MORROW DESIGNS

User's Manual

DISK JOCKEY 2D (tm)

MODEL B

REVISION 2

Table of Contents

Introduction 1
Programming Specifications 3
ROM Jump Table 3
Serial I/O 4
Disk I/O 5
ROM Subroutines. 7
Error Bits Recap 12
Diskette Initialization. 13
Utilizing Disk Jockey Firmware 14
Sample Read Routine. 15
Sample Write Routine 17
Disk System Software 19
I/O Connectors P1 & P2 20
Patches for CP/M*. 21
Hardware Level Registers 25
Bank Selection 32
Interrupt Logic. 33
Bootstrap LED Indicator. 34
Phantom Logic. 35
4 MHz Operation. 36
Power-On Jump Logic. 37
Drive Cable Conventions. 38
Serial I/O Switch Settings 39
Concise DIP Switch Reference 40
Parts List 41
Assembly Instructions. 45
Parts Installation 48
Initial Check-out and Power-up 52
Concise Firmware Memory Map. 53
Software listings. 54
Cold Boot Loader 55
CBIOS Drivers for CP/M 57
Disk Jockey 2D Firmware. 64
Schematics 94

FOR TECHNICAL SUPPORT

OR REPAIR SERVICE

CALL

(415) 524-2104

User's Manual

DISK JOCKEY 2/D tm

INTRODUCTION

The Morrow Designs DISK JOCKEY 2/D Model B (DJ) board features four 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.

4. Bank select logic which allows the board to be enabled or disabled under software control. This logic also can be programed to force the board to be enabled or disabled during power-on/reset sequences.

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 attaching 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 EPROM memory on the DJ board. This EPROM occupies a 1024 byte block of S-100 bus memory address space. A 1024 byte RAM is also provided which is used by the EPROM 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 actual addresses where the I/O registers,EPROM, 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 EPROM, the next 8 bytes constitute the memory mapped I/O, and the last 1024 bytes contain the RAM buffer.

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 0 of the onboard ROM. In boards with standard address decoding PROMS, A = 340:000Q (E000H). The address to call for the utility routines are then:

ADDRESS	STANDARD VALUE		SYMBOLIC VALUE	FUNCTION
	Octal	Hex		
A	340:000	E000	DBOOT	DOS bootstrap routine
A+3	340:003	E003	TERMIN	Serial input
A+6	340:006	E006	TRMOUT	Serial output
A+9	340:011	E009	TKZERO	Recalibrate (seek to TRK0)
A+12	340:014	E00C	TRKSET	Seek
A+15	340:017	E00F	SETSEC	Select sector
A+18	340:022	E012	SETDMA	Set DMA address
A+21	340:025	E015	DREAD	Read a sector of disk data
A+24	340:030	EO18	DWRITE	Write a sector of disk data
A+27	340:033	E01B	SELDRV	Select a disk drive
A+30	340:036	E01E	TPANIC	Test for panic character
A+33	340:041	E021	TSTAT	Serial status input
A+36	340:044	EO24	DMAST	Read current DMA address
A+39	340:047	E027	STATUS	Disk status input
A+42	340:052	E02A	DSKERR	Loop to strobe error LED
A+45	340:055	E02D	SETDEN	Set density
A+48	340:060	E030	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

GENERAL

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.

TERMIN

This subroutine is used to collect input characters from a terminal which is connected to the serial port on the board. The routine waits 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. This routine will not return until a character arrives from the terminal.

TRMOUT

This subroutine is used to transmit characters to a terminal that is connected to the serial port on the board. The routine waits until the TBRE bit in the UART's status register is high. When this bit is high, the data in the C register of the CPU is transfered to the UART's system bus register. This automatically resets the TBRE bit.

TPANIC

This subroutine is used to detect the presence of a "panic" character in the input data stream from the terminal. A program which uses this routine must load the C register with the desired "panic" character. If the UART has collected a character (i.e. the DR bit of the UART's status register is high) and it matches the character in the C register, the routine SETS the ZERO flag of the CPU's FLAGS register. On the other hand, the routine will CLEAR this flag if 1) the DR bit is not high or 2) the character in the UART's system bus register does not match the character in the C register.

TSTAT

This subroutine is used to test the condition of the DR bit in the UART's status register. If the DR bit is high, TSTAT will SET the ZERO flag of the CPU's FLAGS register. If the DR bit is low, TSTAT will CLEAR the ZERO flag of the CPU's FLAGS register. The routine does NOT alter the state of 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 / Fujitsu 8866 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 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
	1024	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 1791/8866 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.

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

Check for error conditions.

