
                       * entry at seldev will form an offset into the table pointed
                       * to by H&L and then pick up the address and jump there.
                       *

3ED3 E606      SELDEV  ANI     6H      ;strip off unwanted bits
3ED5 1600              MVI     D,0     ;form offset
3ED7 5F                MOV     E,A
3ED8 19                DAD     D       ;add offset
3ED9 7E                MOV     A,M     ;pick up high byte
3EDA 23                INX     H
3EDB 66                MOV     H,M     ;pick up low byte
3EDC 6F                MOV     L,A     ;form address
3EDD E9                PCHL            ;go there !

                       ************************************************************
                       *                                                          *
                       * conout: take the proper branch address based on the two  *
                       *         least significant bits of iobyte.                *
                       *                                                          *
                       ************************************************************

3EDE 210C3F    CONOUT  LXI     H,COTBLE        ;beginning of the character out table
3EE1 C3CF3E            JMP     CONIN1  ;do the decode

                       ************************************************************
                       *                                                          *
                       * reader: select the correct reader device for input. The  *
                       *         reader is selected from bits 2 and 3 of iobyte.  *
                       *                                                          *
                       ************************************************************

3EE4 21243F    READER  LXI     H,RTBLE ;beginning of reader input table

                       *
                       * entry at readera will decode bits 2 & 3 of iobyte, used
                       * by csreader.
                       *

3EE7 3A0300    READERA LDA     IOBYTE

                       *
                       * entry at reader1 will shift the bits into position, used
                       * by list and punch.
                       *

3EEA 1F        READR1  RAR
3EEB C3D33E            JMP     SELDEV

                       ************************************************************
                       *                                                          *
                       * punch: select the correct punch device. The selection    *
                       *        comes from bits 4&5 of iobyte.                    *
                       *                                                          *
                       ************************************************************

3EEE 211C3F    PUNCH   LXI     H,PTBLE ;beginning of punch table
3EF1 3A0300            LDA     IOBYTE

                       *
                       * entry at pnch1 rotates bits a little more in prep for
                       * seldev, used by list.
                       *

3EF4 1F        PNCH1   RAR
3EF5 1F                RAR
3EF6 C3EA3E            JMP     READR1

                       ************************************************************
                       *                                                          *
                       * list: select a list device based on bits 6&7 of iobyte   *
                       *                                                          *
                       ************************************************************

3EF9 21143F    LIST    LXI     H,LTBLE ;beginning of the list device routines
3EFC 3A0300            LDA     IOBYTE
3EFF 1F                RAR
3F00 1F                RAR
3F01 C3F43E            JMP     PNCH1

                       ************************************************************
                       *                                                          *
                       * If customizing I/O routines is being performed, the      *
                       * table below should be modified to reflect the changes.   *
                       * all I/O devices are decoded out of iobyte and the jump   *
                       * is taken from the following tables.                      *
                       *                                                          *
                       ************************************************************
```

```
                        *
                        * console input table
                        *

3F04 03E0       CITBLE  DW      CITTY   ;input from tty (currently assigned by intioby,input from 2d)
3F06 473F               DW      CICRT   ;input from crt (currently SWITCHBOARD serial port 1)
3F08 E43E               DW      READER  ;input from reader (depends on reader selection)
3F0A 473F               DW      CIUC1   ;input from user console 1 (currently SWITCHBOARD serial port 1)

                        *
                        * console output table
                        *

3F0C 06E0       COTBLE  DW      COTTY   ;output to tty (currently assigned by intioby,output to 2d)
3F0E 3C3F               DW      COCRT   ;output to crt (currently SWITCHBOARD serial port 1)
3F10 F93E               DW      LIST    ;output to list device (depends on bits 6&7 of iobyte)
3F12 3C3F               DW      COUC1   ;output to user console 1 (currently SWITCHBOARD serial port 1)

                        *
                        * list device table
                        *

3F14 06E0       LTBLE   DW      COTTY   ;output to tty (currently assigned by intioby,output to 2d)
3F16 3C3F               DW      COCRT   ;output to crt (currently SWITCHBOARD serial port 1)
3F18 3C3F               DW      COLPT   ;output to line printer (currently SWITCHBOARD serial port 1)
3F1A 3C3F               DW      COUL1   ;output to user line printer 1 (currently SWITCHBOARD serial port 1)

                        *
                        * punch device table
                        *

3F1C 06E0       PTBLE   DW      COTTY   ;output to the tty (currently assigned by intioby,output to 2d)
3F1E 3C3F               DW      COPTP   ;output to paper tape punch (currently SWITCHBOARD serial port 1)
3F20 3C3F               DW      COUP1   ;output to user punch 1 (currently SWITCHBOARD serial port 1)
3F22 3C3F               DW      COUP2   ;output to user punch 2 (currntlly SWITCHBOARD serial port 1)

                        *
                        * reader device input table
                        *

3F24 03E0       RTBLE   DW      CITTY   ;input from tty (currently assigned by intioby, input from 2d)
3F26 473F               DW      CIPTR   ;input from paper tape reader (currently SWITCHBOARD serial port 1)
3F28 473F               DW      CIUR1   ;input from user reader 1 (currently SWITCHBOARD serial port 1)
3F2A 473F               DW      CIUR2   ;input from user reader 2 (currently SWITCHBOARD serial port 1)

                        *
                        * console status table
                        *

3F2C 533F       CSTBLE  DW      CSTTY   ;status of tty (currently assigned by intioby, ststus from 2d)
3F2E 5B3F               DW      CSCRT   ;status from crt (currently SWITCHBOARD serial port 1)
3F30 C63E               DW      CSREADR ;status from reader (depends on reader device )
3F32 5B3F               DW      CSUC1   ;status from user console 1 (currently SWITCHBOARD serial port 1)

                        *
                        * status fromreader device
                        *

3F34 533F       CSRTBLE DW      CSTTY   ;status from tty (currently assigned by intioby, status of 2d)
3F36 533F               DW      CSPTR   ;status from paper tape reader (currently SWITCHBOARD serial port 1)
3F38 5B3F               DW      CSUR1   ;status from user reader 1 (currently SWITCHBOARD serial port 1)
3F3A 5B3F               DW      CSUR2   ;status of user reader 2 (currently SWITCHBOARD serial port 1)

                        ************************************************************
                        *                                                          *
                        * The following equates set output device to output to     *
                        * the SWITCHBOARD serial port 1.                           *
                        *                                                          *
                        ************************************************************

3F3C =          COCRT   EQU     $       ;output from crt
3F3C =          COUC1   EQU     $       ;output from user console 1
3F3C =          COUL1   EQU     $       ;output from user line printer 1
3F3C =          COPTP   EQU     $       ;output from paper tape punch
3F3C =          COUP1   EQU     $       ;output from user punch 1
3F3C =          COUP2   EQU     $       ;output from user punch 2
3F3C DB02       COLPT   IN      2       ;output from line printer,get status
3F3E E680               ANI     80H     ;wait until ok to send
3F40 CA3C3F             JZ      COLPT
3F43 79                 MOV     A,C     ;output the character
3F44 D301               OUT     1
3F46 C9                 RET

                        ************************************************************
                        *                                                          *
                        * The following equates set the input from the devices to  *
                        * come from the SWITCHBOARD serial port 1                  *
                        *                                                          *
                        ************************************************************
```

```
                ;input from user console 1
3F47 =          CIUC1   EQU     $       ;input from crt
3F47 =          CICRT   EQU     $       ;input from user reader 1
3F47 =          CIUR1   EQU     $       ;input from user reader 2
3F47 =          CIUR2   EQU     $       ;input from paper tape reader, get status
3F47 DB02       CIPTR   IN      2       ;wait for character
3F49 E640               ANI     40H
3F4B CA473F             JZ      CIPTR
3F4E DB01               IN      1
3F50 E67F               ANI     7FH     ;strip off the parity
3F52 C9                 RET

                ;*****************************************************
                ;                                                    *
                ; console status routines, test if a character has arrived *
                ;                                                    *
                ;*****************************************************

3F53 CD21E0     CSTTY   CALL    TSTAT   ;status from disk jockey 2d
3F56 3E00       STAT    MVI     A,0     ;prep for zero return
3F58 C0                 RNZ             ;nothing found
3F59 3D                 DCR     A       ;return with 0FFH
3F5A C9                 RET

                ;*****************************************************
                ;                                                    *
                ; The following equates cause the devices to get status *
                ; from the SWITCHBOARD serial port 1.                *
                ;                                                    *
                ;*****************************************************

3F5B =          CSUR1   EQU     $       ;status of user reader 1
3F5B =          CSUR2   EQU     $       ;status of user reader 2
3F5B =          CSPTR   EQU     $       ;status of paper tape reader
3F5B =          CSUC1   EQU     $       ;status of user console 1
3F5B DB02       CSCRT   IN      2       ;status from crt, get status
3F5D E640               ANI     40H     ;strip of data ready bit
3F5F EE40               XRI     40H     ;make correct polarity
3F61 C3563F             JMP     STAT    ;return proper indication

                ;*****************************************************
                ;                                                    *
                ; The following messages could be put out by the cbios. *
                ;                                                    *
                ;*****************************************************

3F64 0D0A       PROMPT  DB      ACR,ALF ;prompt message - "16K CP/M VERS 1.4"
3F66 31364B20           DB      '16K '
3F6A 43502F4D           DB      'CP/M'
3F6E 20564552           DB      ' VER'
3F72 5320312E           DB      'S 1.'
3F76 34                 DB      '4'
3F77 0D0A               DB      ACR,ALF
3F79 FF                 DB      0FFH

                ;
                ; error message table
                ;

3F7A 8C3F       MSGTBL  DW      ILLDATA ;illegal data
3F7C 983F               DW      DATAREQ ;data request
3F7E A33F               DW      DATALOS ;data lost
3F80 AF3F               DW      CRCERR  ;crc error
3F82 BB3F               DW      ILLSEC  ;illegal sector
3F84 CF3F               DW      ILLDMA  ;illegal dma
3F86 DA3F               DW      WRITPRO ;write protected
3F88 E53F               DW      NOTRDY  ;not ready
3F8A F13F               DW      UNKNOWN ;unknown error

3F8C 0D0A       ILLDATA DB      ACR,ALF
3F8E 494C474C20         DB      'ILGL DATA'
3F97 FF                 DB      0FFH
3F98 0D0A       DATAREQ DB      ACR,ALF
3F9A 4441544120         DB      'DATA REQ'
3FA2 FF                 DB      0FFH
3FA3 0D0A       DATALOS DB      ACR,ALF
3FA5 4441544120         DB      'DATA LOST'
3FAE FF                 DB      0FFH
3FAF 0D0A       CRCERR  DB      ACR,ALF
3FB1 4352432045         DB      'CRC ERROR'
3FBA FF                 DB      0FFH
3FBB 0D0A       ILLSEC  DB      ACR,ALF
3FBD 494C474C20         DB      'ILGL SECTOR/TRACK'
3FCE FF                 DB      0FFH
3FCF 0D0A       ILLDMA  DB      ACR,ALF
3FD1 494C474C20         DB      'ILGL DMA'
3FD9 FF                 DB      0FFH
3FDA 0D0A       WRITPRO DB      ACR,ALF
3FDC 5752542050         DB      'WRT PROT'
3FE4 FF                 DB      0FFH
3FE5 0D0A       NOTRDY  DB      ACR,ALF
3FE7 4E4F542052         DB      'NOT READY'
3FF0 FF                 DB      0FFH
3FF1 0D0A       UNKNOWN DB      ACR,ALF
3FF3 554E4B4F57         DB      'UNKOWN ERROR'
3FFF FF                 DB      0FFH
```

```
NEWFIRM4

                                1   *DISK JOCKEY/2D FIRMWARE REVISION 4
                                2   *
340:000                         3           AORG 340:000Q
                                4   *
340:000 340:000                 5   ORIGIN  EQU  340:000Q
                                6   *
340:000 240:000                 7           SORG 240:000Q
                                8   *
340:000 344:000                 9   RAM     EQU  ORIGIN+4:000Q
340:000 343:370                 10  IO      EQU  ORIGIN+3:370Q
340:000 343:370                 11  UDATA   EQU  IO
340:000 343:371                 12  DREG    EQU  IO+1
340:000 343:371                 13  USTAT   EQU  DREG
340:000 343:372                 14  DCMD    EQU  IO+2
340:000 343:372                 15  DSTAT   EQU  DCMD
340:000 343:374                 16  CMDREG  EQU  IO+4
340:000 343:374                 17  CSTAT   EQU  CMDREG
340:000 343:375                 18  TRKREG  EQU  IO+5
340:000 343:376                 19  SECREG  EQU  IO+6
340:000 343:377                 20  DATREG  EQU  IO+7
                                21  *
                                22  *
340:000 000:200                 23  RCMD    EQU  200Q
340:000 000:240                 24  WCMD    EQU  240Q
340:000 000:004                 25  HEAD    EQU  4
340:000 000:020                 26  LOAD    EQU  20Q
340:000 000:001                 27  DENSTY  EQU  1
340:000 000:030                 28  ULOAD   EQU  30Q
340:000 000:004                 29  RSTBIT  EQU  4
340:000 000:002                 30  ACCESS  EQU  2
340:000 000:040                 31  READY   EQU  40Q
340:000 000:020                 32  INDEX   EQU  20Q
340:000 000:304                 33  RACMD   EQU  304Q
340:000 000:320                 34  CLRCMD  EQU  320Q
340:000 000:035                 35  SVCMD   EQU  35Q
340:000 000:030                 36  SKCMD   EQU  30Q
340:000 000:011                 37  HCMD    EQU  11Q
340:000 000:004                 38  ISTAT   EQU  4
340:000 000:010                 39  OSTAT   EQU  10Q
340:000 000:010                 40  DSIDE   EQU  10Q
340:000 000:004                 41  TZERO   EQU  4
340:000 000:003                 42  MDINT   EQU  3
340:000 000:036                 43  LIGHT   EQU  36Q
340:000 000:076                 44  NOLITE  EQU  76Q
                                45  *
                                46  *
340:000 303 141 340             47  DBOOT   JMP  BOOT
340:003 303 377 340             48  TERMIN  JMP  CIN
340:006 303 360 340             49  TRMOUT  JMP  COUT
340:011 303 157 341             50  TKZERO  JMP  HOME
340:014 303 240 341             51  TRKSET  JMP  SEEK
340:017 303 223 341             52  SETSEC  JMP  SECSET
340:022 303 126 341             53  SETDMA  JMP  DMA
340:025 303 251 341             54  DREAD   JMP  READ
340:030 303 374 341             55  DWRITE  JMP  WRITE
340:033 303 113 341             56  SELDRV  JMP  DRIVE
340:036 303 016 341             57  TPANIC  JMP  CPAN
340:041 303 031 341             58  TMSTAT  JMP  TMSTAT
340:044 303 103 341             59  DMAST   JMP  DMSTAT
340:047 303 037 341             60  STATUS  JMP  DISKST
340:052 303 333 340             61  DSKERR  JMP  LERROR
340:055 303 277 343             62  SETDEN  JMP  DENFIX
340:060 303 336 343             63  SETSID  JMP  SIDEFX
                                64  *
                                65  *
340:063 000:056                 66          DS   56Q
                                67  *
                                68  *
340:141                         69  BOOT
340:141 061 372 346             70          LXI  SP,TRACK+1 initialize the SP
340:144 041 341 346             71          LXI  H,TIMER-4 memory test data
340:147 021 353 342             72          LXI  D,STABLE the ROM compare
340:152 006 004                 73          MVI  B,4        -data and count
340:154                         74  TESTL
340:154 032                     75          LDAX D          get the ROM data
340:155 276                     76          CMP  M          compare w/memory
340:156 302 172 340             77          JNZ  DRESET     do timeout?
340:161 043                     78          INX  H          move the
340:162 023                     79          INX  D          -two pointers
340:163 005                     80          DCR  B          dec the count
340:164 302 154 340             81          JNZ  TESTL      test more?
340:167 303 175 340             82          JMP  DSETUP     no!
340:172                         83  DRESET
340:172 315 351 343             84          CALL TIMOUT     reset time out
340:175                         85  DSETUP
340:175 041 001 000             86          LXI  H,1
340:200 345                     87          PUSH H          track 0, sector 1
340:201 056 003                 88          MVI  L,MDINT    set up side
340:203 345                     89          PUSH H          -select .. also
340:204 046 377                 90          MVI  H,377Q     -parameter
340:206 345                     91          PUSH H          -and
340:207 345                     92          PUSH H          -track info
340:210 345                     93          PUSH H          -for the 4
340:211 345                     94          PUSH H          -drives
340:212 041 000 000             95          LXI  H,0        initialize
340:215 345                     96          PUSH H          -the track
340:216 063                     97          INX  SP         -zero flag
340:217 046 010                 98          MVI  H,10Q      current disk
340:221 345                     99          PUSH H          -and new disk
340:222 046 376                 100         MVI  H,376Q     initialize DRVSEL
340:224 345                     101         PUSH H          -and HDFLAG
340:225 046 347                 102         MVI  H,RAM+3:000Q/256
340:227 345                     103         PUSH H          DMA address
340:230 046 030                 104         MVI  H,30Q      temporary TIMER
340:232 345                     105         PUSH H          -constant
340:233 076 003                 106         MVI  A,MDINT    initialize 1791
340:235 062 372 343             107         STA  DCMD       -control bits
340:240 076 320                 108         MVI  A,CLRCMD   1791 reset
340:242 062 374 343             109         STA  CMDREG     -command
340:245                         110 LDHEAD
340:245 257                     111         XRA  A          load the head
340:246 315 045 343             112         CALL HDCHK      -and test for
340:251 322 267 340             113         JNC  DOOROK     -drive ready
340:254 076 036                 114         MVI  A,LIGHT    turn on the
340:256 062 352 346             115         STA  DRVSEL     -error LED
340:261 315 351 343             116         CALL TIMOUT     time out to
340:264 303 245 340             117         JMP  LDHEAD     -close drive door
340:267                         118 DOOROK
```

```
340:267 076 076          MVI   A,NOLITE turn off the
340:271 062 352 346      STA   DRVSEL   -error LED
340:274 066 003          MVI   M,MDINT  open data reg
340:276 341              POP   H        discard old TIMER
340:277 315 242 343      CALL  MEASUR   head load time
340:302 301              POP   B        recover boot
340:303 305              PUSH  B        addr from DMAADDR
340:304 325              PUSH  D        -new TIMER value
340:305 052 355 342      LHLD  STABLE+2 set up
340:310 345              PUSH  H        -time out
340:311 052 353 342      LHLD  STABLE   -test
340:314 345              PUSH  H        -data
340:315 000              NOP   .        debug instruction
340:316 305              PUSH  B        boot address
340:317 006 012          MVI   B,12Q    number of retrys
340:321                  LDLOOP
340:321 305              PUSH  B        save the retry no
340:322 315 251 341      CALL  READ     read boot sector
340:325 301              POP   B        restore retry no
340:326 320              RNC   .        successful read?
340:327 005              DCR   B        no! count down
340:330 302 321 340      JNZ   LDLOOP   try again
340:333                  LERROR
340:333 016 077          MVI   C,77Q
340:335 021 303 242      LXI   D,242:303Q
340:340                  LELOOP
340:340 033              DCX   D
340:341 172              MOV   A,D
340:342 263              ORA   E
340:343 302 340 340      JNZ   LELOOP
340:346 076 040          MVI   A,40Q    blink
340:350 251              XRA   C        -the LED at
340:351 062 371 343      STA   DREG     -top of the
340:354 117              MOV   C,A      -circuit board
340:355 303 335 340      JMP   LERROR+2
                         *
                         *
340:360                  COUT
340:360 072 371 343      LDA   USTAT    get UART status
340:363 346 010          ANI   OSTAT    output ready bit
340:365 302 360 340      JNZ   COUT     test output ready
340:370 171              MOV   A,C      character data
340:371 057              CMA
340:372 062 370 343      STA   UDATA    send to UART
340:375 057              CMA
340:376 311              RET
                         *
                         *
340:377                  CIN
340:377 072 371 343      LDA   USTAT    get UART status
341:002 346 004          ANI   ISTAT    input ready bit
341:004 302 377 340      JNZ   CIN      test input ready
341:007 072 370 343      LDA   UDATA    get the character
341:012 057              CMA   .        true data
341:013 346 177          ANI   177Q     trim to 7 bits
341:015 311              RET
                         *
                         *
341:016                  CPAN
341:016 072 371 343      LDA   USTAT    get UART status
341:021 346 004          ANI   ISTAT    input ready bit
341:023 300              RNZ   .        test for character
341:024 315 377 340      CALL  CIN      get character
341:027 271              CMP   C        test for panic
341:030 311              RET
                         *
                         *
341:031                  TMSTAT
341:031 072 371 343      LDA   USTAT    get UART status
341:034 346 004          ANI   ISTAT    input ready bit
341:036 311              RET
                         *
                         *
341:037                  DISKST
341:037 072 376 343      LDA   SECREG   get current
341:042 107              MOV   B,A      -sector no in B
341:043 072 375 343      LDA   TRKREG   get current
341:046 117              MOV   C,A      -track no in C
341:047 072 366 346      LDA   DCREG    get current
341:052 057              CMA   .        -density in
341:053 346 001          ANI   1        -the msb
341:055 017              RRC   .        -position
341:056 127              MOV   D,A      save in D
341:057 072 367 346      LDA   SIDE     put the
341:062 027              RAL   .        -side
341:063 027              RAL   .        -select
341:064 027              RAL   .        -flag
341:065 202              ADD   D        -in bit
341:066 127              MOV   D,A      -position 6
341:067 072 375 346      LDA   SECLEN   put the
341:072 027              RAL   .        -sector length
341:073 027              RAL   .        -code in bits
341:074 202              ADD   D        -2 & 3
341:075 127              MOV   D,A
341:076 072 354 346      LDA   CDISK    put the current
341:101 202              ADD   D        -disk no in bits
341:102 311              RET   .        -0 & 1
                         *
                         *
341:103                  DMSTAT
341:103 345              PUSH  H        save the H-L pair
341:104 052 347 346      LHLD  DMAADR   DMA addr to H-L
341:107 104              MOV   B,H      move the DMA
341:110 115              MOV   C,L      -addr to B-C
341:111 341              POP   H        recover H-L
341:112 311              RET
                         *
                         *
341:113                  DRIVE
341:113 076 374          MVI   A,374Q   test for the
341:115 201              ADD   C        -new drive number
341:116 076 020          MVI   A,20Q    less than 4
341:120 330              RC
341:121 171              MOV   A,C      store the new
341:122 062 353 346      STA   DISK     drive in DISK
341:125 311              RET
                         *
                         *
341:126                  DMA
341:126 041 010 040      LXI   H,B-ORIGIN test the
```

```
341:131 011
341:132 322 150 341
341:135 041 000 034
341:140 011
341:141 332 150 341
341:144 067
341:145 076 020
341:147 311
341:150
341:150 140
341:151 151
341:152 042 347 346
341:155 257
341:156 311


341:157
341:157 315 173 341
341:162 365
341:163 237
341:164 062 371 346
341:167 361
341:170 303 052 342
341:173
341:173 315 357 342
341:176 330
341:177 257
341:200 062 355 346
341:203 062 351 346
341:206 041 000 000
341:211 076 011
341:213 315 154 343
341:216 346 004
341:220 300
341:221 067
341:222 311


341:223
341:223 257
341:224 261
341:225 067
341:226 310
341:227 171
341:230 376 033
341:232 077
341:233 330
341:234 062 370 346
341:237 311


341:240
341:240 171
341:241 376 115
341:243 077
341:244 330
341:245 062 371 346
341:250 311
```