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. If the C register contains a zero, the carry flag is set and the routine returns immediately. If the C register is non-zero, the low order five bits are transfered to the RAM location SECTOR, the carry flag is cleared and the routine 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 the 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 1024 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 1024 byte data transfer to or from the disk would cause memory references to the I/O registers 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. Accordingly, the data in C is trimmed to the low order two bits and stored in the RAM location 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 0 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 0 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 1791/8866 Floppy Disk Controller operates in two modes: single density FM (Frequency Modulation) mode or double density MFM (Modified Frequency Modulation) mode. Bit 0 of the C register determines what density the 1791/8866 will operate in when the next disk transfer operation is initiated (0=single,1=double). Care must be exercised in the use of this routine. Under certain conditions, if the density is changed in between disk transfers that occur on the same track, the micro-program that the 1791/8866 controller executes could fall into an error loop from which it could not recover. In such a case the system would have to be reset before further disk operations could be performed. The density mode of the 1791/8866 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 1791/8866. 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

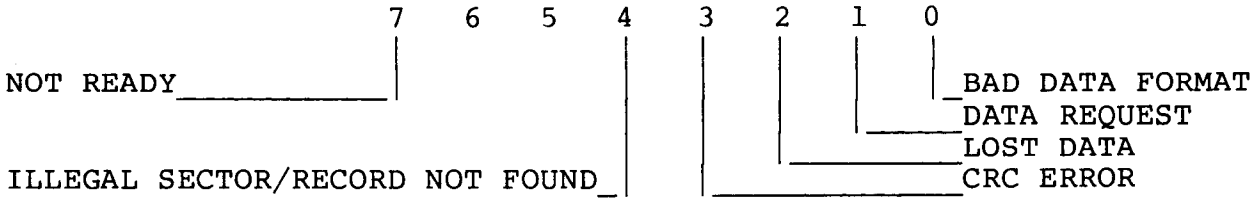
half diskette revolutions) than it would be if the initial assumption concerning the density was wrong.

TKZERO - This subroutine positions the read/write head to the outer-most track of the diskette: track 00. 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 1791/8866 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 1791/8866 controller cannot read the header information in either density, its status is copied into the CPU's A register, the head of the drive is positioned over track zero, and the operation is terminated with the carry set. When the Disk Jockey firmware positions the head to a new track, it reads a header both to determine the proper density and to find out the length and number of the sectors on the new track. The DJ RAM location SECLEN is updated

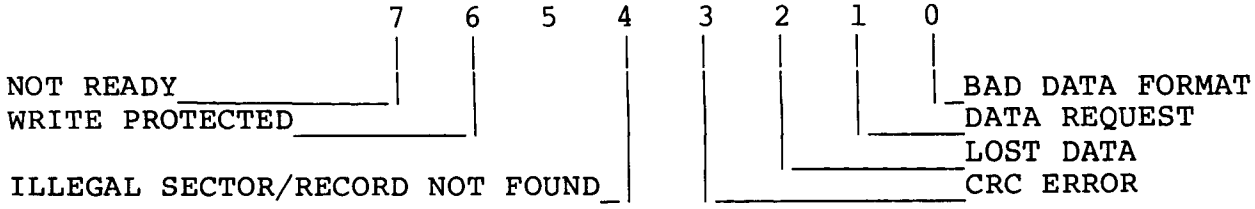
during read header operations and contains encoded data that determines both the number and the size of sectors on the current track. After (possibly) positioning the head the firmware takes the sector address determined by the most recent set sector operation and compares it to the total number of sectors on the current track. If the desired sector is too large, 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 1791/8866 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 1791/8866 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

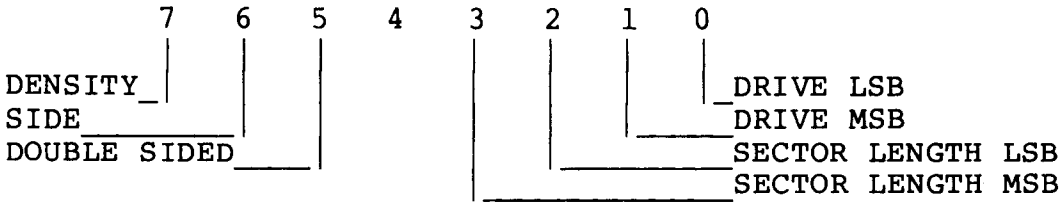


DBOOT - Branching to this routine will initiate a bootstrap load operation from the floppy disk. 128 bytes of data will be read (single density mode) into the first half of the 4th page of the Disk Jockey RAM (normally 344:000Q or E400H). The bootstrap routine terminates with a branch to the first location 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 to insure that the system is stable. 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, the density of the data during the most recent disk transfer operation, and whether the drive selected during the most recent disk operation was double sided WITH double sided media in place. 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

DENSITY BIT	
0	SINGLE
1	DOUBLE

DOUBLE SIDED = 1 Indicates double sided drive and diskette

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.