```
239             DAD   B         -DMA address
240             JNC   DMASET    -for conflict
241             LXI   H,-RAM    -with the I/O
242             DAD   B         -on the DJ/2D
243             JC    DMASET    controller
244             STC
245             MVI   A,20Q
246             RET
247   DMASET
248             MOV   H,B       get the DMA addr
249             MOV   L,C       to the H-L pair
250             SHLD  DMAADR    store
251             XRA   A         clear the error
252             RET   .         -flag and return
253   *
254   *
255   HOME
256             CALL  HENTRY    head to trk zero
257             PUSH  PSW       save the flags
258             SBB   A         update the
259             STA   TRACK     -track register
260             POP   PSW       recover the flags
261             JMP   LEAVE     unload the head
262   HENTRY
263             CALL  HDLOAD    load the head
264             RC    .         test for ready error
265             XRA   A         update
266             STA   TZFLAG    -the two
267             STA   HDFLAG    -flags
268             LXI   H,0       time out constant
269             MVI   A,HCMD    do the home
270             CALL  CENTRY    -command
271             ANI   TZERO     track zero bit
272             RNZ
273             STC   .         error flag
274             RET
275   *
276   *
277   SECSET
278             XRA   A         test for
279             ORA   C         -sector zero
280             STC   .         error flag
281             RZ
282             MOV   A,C       test for
283             CPI   27        -sector
284             CMC   .         too large
285             RC
286             STA   SECTOR    save
287             RET
288   *
289   *
290   SEEK
291             MOV   A,C       test for
292             CPI   77        -track
293             CMC   .         -too large
294             RC
295             STA   TRACK     save
296             RET
297   *
298   *
```

```
341:251
341:251 315 065 342
341:254 341
341:255 076 100
341:257 273
341:260 312 350 341
341:263 345
341:264 031
341:265 031
341:266 345
341:267 371
341:270 035
341:271 041 377 343
341:274 076 200
341:276 062 374 343
341:301 106
341:302
341:302 116
341:303 305
341:304 106
341:305 035
341:306 302 302 341
341:311 025
341:312 362 302 341
341:315 116
341:316 305
341:317 061 306 346
341:322 341
341:323 321
341:324
341:324 053
341:325 106
341:326 032
341:327 167
341:330 170
341:331 022
341:332 023
341:333 175
341:334 273
341:335 302 324 341
341:340 174
341:341 272
341:342 302 324 341
341:345 303 030 342


341:350
341:350 007
341:351 117
341:352 021 377 343
341:355 076 200
341:357 062 374 343
341:362
341:362 032
341:363 167
341:364 043
341:365 015
341:366 302 362 341
341:371 303 030 342
```

```
299   READ
300             CALL  PREP      prepare for read
301             POP   H         recover DMA addr
302             MVI   A,100Q    test the
303             CMP   E         -read command for
304             JZ    SINGLE    -single density
305             PUSH  H         save DMA addr
306             DAD   D         ending
307             DAD   D         -address+1
308             PUSH  H         save also
309             SPHL  .         adjust SP
310             DCR   E         adjust byte cnt
311             LXI   H,DATREG  data register
312             MVI   A,RCMD    do the read
313             STA   CMDREG    -command
314             MOV   B,M       first byte of data
315   RLOOP
316             MOV   C,M       2nd byte of data pair
317             PUSH  B         save the data pair
318             MOV   B,M       1st byte of next pair
319             DCR   E         dec low byte of cnt
320             JNZ   RLOOP
321             DCR   D         dec high byte of cnt
322             JP    RLOOP
323             MOV   C,M       get last byte
324             PUSH  B         store last pair
325             LXI   SP,STACK-6 adjust SP
326             POP   H         get the end addr+1
327             POP   D         get the begin addr
328   ALOOP
329             DCX   H         early data pointer
330             MOV   B,M       get early data
331             LDAX  D         get late data
332             MOV   M,A       swap the
333             MOV   A,B       -two bytes
334             STAX  D         -of data
335             INX   D         advance late ptr
336             MOV   A,L       compare
337             CMP   E         -the two
338             JNZ   ALOOP     -data
339             MOV   A,H       -pointers
340             CMP   D         -for a
341             JNZ   ALOOP     -match
342             JMP   CBUSY
343   *
344   *
345   SINGLE
346             RLC   .         initialize the data
347             MOV   C,A       -count to 128
348             LXI   D,DATREG  1791 data register
349             MVI   A,RCMD    issue the
350             STA   CMDREG    -read command
351   SHORTL
352             LDAX  D         get data from disk
353             MOV   M,A       move data to memory
354             INX   H         increment data pointer
355             DCR   C         decrement data count
356             JNZ   SHORTL    test for
357             JMP   CBUSY     -transfer done
358   *
```

```
341:374
341:374 315 065 342
341:377 341
342:000 371
342:001 035
342:002 041 377 343
342:005 076 240
342:007 062 374 343
342:012 301
342:013 161
342:014
342:014 160
342:015 301
342:016 161
342:017 035
342:020 302 014 342
342:023 025
342:024 362 014 342
342:027 160
342:030
342:030 072 374 343
342:033 037
342:034 332 030 342
342:037 027
342:040 346 337
342:042 312 046 342
342:045 067
342:046
342:046 052 312 346
342:051 371
342:052
342:052 365
342:053 072 366 346
342:056 356 020
342:060 062 372 343
342:063 361
342:064 311


342:065
342:065 321
342:066 041 000 000
342:071 071
342:072 061 314 346
342:075 345
342:076 052 347 346
342:101 345
342:102 325
342:103 041 046 342
342:106 345
342:107 315 357 342
342:112 330
342:113 072 375 343
342:116 074
342:117 314 173 341
342:122 332 237 342
342:125 041 375 343
342:130 072 371 346
342:133 276
```

```
359  *
360  WRITE
361          CALL  PREP      prepare for write
362          POP   H         recover DMA addr
363          SPHL  .         adjust SP
364          DCR   E         adjust byte cnt
365          LXI   H,DATREG  data reg
366          MVI   A,WCMD    do a write
367          STA   CMDREG    -command
368          POP   B         get the 1st data pair
369          MOV   M,C       write first byte
370  WLOOP
371          MOV   M,B       write high byte
372          POP   B         get next data pair
373          MOV   M,C       write low byte
374          DCR   E         dec low byte of cnt
375          JNZ   WLOOP
376          DCR   D         dec high byte of cnt
377          JP    WLOOP
378          MOV   M,B       write last byte
379  CBUSY
380          LDA   CSTAT     get 1791 status
381          RAR   .         busy bit to carry
382          JC    CBUSY
383          RAL   .         restore the ACC
384          ANI   337Q      error bit mask
385          JZ    RETURN    go to the exit
386          STC   .         set the error flag
387  RETURN
388          LHLD  STACK-2   get the user SP
389          SPHL  .         restore the user SP
390  LEAVE
391          PUSH  PSW
392          LDA   DCREG     1791 control bits
393          XRI   LOAD      toggle the
394          STA   DCMD      -load bit
395          POP   PSW       -the 1791 data reg
396          RET
397  *
398  *
399  PREP
400          POP   D         get return addr
401          LXI   H,0       get the user's
402          DAD   SP        -stack pointer
403          LXI   SP,STACK  local stack
404          PUSH  H         save user's SP
405          LHLD  DMAADR    DMA address
406          PUSH  H         save DMA addr
407          PUSH  D         save return addr
408          LXI   H,RETURN  error
409          PUSH  H         -exit
410          CALL  HDLOAD    load the head
411          RC    .         disk not ready?
412          LDA   TRKREG    get the old trk
413          INR   A         test for head
414          CZ    HENTRY    -not calibrated
415          JC    SERROR    seek error?
416          LXI   H,TRKREG  present trk
417          LDA   TRACK     the new track
418          CMP   M         test for head motion
```

```
342:134 043
342:135 043
342:136 167
342:137 171
342:140 062 372 343
342:143 312 175 342
342:146 257
342:147 062 351 346
342:152 072 372 343
342:155 346 010
342:157 037
342:160 037
342:161 037
342:162 306 030
342:164 041 000 000
342:167 315 154 343
342:172 332 237 342
342:175
342:175 072 351 346
342:200 267
342:201 302 311 342
342:204 006 002
342:206
342:206 076 035
342:210 315 147 343
342:213 346 231
342:215 312 245 342
342:220 072 366 346
342:223 356 001
342:225 062 366 346
342:230 062 372 343
342:233 005
342:234 302 206 342
342:237
342:237 315 157 341
342:242 303 236 343
342:245
342:245 006 012
342:247
342:247 021 377 343
342:252 041 372 346
342:255 076 304
342:257 062 374 343
342:262
342:262 032
342:263 167
342:264 054
342:265 302 262 342
342:270 041 374 343
342:273 315 176 343
342:276 267
342:277 312 311 342
342:302 005
342:303 302 247 342
342:306 303 237 342
342:311
342:311 072 375 346
342:314 117
342:315 006 000
342:317 041 353 342
```

```
419          INX   H         advance to the
420          INX   H         -data register
421          MOV   M,A       save the new trk
422          MOV   A,C       turn off data
423          STA   DCMD      -access control bit
424          JZ    TVERFY    test for seek
425          XRA   A         force a read
426          STA   HDFLAG    -header operation
427          LDA   DSTAT     get the
428          ANI   DSIDE     -double
429          RAR   .         -sided
430          RAR   .         -flag
431          RAR   .         -to do 3 ms
432          ADI   SKCMD     -step operation
433          LXI   H,0       do a seek
434          CALL  CENTRY    -command
435          JC    SERROR    seek error?
436  TVERFY
437          LDA   HDFLAG    get the force
438          ORA   A         -verify track flag
439          JNZ   CHKSEC    no seek & head OK
440          MVI   B,2       verify retry no
441  SLOOP
442          MVI   A,SVCMD   do a verify
443          CALL  COMAND    -command
444          ANI   231Q      error bit mask
445          JZ    RDHDR     no error!
446          LDA   DCREG     1791 control reg
447          XRI   DENSTY    flip the density bit
448          STA   DCREG     update and
449          STA   DCMD      -change density
450          DCR   B         dec retry count
451          JNZ   SLOOP     -and try again
452  SERROR
453          CALL  HOME      there is a
454          JMP   BERROR    -hard seek error
455  RDHDR
456          MVI   B,12Q     number of retrys
457  RHLOOP
458          LXI   D,DATREG  data register
459          LXI   H,TRACK+1 storage area
460          MVI   A,RACMD   do the read
461          STA   CMDREG    -header command
462  RHL1
463          LDAX  D         get a data byte
464          MOV   M,A       store in memory
465          INR   L         inc mem pointer
466          JNZ   RHL1      test for more data
467          LXI   H,CSTAT   wait for 1791
468          CALL  BUSY      to finish cmd
469          ORA   A         test for errors
470          JZ    CHKSEC    transfer OK?
471          DCR   B         dec retry count
472          JNZ   RHLOOP    test for
473          JMP   SERROR    -hard error
474  CHKSEC
475          LDA   SECLEN    get the sector
476          MOV   C,A       -size and setup
477          MVI   B,0       -the offset
478          LXI   H,STABLE  sec size tbl
```

```
342:322 011
342:323 072 370 346
342:326 107
342:327 206
342:330 076 020
342:332 330
342:333 341
342:334 170
342:335 062 376 343
342:340 041 100 000
342:343
342:343 015
342:344 124
342:345 135
342:346 370
342:347 051
342:350 303 343 342


342:353
342:353 345
342:354 345
342:355 360
342:356 367


342:357
342:357 041 353 346
342:362 116
342:363 043
342:364 136
342:365 161
342:366 043
342:367 173
342:370 271
342:371 176
342:372 066 004
342:374 043
342:375 312 045 343
343:000 345
343:001 026 000
343:003 102
343:004 031
343:005 031
343:006 072 366 346
343:011 167
343:012 043
343:013 021 375 343
343:016 032
343:017 167
343:020 341
343:021 011
343:022 011
343:023 176
343:024 062 366 346
343:027 043
343:030 176
343:031 022
343:032 076 177
343:034
```

```
479             DAD   B         add the offset
480             LDA   SECTOR    get the sector
481             MOV   B,A       save in B
482             ADD   M         compare w/table entry
483             MVI   A,20Q     error flag
484             RC    .         error return
485             POP   H         return addr to TOS
486             MOV   A,B       save the sector
487             STA   SECREG    -in sector reg
488             LXI   H,100Q    half page count
489   SZLOOP
490             DCR   C         sec size count
491             MOV   D,H       half size count
492             MOV   E,L       -to the D-E pair
493             RM    .         return if done
494             DAD   H         double the xfer
495             JMP   SZLOOP    -size count
496   *
497   *
498   STABLE
499             DB    345Q
500             DB    345Q
501             DB    360Q
502             DB    367Q
503   *
504   *
505   HDLOAD
506             LXI   H,DISK
507             MOV   C,M       new disk no to C
508             INX   H
509             MOV   E,M       current disk to E
510             MOV   M,C       update current disk
511             INX   H         head load constant
512             MOV   A,E       test for
513             CMP   C         -disk change
514             MOV   A,M       head load flag
515             MVI   M,HEAD    update head load
516             INX   H         addr of disk table
517             JZ    HDCHK     no disk change?
518             PUSH  H         save table address
519             MVI   D,0       set up the
520             MOV   B,D       -offset address.
521             DAD   D         get the current
522             DAD   D         -disk parameters
523             LDA   DCREG     save the
524             MOV   M,A       density info
525             INX   H         current track
526             LXI   D,TRKREG
527             LDAX  D         get current trk
528             MOV   M,A       save
529             POP   H         recover tbl addr
530             DAD   B         add the
531             DAD   B         -offset
532             MOV   A,M       get control bits
533             STA   DCREG     update DCREG
534             INX   H         get the old
535             MOV   A,M       track number
536             STAX  D         and update 1791
537             MVI   A,177Q    disk select bits
538   DSROT
```

```
343:034 007
343:035 015
343:036 362 034 343
343:041 062 352 346
343:044 257
343:045
343:045 041 372 343
343:050 246
343:051 062 351 346
343:054 365
343:055 072 352 346
343:060 117
343:061 072 367 346
343:064 057
343:065 241
343:066 062 371 343
343:071 072 366 346
343:074 117
343:075 072 371 346
343:100 326 001
343:102 237
343:103 075
343:104 057
343:105 261
343:106 167
343:107 356 002
343:111 117
343:112 361
343:113 302 131 343
343:116 345
343:117 052 345 346
343:122
343:122 053
343:123 174
343:124 265
343:125 302 122 343
343:130 341
343:131
343:131 176
343:132 346 040
343:134 310
343:135
343:135 072 366 346
343:140 366 030
343:142 167
343:143 076 200
343:145 067
343:146 311


343:147
343:147 052 345 346
343:152 051
343:153 051
343:154
343:154 353
343:155 041 374 343
343:160 303 166 343
343:163
343:163 303 357 342
```

```
539             RLC   .         rotate to
540             DCR   C         -select the
541             JP    DSROT     -proper drive
542             STA   DRVSEL    save
543             XRA   A         force head load
544   HDCHK
545             LXI   H,DSTAT   test for
546             ANA   M         -head loaded
547             STA   HDFLAG    save the head
548             PUSH  PSW       -loaded status
549             LDA   DRVSEL    get current drive
550             MOV   C,A       save
551             LDA   SIDE      get current side
552             CMA   .         and merge
553             ANA   C         -with drive select
554             STA   DREG      select drive & side
555             LDA   DCREG     1791 control bits
556             MOV   C,A       save
557             LDA   TRACK     get the new trk
558             SUI   1         force single
559             SBB   A         -density
560             DCR   A         -if track = 0
561             CMA   .         compliment
562             ORA   C         merge w/control bits
563             MOV   M,A       set 1791 control
564             XRI   ACCESS    toggel access bit
565             MOV   C,A       save PREP routine
566             POP   PSW       head load status
567             JNZ   RDYCHK    conditionally
568             PUSH  H         -wait for head
569             LHLD  TIMER     -load time out
570   TLOOP
571             DCX   H         count down
572             MOV   A,H       -40 ms for
573             ORA   L         -head load
574             JNZ   TLOOP     -time out
575             POP   H         disk status addr
576   RDYCHK
577             MOV   A,M       test for
578             ANI   READY     -disk ready
579             RZ
580   UNLOAD
581             LDA   DCREG     force a
582             ORI   ULOAD     -head
583             MOV   M,A       -unload
584             MVI   A,200Q    set disk
585             STC   .         -not ready
586             RET   .         -error flag
587   *
588   *
589   COMAND
590             LHLD  TIMER     get index count
591             DAD   H         -and multiply
592             DAD   H         -by four
593   CENTRY
594             XCHG  .         save in D-E pair
595             LXI   H,CSTAT   issue command
596             JMP   PATCH+3   jump around patch
597   PATCH
598             JMP   HDLOAD    patch for old ATE
```

```
343:166 000
343:167 000
343:170 167
343:171
343:171 176
343:172 037
343:173 322 171 343
343:176
343:176 176
343:177 037
343:200 176
343:201 320
343:202 033
343:203 172
343:204 263
343:205 302 176 343
343:210 345
343:211 043
343:212 126
343:213 072 366 346
343:216 356 004
343:220 062 372 343
343:223 356 004
343:225 343
343:226 062 372 343
343:231 066 320
343:233 343
343:234 162
343:235 341
343:236
343:236 076 021
343:240 067
343:241 311


343:242
343:242 021 000 000
343:245 041 372 343
343:250 016 020
343:252
343:252 176
343:253 241
343:254 302 252 343
343:257
343:257 176
343:260 241
343:261 312 257 343
343:264
343:264 023
343:265 343
343:266 343
343:267 343
343:270 343
343:271 176
343:272 241
343:273 302 264 343
343:276 311


343:277
```

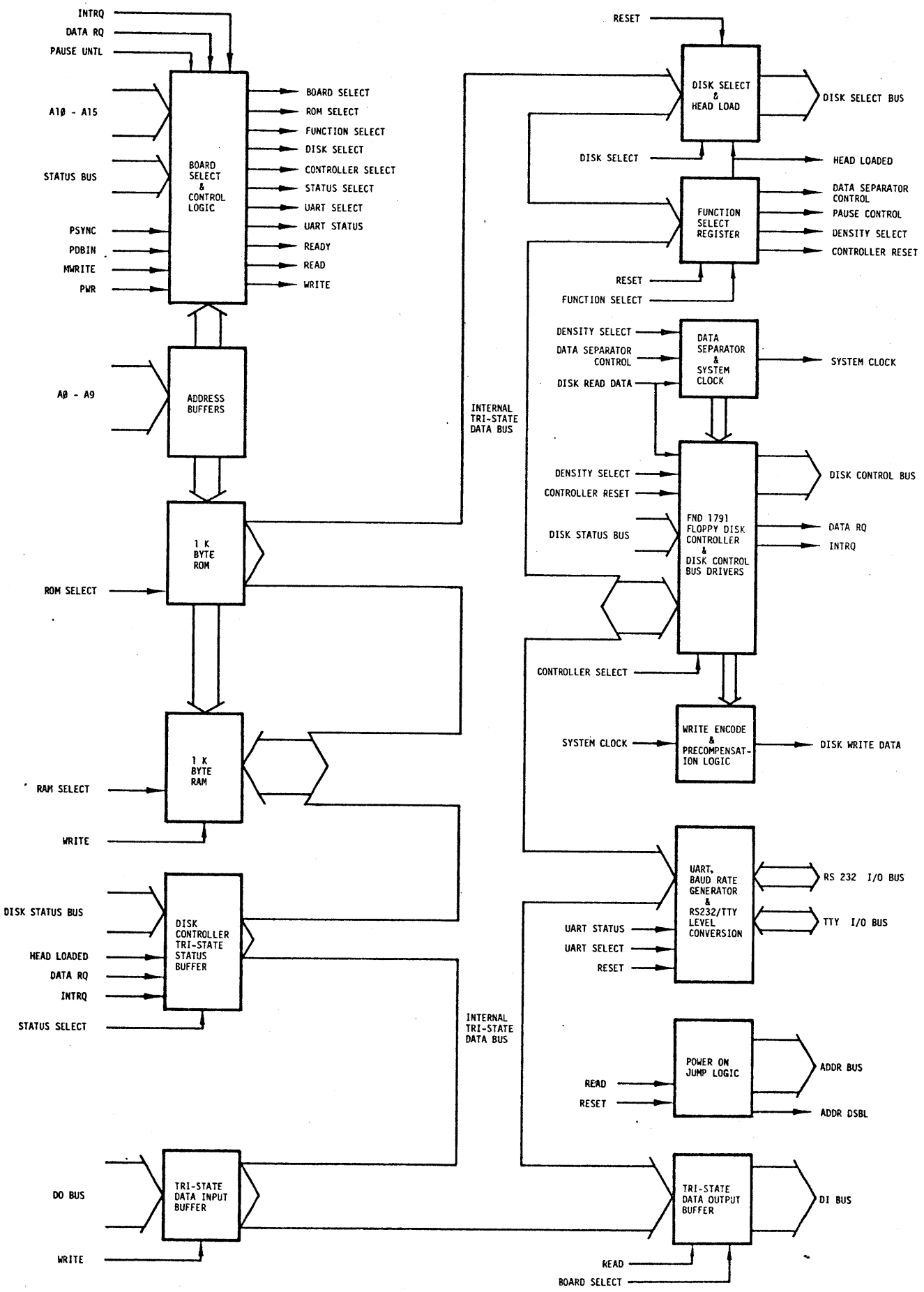
```
599             NOP   fill      instruction
600             NOP   fill      instruction
601             MOV   M,A       to the 1791
602   NBUSY
603             MOV   A,M       wait
604             RAR   .         -for the
605             JNC   NBUSY     -busy flag
606   BUSY
607             MOV   A,M       test for
608             RAR   .         -device busy
609             MOV   A,M       restore status
610             RNC   .         return if not busy
611             DCX   D         test for
612             MOV   A,D       -two disk
613             ORA   E         -revolutions
614             JNZ   BUSY      47 machine cycles
615             PUSH  H         save cmd address
616             INX   H         track register
617             MOV   D,M       save present track
618             LDA   DCREG     1791 control bits
619             XRI   RSTBIT    reset the 1791
620             STA   DCMD      -controller to
621             XRI   RSTBIT    -clear the
622             XTHL  .         -command busy
623             STA   DCMD      -fault
624             MVI   M,CLRCMD  force an interrupt
625             XTHL  .         restore the
626             MOV   M,D       -the track no
627             POP   H         restore the stack
628   BERROR
629             MVI   A,21Q     lost record
630             STC   .         -error flag
631             RET
632   *
633   *
634   MEASUR
635             LXI   D,0       initialize count
636             LXI   H,DSTAT   status port
637             MVI   C,INDEX   index bit flag
638   INDXHI
639             MOV   A,M       wait for
640             ANA   C         -index
641             JNZ   INDXHI    -pulse low
642   INDXLO
643             MOV   A,M       wait for
644             ANA   C         -index
645             JZ    INDXLO    -pulse high
646   INDXCT
647             INX   D         advance count
648             XTHL  .         four
649             XTHL  .         -dummy
650             XTHL  .         -instructions
651             XTHL  .         -for delay
652             MOV   A,M       wait
653             ANA   C         -for next
654             JNZ   INDXCT    -low index
655             RET   .         98 machine cycles
656   *
657   *
658   DENFIX
```

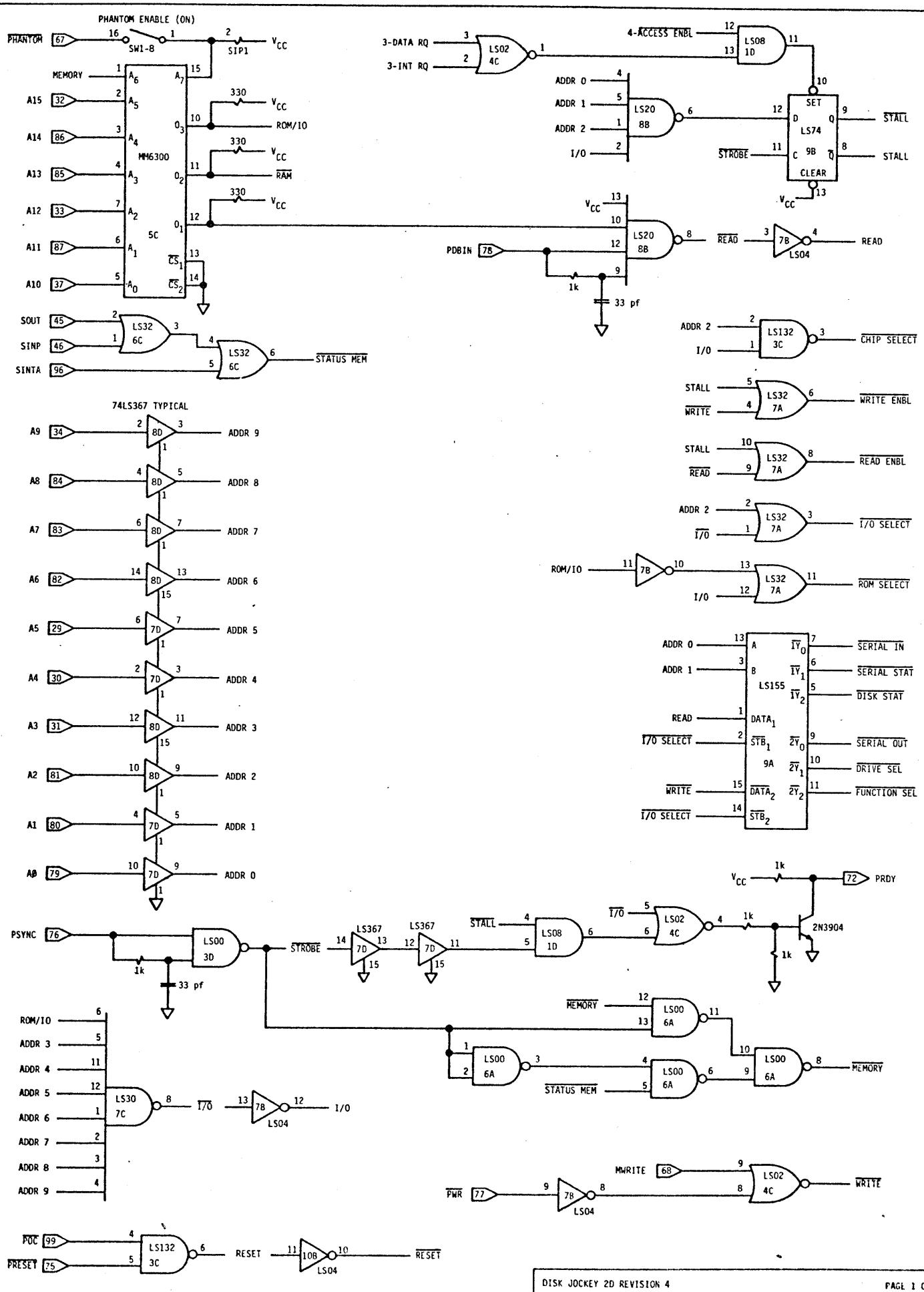
```
343:277 171
343:300 346 001
343:302 057
343:303 107
343:304 041 353 346
343:307 136
343:310 026 000
343:312 043
343:313 176
343:314 253
343:315 365
343:316 043
343:317 043
343:320 031
343:321 031
343:322 176
343:323 366 001
343:325 240
343:326 167
343:327 361
343:330 300
343:331 176
343:332 062 366 346
343:335 311