RECAP OF REGISTER A ERROR BITS

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

Register A Error Codes

"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									

DISKETTE INITIALIZATION

Before a new diskettte 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 in the section above can be used to write these header fields. This is a safety measure to ensure that an erroneous branch to the firmware EPROM 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. CP/M diskettes furnished by Morrow Designs contain a command called FORMT# to allow the user to format diskettes in any of the four IBM compatable formats.

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.

Suppose sectors 5, 6, 7 and 8 of track 12, drive 1 are to be readto or from memory starting a location 7:000Q (700H). The following programs will do this:

Example of Disk Read

```
001:000 061 356 346      1    READ     LXI   SP,200H    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
```

Utilizing Disk Jockey Firmware

Example of Disk Read

```
0100    31 EE E6        1   READ    LXI   SP,200H   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
```

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     1    WRITE    LXI   SP,200H     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
001:013 001 000 177     6             LXI   B,8000H-100H set initial adrs.
001:016 315 011 342     7             CALL  SETDMA
001:021 076 004         8             MVI   A,4         initial track number
001:023 062 112 001     9    TLOOP    STA   TEMP        save track number
001:026 117             10            MOV   C,A         seek to correct trk
001:027 315 313 342     11            CALL  TRKSET
001:032 001 001 032     12            LXI   B,32:001Q   sector count&number
001:035 305             13   SLOOP    PUSH  B           save sect and count
001:036 315 352 341     14            CALL  DMAST       get current address
001:041 041 000 001     15            LXI   H,100H      update to next sect
001:044 011             16            DAD   B
001:045 345             17            PUSH  H           move address to B-C
001:046 301             18            POP   B
001:047 315 011 342     19            CALL  SETDMA      set up new address
001:052 301             20            POP   B           restore sect cnt&num
001:053 305             21            PUSH  B
001:054 315 166 342     22            CALL  SETSEC      set up next sector
001:057 315 123 342     23            CALL  DWRITE      write the data
001:062 332 107 001     24            JC    ERROR       test for error
001:065 301             25            POP   B           recover sect cnt&num
001:066 014             26            INR   C           update sector
001:067 005             27            DCR   B           update count
001:070 302 035 001     28            JNZ   SLOOP
001:073 072 112 001     29            LDA   TEMP        get current track
001:076 074             30            INR   A           update track
001:077 376 007         31            CPI   7           check if all done
001:101 302 023 001     32            JNZ   TLOOP       continue to next trk
001:104 303 104 001     33   DONE     JMP   DONE
001:107 303 107 001     34   ERROR    JMP   ERROR       error exit
001:112 000             35   TEMP     DB    0           track storage
                        36
```

WRITE:

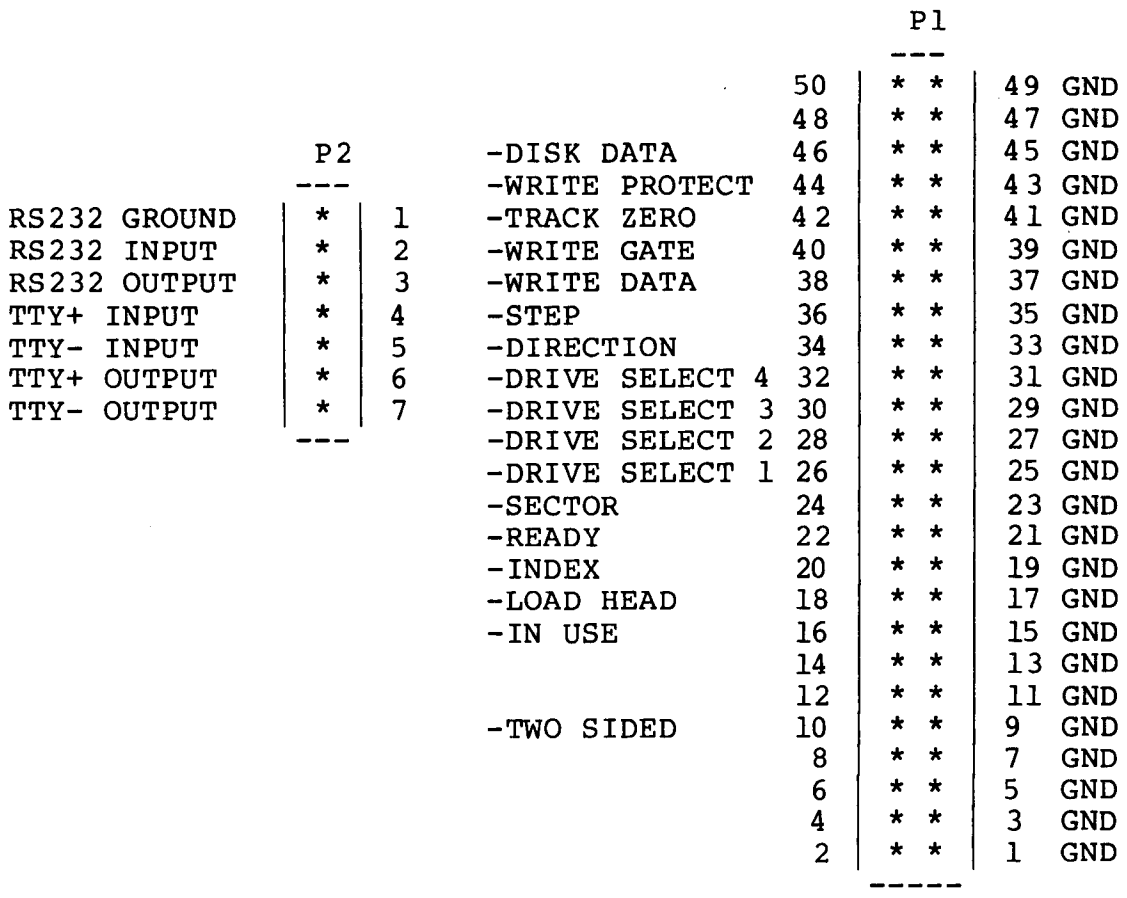
The following program writes from memory starting at 200:000Q (8000H) onto tracks 4,5, and 6 of disk drive 1.