343:336
343:336 171
343:337 346 001
343:341 027
343:342 027
343:343 027
343:344 027
343:345 062 367 346
343:350 311


343:351
343:351 041 000 000
343:354
343:354 053
343:355 174
343:356 265
343:357 343
343:360 343
343:361 302 354 343
343:364 311


343:365 340
343:366 000
343:367 303


346:314
346:314 000:031

346:345 000 040
346:347 000 347
346:351 000
```

```
659             MOV   A,C       trim excess
660             ANI   1         -bits,
661             CMA   .         compliment
662             MOV   B,A       -B and save
663             LXI   H,DISK    new disk
664             MOV   E,M       get disk no
665             MVI   D,0       offset addr
666             INX   H         current disk
667             MOV   A,M       move to  ACC
668             XRA   E         compare w/new
669             PUSH  PSW       save status
670             INX   H         disk table
671             INX   H         -address
672             DAD   D         add the
673             DAD   D         offset
674             MOV   A,M       get parameters
675             ORI   1         make off density
676             ANA   B         set new density
677             MOV   M,A       update
678             POP   PSW       check for nd=cd
679             RNZ   .         new disk not old
680             MOV   A,M       update CDISK
681             STA   DCREG     -also
682             RET
683   *
684   *
685   SIDEFX
686             MOV   A,C       get the side bit
687             ANI   1         trim excess bits
688             RAL   .         move the bit
689             RAL   .         -to the side
690             RAL   .         -select bit
691             RAL   .         -position
692             STA   SIDE      save
693             RET
694   *
695   *
696   TIMOUT
697             LXI   H,0       time out delay
698   TILOOP
699             DCX   H         decrement
700             MOV   A,H       test for
701             ORA   L         -count zero
702             XTHL  .         long
703             XTHL  .         -NOP
704             JNZ   TILOOP
705             RET
706   *
707   *
708             DB    DBOOT/256 backward
709             DB    0         -jump
710   DVREND    DB    303Q      -instruction
711   *
712   *
713             AORG  RAM+2:314Q
714   STACK     DS    31Q
715   *
716   TIMER     DW    30:0000Q  head load time
717   DMAADR    DW    347:0000Q dma address
718   HDFLAG    DB    0         read header fl
```

```
346:352 376             719  DRVSEL  DB   376Q     drive select constant
346:353 000             720  DISK    DB   0        new drive
346:354 010             721  CDISK   DB   10Q      current drive
346:355 000             722  TZFLAG  DB   0        track zero indicator
346:356 003             723  DOPRAM  DB   3        drive 0 parameters
346:357 377             724  DOTRK   DB   377Q     drive 0 track no
346:360 003             725  D1PRAM  DB   3        drive 1 parameters
346:361 377             726  D1TRK   DB   377Q     drive 1 track no
346:362 003             727  D2PRAM  DB   3        drive 2 parameters
346:363 377             728  D2TRK   DB   377Q     drive 2 track no
346:364 003             729  D3PRAM  DB   3        drive 3 parameters
346:365 377             730  D3TRK   DB   377Q     drive 3 track no
346:366 003             731  DCREG   DB   3        current parameters
346:367 000             732  SIDE    DB   0        new side select
346:370 000             733  SECTOR  DB   0        new sector
346:371 000             734  TRACK   DB   0        new track
346:372 000             735  TRKNO   DB   0        disk
346:373 000             736  SIDENO  DB   0        -sector
346:374 000             737  SECTNO  DB   0        -header
346:375 000             738  SECLEN  DB   0        -data
346:376 000             739  CRCLO   DB   0        -buffer
346:377 000             740  CRCHI   DB   0
```

BLOCK DIAGRAM

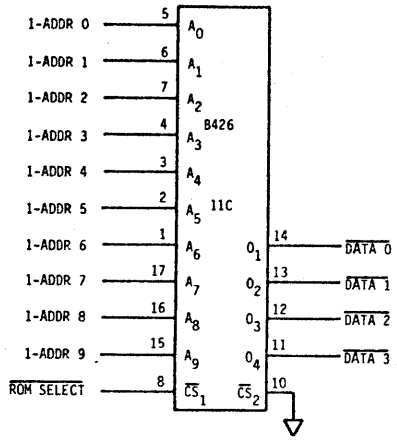


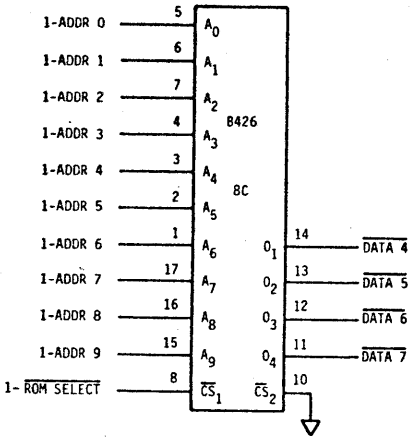


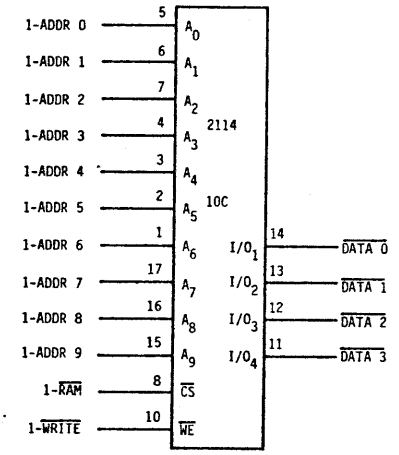
DISK JOCKEY 2D REVISION 4

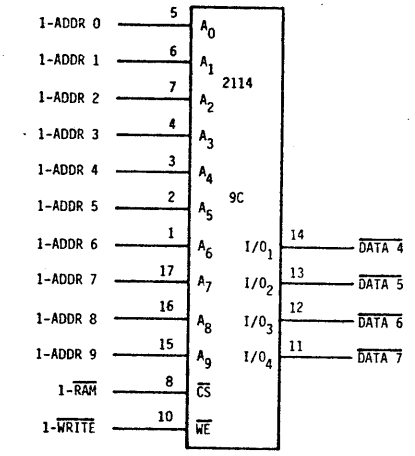
PAGE 1 OF 5

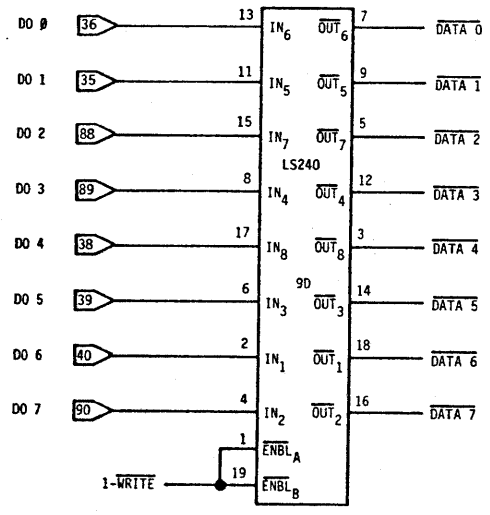
BOARD SELECT LOGIC,ADDRESS BUFFERS

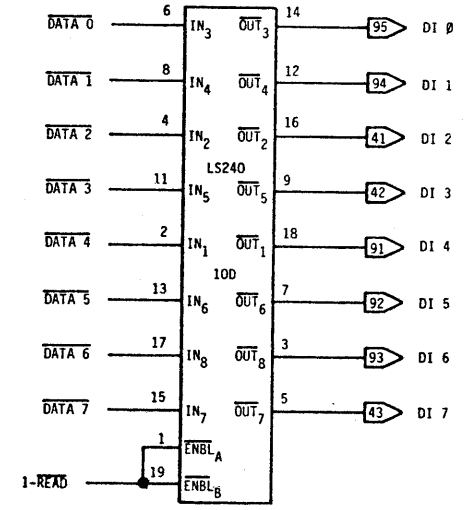


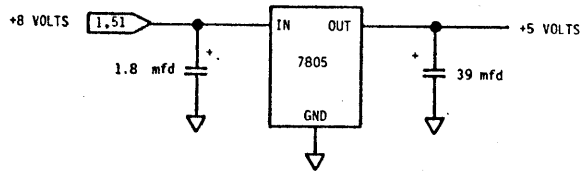


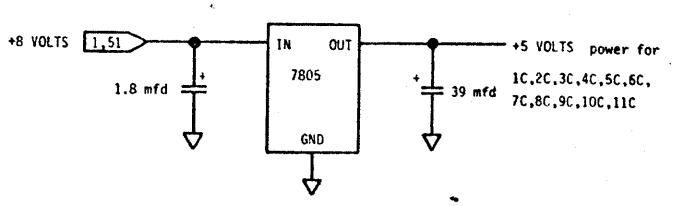


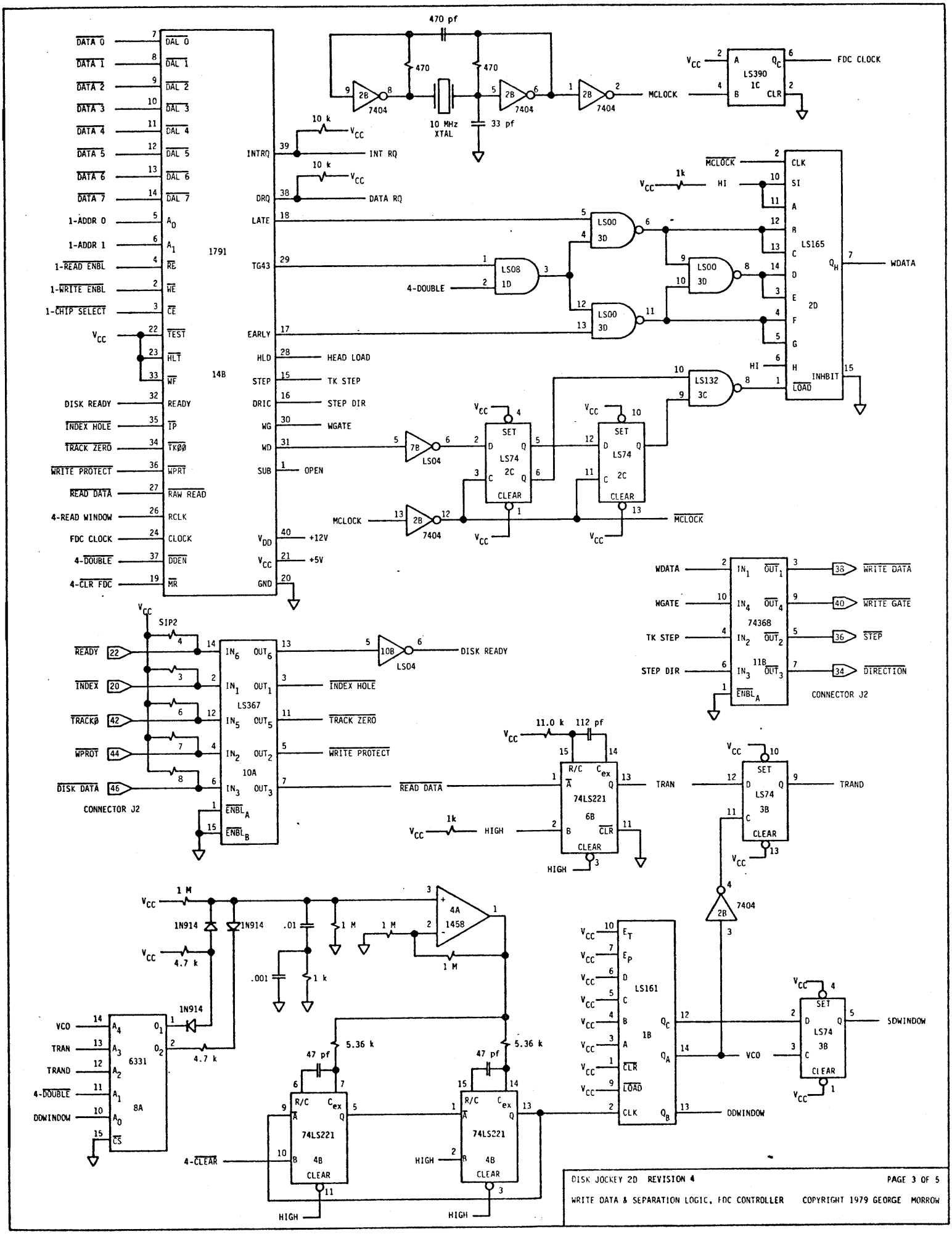


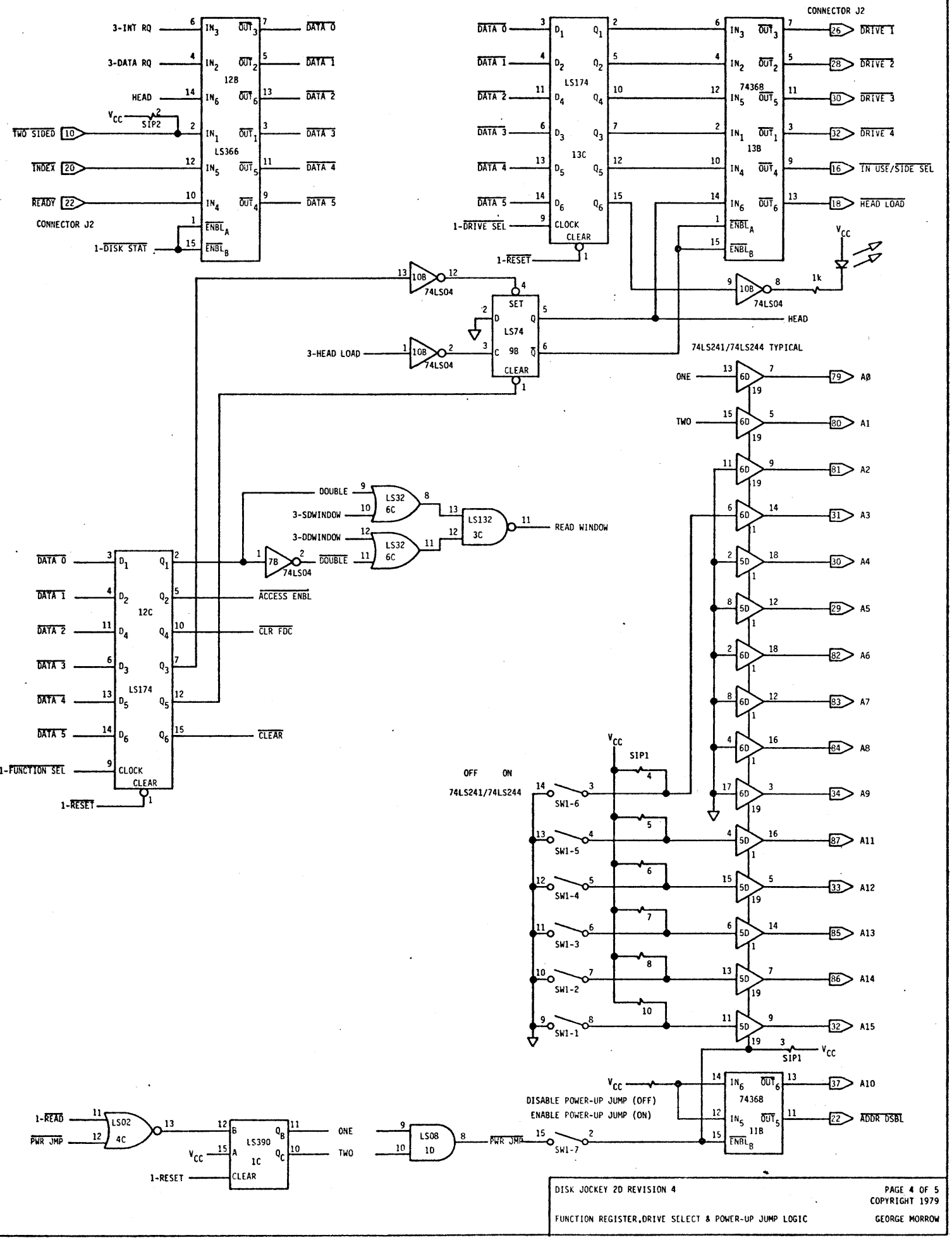









1791
14B
DATA 0
DATA 1
DATA 2
DATA 3
DATA 4
DATA 5
DATA 6
DATA 7
1-ADDR 0
1-ADDR 1
1-READ ENBL
1-WRITE ENBL
1-CHIP SELECT
DISK READY
INDEX HOLE
TRACK ZERO
WRITE PROTECT
READ DATA
4-READ WINDOW
FDC CLOCK
4-DOUBLE
4-CLR FDC
INTRQ
DRQ
LATE
TG43
EARLY
HLD
STEP
DRIC
WG
WD
SUB
INT RQ
DATA RQ
HEAD LOAD
TK STEP
STEP DIR
WGATE
OPEN
+12V
+5V
470 pf
470
10 MHz XTAL
33 pf
MCLOCK
FDC CLOCK
LS390
LS165
LS08
LS00
LS132
LS74
WDATA
INHBIT
4-DOUBLE
HI
LOAD
SIP2
LS367
CONNECTOR J2
DISK READY
LS04
WRITE DATA
WRITE GATE
STEP
DIRECTION
74368
11.0 k
112 pf
74LS221
TRAN
TRAND
HIGH
1 M
1N914
4.7 k
.01
.001
1 k
1458
6331
VCO
TRAN
TRAND
4-DOUBLE
DDWINDOW
47 pf
5.36 k
4-CLEAR
LS161
SDWINDOW
DDWINDOW
DISK JOCKEY 2D REVISION 4
PAGE 3 OF 5
WRITE DATA & SEPARATION LOGIC, FDC CONTROLLER
COPYRIGHT 1979 GEORGE MORROW

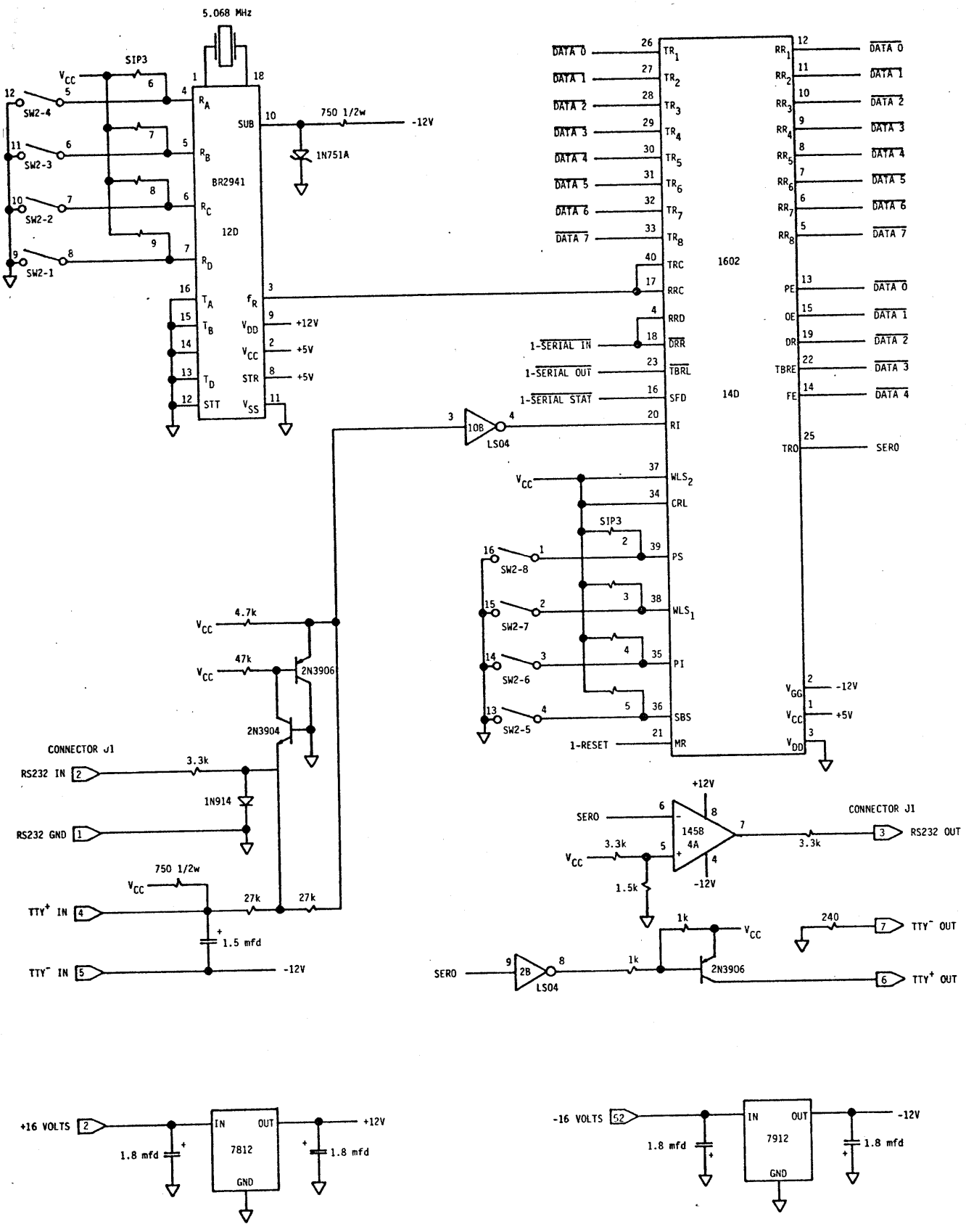


DISK JOCKEY 2D REVISION 4

PAGE 4 OF 5

FUNCTION REGISTER, DRIVE SELECT & POWER-UP JUMP LOGIC

GEORGE MORROW



DISK JOCKEY 2D REVISION 4

PAGE 5 OF 5

SERIAL INTERFACE LOGIC

Morrow Designs, Inc.

Thinker Toys™ 5221 Central Avenue, Richmond, CA 94804 (415) 524-2101

FIELD ENGINEERING MEMO

TO: All DISCUS Owners and Dealers

SUBJECT: Update for DISCUS Double Density Operating Systems

FROM: George Morrow

In a world of sell-and-forget, you may be gratified to learn that Morrow Designs sells and remembers. That's why we're offering an operating systems upgrade for all DISCUS Double Density Disk Drives. At cost. Or below.

Here's why: We found an anomaly in Western Digital's 1791 Floppy Disk Controller chip. Nothing major.

But under certain conditions, you could get an "error" from CP/M® Operating System (BDOS ERROR - "BAD SECTOR"). When this happens, certain information on the disk is no longer readable. But it's software correctible. So, we're correcting it. At cost. Or below.

Here's what you do:

For the Double Density DISCUS Controller Model B

a) If the 2708 EPROM has a label marked "B/V2", no action is required -- you have the latest version of the driver software which corrects the 1791 anomaly.

b) If the 2708 EPROM has no label, it must be replaced or reprogrammed: Send us a check for $15.00 or a 2708 EPROM which is erased and functional.

For the Double Density DISCUS Controller REVs 0, 1, 3 and 4

a) If you have 2.0, 2.1, or 2.2 CP/M, send us a clean diskette or your check for $7.00. We'll send back the latest 2.2 CP/M diskette. (Remember to include the serial number with your reply.)

CBIOS Vers 3.0 is o/c

b) If you have Lifeboat 1.4 CP/M, it's going to be a little more expensive. $42.00. For that we'll send you the latest version of CP/M 2.2 on a diskette and complete documentation for the upgrade. Or, if you send us a clean diskette, the cost is only $35.00. Normally, the cost of the documentation alone is $35.00. Again, we must have the serial number of your CP/M in order to make this upgrade.

NOTES ON LIFEBOAT 2D CP/M FOR THINKER TOYS

There are several features of Lifeboat's 2D version of CP/M with which users accustomed to single density CP/M on 8 inch drives may not be familiar. These features will be explained below.

ASSIGNING DENSITY

2D CP/M must be aware of the density of a diskette before it can successfully perform a read or write operation. The command file "DENSITY" allows the user to inform CP/M of the density which a given drive will be assigned. If a wrong density diskette is placed in a drive, and that drive is subsequently accessed, the system will fall into an irrecoverable error.

The default assignment of densities in the production CP/M disk is: A, C, and D drives = Double Density; B drive = Single Density

To change this arrangement temporarily, type DENSITY and follow the prompts. To make a permanent change, follow the instructions contained in the "ASM" file TTUSER.

FORMATTING A DISKETTE

The two command programs, FORMAT1D and FORMAT2D, will format a diskette in single and dual density respectively. FORMAT1D will write sector headers for 26 sectors per track, 128 bytes per sector. FORMAT2D will write sector headers for 26 sectors per track, 256 bytes per sector. A disk can be formatted in either density regardless of the density assigned to the formatting drive under CP/M-- however the drive will not be able to read the disk it has just formatted unless the drive has been assigned the proper density.

USER and TTUSER

2D CP/M dedicates the equivalent of three single density sectors, or one and a half pages of memory, for user I/O. In single density CP/M this was subsumed under the CBIOS. In a 24K system, locations 5E80 to 5FFF contain user I/O.

To alter CP/M size or change the I/O routines or both from the original production configuration, the USER or TTUSER file must be edited to reflect the desired changes and re-assembled to create a HEX file of the new I/O. The active I/O on production diskettes was assembled from the file called TTUSER, while a simpler file called USER provides an alternative specimen which does not implement I/O byte.

Both source files are amply commented. It should be noted that to retain the file TTUSER as the actual I/O driver after a MOVCPM command, only the EQUATE labeled "MSIZE" need be changed in the edit prior to re-assembly. However, after re-assembly, the PRN file of TTUSER should be examined in order to find the new "OFFSET" variable which will be needed in order to overlay the driver onto the new CP/M system.

A simple MOVCPM N command will create a new CP/M of "N" K size with only the console driver implemented. Thus no overlay is necessary if the only device CP/M is to be aware of is the console terminal.

RECONFIGURING A SYSTEM

Once the USER or TTUSER file has been edited and re-assembled, the following procedure may used to incorporate the new drivers into CP/M:

-Note the OFFSET of the new CP/M from the PRN file of USER or TTUSER.

-Type "MOVCPM N *", where N represents the memory size in kilobytes. The smallest CP/M size is 17K.

-Type "SAVE 35 CPMN.COM", with N as above.

-Type "DDT CPMN.COM", with N as above.

-Type "IUSER.HEX", or "ITTUSER.HEX"

-Type "ROFFSET", where OFFSET is the value obtained from the PRN file USER or TTUSER. For a 24K system, one would type "RC380; for a 32K system, RA380 etc.

-Type control C

-Type "SYSGEN"; CP/M will request for the source drive.

-Type Return; the source for the new system is already in memory.

-Type the destination drive-- A,B,C or D. Make sure that the drive in question has a disk formatted in the proper density.

-Reset the system and boot the new disk.

SAVEUSER

The SAVEUSER command places whatever I/O that happens to be in memory onto the CP/M boot program. Thus new I/O drivers can be patched in from a front panel or monitor, and made permanent through the SAVEUSER command. A subsequent MOVCPM command will overwrite this patch, so once a driver has been tested it should be incorporated into a USER source file as soon as possible. The memory locations to patch in I/O drivers can be found in the listing of TTUSER included with the CP/M diskette.

AUTO.COM

The AUTO.COM function does not work at this time.