```
0100    31 EE E6        1   WRITE   LXI   SP,200H     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    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 number
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&num
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&num
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 trk
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 CP/M. The supplied CP/M is tailored to the I/O on the Disk Jockey 2D controller. CP/M expects that a serial TTY/RS-232 terminal is connected to P2 (serial port) of the Disk Jockey. CP/M is supplied on a write protected diskette (notch open) which should be kept that way. DO NOT COVER THE NOTCH ON THE DISKETTE. The system is designed to self load when the disk is placed in drive A and a branch is made to 340:000Q (E000H). The CP/M diskette is accompanied by a series of manuals describing how to back-up a CP/M diskette.

Copies of CP/M which are purchased through Morrow Designs are supplied on a diskette which loads 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 when DBOOT is called 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 or before the 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.

Illustrated below are the details of the pin connections of P1 and P2. 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.



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 Morrow Designs. Copies of CP/M sold through Morrow Designs 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 0.

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 347:000Q (0E700H) and is designed to read CP/M into memory from location 51:000Q (2900H) to 77:377Q (3FFFH). After loading CP/M, the LOAD routine branches to location 76:000Q (3E00H) 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 (0E749H) and is the reverse of LOAD. SAVE writes out on the disk starting at track 0 sector 2 all memory locations between 51:000Q (2900H) and 77:377Q (3FFFH). After performing this operation, SAVE comes to a dynamic halt at STALL 347:133Q (0E75BH).

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.

*CP/M is a trademark of Digital Research

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.

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 fourteen one-byte registers-- five of them read only, six 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 1791/8866 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 1791/8866 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 1791/8866, we do not describe these registers in great detail since a data sheet for the 1602 is also included in the documentation.

The 1791/8866 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

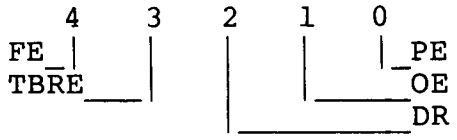
Register 0 - The inverted UART data output register
Location 343:370 (E3F8 hex) standard Disk Jockey:

Data 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

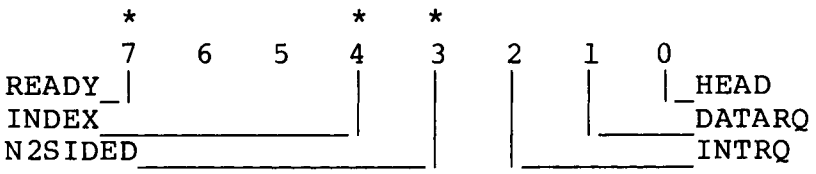


FE = Framing Error
TBRE = Transmitter Buffer Register Empty
DR = Data Ready
OE = Overrun Error
PE = Parity Error

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.

READY - This bit is a 1 when the currently selected drive is powered up with a diskette in place and the door closed.

INDEX - This line reflects the status of the INDEX line from the floppy disk drive. It goes to a 1 once per revolution of the diskette.

N2SIDED- This line is a 0 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.

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 1791/8866 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 0.