```
      1:
 2:
 3:                   ;*********************************************************
 4:                   ;       SAMPLE USER AREA
 5:                   ;       FOR THINKER TOYS 2D CONTROLLER AND SWITCHBOARD
 6:                   ;*********************************************************
 7:                   ;
 8:                   ;       THIS DRIVER IMPLEMENTS I/O BYTE
 9:                   ;THE CONSOLE DEVICE IS AN RS232 OR TTY TERMINAL
10:                   ;ATTACHED TO THE I/O CONNECTOR OF THE DJ2D BOARD
11:                   ;THE LIST DEVICE IS AN RS232 OR TTY PRINTER
12:                   ;ATTACHED TO THE SECOND SERIAL PORT ON THE SWITCHBOARD
13:                   ;       THESE ROUTINES CAN BE USED AS A BASIS
14:                   ;FOR THE DEVELOPMENT OF YOUR OWN I/O
15:                   ;       NOTE ALSO LOCATIONS OF WHERE TO SET THE
16:                   ;DENSITY ON EACH DRIVE AND SETTING OF THE MODE BYTE
17:                   ;WHICH CAN RUN A FILE NAMED "AUTO.COM" ON EITHER
18:                   ;COLD OR WARM BOOT.
19:                   ;       FOR EXAMPLE, RENAME BASIC.COM TO AUTO.COM
20:                   ;USING THE COMMAND "REN AUTO.COM=BASIC.COM" AND
21:                   ;HAVE BASIC AUTOMATICALLY ACTIVATED ON EACH COLD BOOT.
22:                   ;
23:                   ;*********************************************************
24:                   ;       SYSTEM EQUATES
25:                   ;*********************************************************
26:                   ;
27:  0018 =           MSIZE   EQU     24              ;CP/M SYSTEM SIZE IN KBYTES
28:  2000 =           BIAS    EQU     (MSIZE-16)*1024
29:  4600 =           CPMB    EQU     2600H+BIAS      ;LOCATION OF CCP
30:  4E00 =           BDOS    EQU     2E00H+BIAS      ;LOCATION OF BDOS
31:  5B00 =           BIOS    EQU     3B00H+BIAS      ;LOCATION OF BIOS
32:  5E80 =           USER    EQU     BIOS+380H       ;START OF USER AREA
33:  C380 =           OFFSET  EQU     1E80H-BIOS      ;TO SYSGEN IMAGE
34:                   ;
35:                   ;*********************************************************
36:                   ;       DISK PARAMETERS - DOUBLE DENSITY DISK JOCKEY
37:                   ;*********************************************************
38:                   ;       ON DISK                 IN SYSGEN       IN 24K SYSTEM
39:                   ;       TRACK   SECTOR          ADDRESS
40:                   ;BOOT   0       1                900H           0E700H
41:                   ;CCP    1       1               0980H           4600H
42:                   ;BDOS   1       17              1180H           4E00H
43:                   ;BIOS   1       43              1E80H           5B00H
44:                   ;MODE   1       49              21FFH           5E7FH
45:                   ;USER   1       50-52           2200H           5E80H
46:                   ;TOP OF SYSTEM                  237FH           5FFFH
47:                   ;
48:                   ;DOUBLE DENSITY SKIP TABLE
49:                   ;       1,2,19,20,37,38,3,4,21,22,39,40,5,6,23,24,41,42
50:                   ;       7,8,25,26,43,44,9,10,27,28,45,46,11,12,29,30,47,48
51:                   ;       13,14,31,32,49,50,15,16,33,34,51,52,17,18,35,36
52:                   ;
53:                   ;*********************************************************
54:                   ;MODE BYTE OPTIONS AND DENSITY SETTINGS OF VARIOUS DRIVES
55:                   ;*********************************************************
56:                   ;
57:  5E78                     ORG     USER-8          ;5E78H IN 24K SYSTEM
58:  5E78 01000101   DNSTY:  DB      1,0,1,1         ;DRIVE B SD, OTHERS DD
59:                           ;B0=DENSITY             0=SNGL, 1=DBL
60:  5E7C 000000             DB      0,0,0           ;RESERVED
```

```
5E7F 00         MODE:   DB      0               ;MODE BYTE
                        ;BIT0=1 DOES AUTO ON COLD BOOT
                        ;BIT1=1 DOES AUTO ON WARM BOOT
                ********************************************************
                *       SAMPLE USER AREA
                ********************************************************
5E80                    ORG     USER            ;5E80H IN DIST SYSTEM

                ********************************************************
                * JUMP TABLE - JMPS MUST REMAIN HERE, IN SAME ORDER
                ********************************************************

5E80 C33C5F             JMP     INIT            ;INITIALIZATION
5E83 C3985E             JMP     CONST           ;CONSOLE STATUS
5E86 C3A45E             JMP     CONIN           ;CONSOLE INPUT
5E89 C3B65E             JMP     CONOUT          ;CONSOLE OUTPUT
5E8C C3D15E             JMP     LIST            ;LIST OUTPUT
5E8F C3C65E             JMP     PUNCH           ;PUNCH OUTPUT
5E92 C3BC5E             JMP     READER  ;READER INPUT
5E95 C3335F             JMP     PRST            ;PRINTER STATUS

0004 =          CDISK   EQU     4       ;current disk storage location
0003 =          IOBYTE  EQU     3H      ;iobyte storage location

                ************************************************************
                *                                                          *
                * Iobyte allows selection of different I/O devices. It     *
                * can be initialized in any way by changing the equate     *
                * bellow.                                                  *
                * Initial iobyte is currently defined as :                 *
                * console = tty                                            *
                * reader = tty                                             *
                * punch = tty                                              *
                * list = tty                                               *
                *                                                          *
                ************************************************************

0000 =          INTIOBY EQU     0       ;initial iobyte,

                ************************************************************
                *                                                          *
                * The following equates reference the disk jockey/2d       *
                * controller board. If your controller is non-standard     *
                * then all the equates can be changed by re-assigning the  *
                * value of ORIGIN to be the starting address of your       *
                * controller.                                              *
                *                                                          *
                ************************************************************

E000 =          ORIGIN  EQU     0E000H  ;disk jockey/2d beginning address
E003 =          INPUT   EQU     ORIGIN+3        ;serial input routine
E006 =          OUTPUT  EQU     ORIGIN+6        ;serial output routine
E021 =          TSTAT   EQU     ORIGIN+21H      ;serial device status routine
000D =          ACR     EQU     0DH     ;carriage return
000A =          ALF     EQU     0AH     ;line feed
E006 =          COTTY   EQU     OUTPUT  ;default character output
E003 =          CITTY   EQU     INPUT   ;default character input

                ************************************************************
                *
```

```
                ***************************************************************
                *                                                             *
                * const: get the status for the currently assigned console    *
                *        device. The console device can be gotten from        *
                *        iobyte, then a jump to the correct console status    *
                *        routine is performed.                                *
                *                                                             *
                ***************************************************************

5E98 21045F     CONST   LXI     H,CSTBLE        ;beginning of jump table
5E9B C3A75E             JMP     CONIN1  ;select corre jump

                ***************************************************************
                *                                                             *
                * csreader: if the console is assigned to the reader then     *
                *           a jump will be made here, where another jump      *
                *           will occur to the correct reader status.          *
                *                                                             *
                ***************************************************************

5E9E 210C5F     CSREADR LXI     H,CSRTBLE       ;beginning of reader status tal
5EA1 C3BF5E             JMP     READERA

                ***************************************************************
                *                                                             *
                * conin: take the correct jump for the console input          *
                *        routine. The jump is based on the two least sig-     *
                *        nificant bits of iobyte.                             *
                *                                                             *
                ***************************************************************

5EA4 21DC5E     CONIN   LXI     H,CITBLE        ;beginning of character input t

                *
                * entry at coninl will decode the two least significant bits
                * of iobyte. This is used by conin,conout, and const.
                *

5EA7 3A0300     CONIN1  LDA     IOBYTE
5EAA 17                 RAL

                *
                * entry at seldev will form an offset into the table pointed
                * to by H&L and then pick up the address and jump there.
                *

5EAB E606       SELDEV  ANI     6H      ;strip off unwanted bits
5EAD 1600               MVI     D,0     ;form affset
5EAF 5F                 MOV     E,A
5EB0 19                 DAD     D       ;add offset
5EB1 7E                 MOV     A,M     ;pick up high byte
5EB2 23                 INX     H
5EB3 66                 MOV     H,M     ;pick up low byte
5EB4 6F                 MOV     L,A     ;form address
5EB5 E9                 PCHL            ;go there !

                ***************************************************************
                *                                                             *
                * conout: take the proper branch address based on the two     *
                *         least significant bits of iobyte.                   *
                *                                                             *
                ***************************************************************
```

```
5EB6 21E45E    CONOUT  LXI     H,COTBLE        ;beginning of the character out tabl
5EB9 C3A75E            JMP     CONIN1  ;do the decode

               ***************************************************
               *                                                 *
               * reader: select the correct reader device for input. The  *
               *         reader is selected from bits 2 and 3 of iobyte.  *
               *                                                 *
               ***************************************************

5EBC 21FC5E    READER  LXI     H,RTBLE ;beginning of reader input table

               *
               * entry at readera will decode bits 2 & 3 of iobyte, used
               * by csreader.
               *

5EBF 3A0300    READERA LDA     IOBYTE

               *
               * entry at reader1 will shift the bits into position, used
               * by list and punch.
               *

5EC2 1F        READR1  RAR
5EC3 C3AB5E            JMP     SELDEV

               ***************************************************
               *                                                 *
               * punch: select the correct punch device. The seection     *
               *        comes from bits 4&5 of iobyte.           *
               *                                                 *
               ***************************************************

5EC6 21F45E    PUNCH   LXI     H,PTBLE ;beginning of punch table
5EC9 3A0300            LDA     IOBYTE

               *
               * entry at pnch1 rotates bits a little more in prep for
               * seldev, used by list.
               *

5ECC 1F        PNCH1   RAR
5ECD 1F                RAR
5ECE C3C25E            JMP     READR1

               ***************************************************
               *                                                 *
               * list: select a list device based on bits 6&7 of iobyte   *
               *                                                 *
               ***************************************************

5ED1 21EC5E    LIST    LXI     H,LTBLE ;beginning of the list device routines
5ED4 3A0300            LDA     IOBYTE
5ED7 1F                RAR
5ED8 1F                RAR
5ED9 C3CC5E            JMP     PNCH1

               ***************************************************
               *                                                 *
               * If customizing I/O routines is being performed, the      *
               * table below should be modified to reflect the changes.   *
               * all I/O devices are decoded out of iobyte and the jump   *
```

```
                ************************************************************
                *                                                          *
                * If customizing I/O routines is being performed, the      *
                * table below should be modified to reflect the changes.   *
                * all I/O devices are decoded out of iobyte and the jump   *
                * is taken from the following tables.                      *
                *                                                          *
                ************************************************************


                *
                * console input table
                *

5EDC 03E0       CITBLE  DW      CITTY   ;input from tty (currently assigned by intioby,input from 2d)
5EDE 1F5F               DW      CICRT   ;input from crt (currently SWITCHBOARD serial port 1)
5EE0 BC5E               DW      READER  ;input from reader (depends on reader selection)
5EE2 1F5F               DW      CIUC1   ;input from user console 1 (currently SWITCHBOARD serial port 1)

                *
                * console output table
                *

5EE4 06E0       COTBLE  DW      COTTY   ;output to tty (currently assigned by intioby,output to 2d)
5EE6 145F               DW      COCRT   ;output to crt (currently SWITCHBOARD serial port 1)
5EE8 D15E               DW      LIST    ;output to list device (depends on bits 6&7 of iobyte)
5EEA 145F               DW      COUC1   ;output to user console 1 (currently SWITCHBOARD serial port 1)

                *
                * list device table
                *

5EEC 06E0       LTBLE   DW      COTTY   ;output to tty (currently assigned by intioby,output to 2d)
5EEE 145F               DW      COCRT   ;output to crt (currently SWITCHBOARD serial port 1)
5EF0 145F               DW      COLPT   ;output to line printer (currently SWITCHBOARD serial port 1)
5EF2 145F               DW      COUL1   ;output to user line printer 1 (currently SWITCHBOARD serial port 1)

                *
                * punch device table
                *

5EF4 06E0       PTBLE   DW      COTTY   ;output to the tty (currently assigned by intioby,output to 2d)
5EF6 145F               DW      COPTP   ;output to paper tape punch (currently SWITCHBOARD serial port 1)
5EF8 145F               DW      COUP1   ;output to user punch 1 (currently SWITCHBOARD serial port 1)
5EFA 145F               DW      COUP2   ;output to user punch 2 (currntlly SWITCHBOARD serial port 1)

                *
                * reader device input table
                *
```

```
                        * reader device input table
                        *
5EFC 03E0       RTBLE   DW      CITTY   ;input from tty (currently assigned by intioby, input from 2d)
5EFE 1F5F               DW      CIPTR   ;input from paper tape reader (currently SWITCHBOARD serial port 1)
5F00 1F5F               DW      CIUR1   ;input from user reader 1 (currently SWITCHBOARD serial port 1)
5F02 1F5F               DW      CIUR2   ;input from user reader 2 (currently SWITCHBOARD serial port 1)

                        *
                        * console status table
                        *
5F04 2B5F       CSTBLE  DW      CSTTY   ;status of tty (currently assigned by intioby, ststus from 2d)
5F06 335F               DW      CSCRT   ;status from crt (currently SWITCHBOARD serial port 1)
5F08 9E5E               DW      CSREADR ;status from reader (depends on reader device )
5F0A 335F               DW      CSUC1   ;status from user console 1 (currently SWITCHBOARD serial port 1)

                        *
                        * status fromreader device
                        *
5F0C 2B5F       CSRTBLE DW      CSTTY   ;status from tty (currently assigned by intioby, status of 2d)
5F0E 335F               DW      CSPTR   ;status from paper tape reader (currently SWITCHBOARD serial port 1)
5F10 335F               DW      CSUR1   ;status from user reader 1 (currently SWITCHBOARD serial port 1)
5F12 335F               DW      CSUR2   ;status of user reader 2 (currently SWITCHBOARD serial port 1)

                        ************************************************************
                        *                                                          *
                        * The following equates set output device to output to     *
                        * the SWITCHBOARD serial port 1.                           *
                        *                                                          *
                        ************************************************************

5F14 =          COCRT   EQU     $       ;output from crt
5F14 =          COUC1   EQU     $       ;output from user console 1
5F14 =          COUL1   EQU     $       ;output from user line printer 1
5F14 =          COPTP   EQU     $       ;output from paper tape punch
5F14 =          COUP1   EQU     $       ;output from user punch 1
5F14 =          COUP2   EQU     $       ;output from user punch 2
5F14 DB02       COLPT   IN      2       ;output from line printer,get status
5F16 E680               ANI     80H     ;wait until ok to send
5F18 CA145F             JZ      COLPT
5F1B 79                 MOV     A,C     ;output the character
5F1C D301               OUT     1
5F1E C9                 RET

                        ************************************************************
                        *                                                          *
                        * The following equates set the input from the devices to  *
                        * come from the SWITCHBOARD serial port 1                  *
```

```
                ***************************************************
                *                                                 *
                * The following equates set the input from the devices to *
                * come from the SWITCHBOARD serial port 1         *
                *                                                 *
                ***************************************************

5F1F =          CIUC1   EQU     $       ;input from user console 1
5F1F =          CICRT   EQU     $       ;input from crt
5F1F =          CIUR1   EQU     $       ;input from user reader 1
5F1F =          CIUR2   EQU     $       ;input from user reader 2
5F1F DB02       CIPTR   IN      2       ;input from paper tape reader, get status
5F21 E640               ANI     40H     ;wait for character
5F23 CA1F5F             JZ      CIPTR
5F26 DB01               IN      1
5F28 E67F               ANI     7FH     ;strip off the parity
5F2A C9                 RET

                ***************************************************
                *                                                 *
                * console status routines, test if a character has arrived *
                *                                                 *
                ***************************************************

5F2B CD21E0     CSTTY   CALL    TSTAT   ;status from disk jockey 2d
5F2E 3E00       STAT    MVI     A,0     ;prep for zero return
5F30 C0                 RNZ             ;nothing found
5F31 3D                 DCR     A       ;return with 0FFH
5F32 C9                 RET

                ***************************************************
                *                                                 *
                * The following equates cause the devices to get status *
                * from the SWITCHBOARD serial port 1.             *
                *                                                 *
                ***************************************************

5F33 =          PRST    EQU     $       ;STATUS OF PRINTER
5F33 =          CSUR1   EQU     $       ;status of user reader 1
5F33 =          CSUR2   EQU     $       ;status of user reader 2
5F33 =          CSPTR   EQU     $       ;status of paper tape reader
5F33 =          CSUC1   EQU     $       ;status of user console 1
5F33 DB02       CSCRT   IN      2       ;status from crt, get status
5F35 E640               ANI     40H     ;strip of data ready bit
5F37 EE40               XRI     40H     ;make correct polarity
5F39 C32E5F             JMP     STAT    ;return proper indication

                ***************************************************
                *                                                 *
                * THE FOLLOWING IS A TERMINAL INITIALIZATION ROUTINE. *
                * IT CAN BE USED TO PERFORM ANY INITIALIZATION YOU MAY *
                * REQUIRE. CURRENTLY IT IS NOT NEEDED.            *
                *                                                 *
                ***************************************************

5F3C =          INIT    EQU     $       ;
5F3C 3E00               MVI     A,INTIOBY
5F3E 320300             STA     IOBYTE
5F41 C9                 RET

A>
```