PERCOM

LFD-400

SYSTEMS MANUAL

DATE: SEPTEMBER 26, 1977

PERCOM DATA COMPANY
211 N. KIRBY
GARLAND, TEXAS 75042

PERCOM DATA CO. INC.
211 N. Kirby
Garland, Tx 75042
(214) 272-3421

STATEMENT OF LIMITED WARRANTY

For a period of 90 days from the date of delivery, Percom Data Co. Inc. warrants to the original purchaser that the computing equipment described herin shall be free from defects in material and workmanship under normal use and service. During this period, if a defect should occur, the equipment must be returned to the Percom Data Co. Service Facility at the above address for repair. The purchaser must prepay all shipping and insurance charges and must supply proof of purchase from Percom Data Co. or an authorized Percom dealer or distributor. Purchaser's sole and exclusive remedy in the event of defect is expressly limited to the correction of the defect by adjustment, repair or replacement at Percom's election and sole expense, except there shall be no obligation to replace or repair items which by their nature are expendable. No representation or other affirmation of fact, including, but not limited to, statements regarding capacity, suitability for use, or performance of the equipment, shall be or be deemed to be a warranty or representation by Percom Data Co. Inc., for any purpose, nor give rise to any liability or obligation of Percom Data Co. Inc. whatsoever.

EXCEPT AS SPECIFICALLY PROVIDED IN THIS AGREEMENT, THERE ARE NO OTHER WARRANTIES, EXPRESSED OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, ANY IMPLIED WARRANTIES OF MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE AND IN NO EVENT SHALL PERCOM DATA CO. INC. BE LIABLE FOR LOSS OF PROFITS OR BENEFITS, INDIRECT, SPECIAL, CONSEQUENTIAL OR OTHER SIMILAR DAMAGES ARISING OUT OF ANY BREACH OF THIS WARRANTY OR OTHERWISE.

IMPORTANT NOTICE

Although the information contained in this publication has been thoroughly checked for accuracy and reliability, PERCOM DATA CO. INC. shall have no liability or responsibility to a customer or any other person or entity with respect to any liability, loss or damage caused or alleged to be caused directly or indirectly by products or programs sold by PERCOM DATA CO. INC., including but not limited to any interruption of service, loss of business or anticipatory profits or consequential damages resulting from the use or operation of such products or programs. Furthermore, PERCOM DATA CO. INC. does not represent the described equipment or programs as suitable for any purpose and does not assume any liability arising out of the application or use of any product, circuit or program described herein.

PERCOM DATA CO. INC. reserves the right to make changes to any products or specifications described herein without notice.

1.0 SYSTEM DESCRIPTION

The Percom LFD-400/800(EX) is a low cost mini-disk data storage system designed for use with 6800/6809 microcomputers.

The LFD-400/800 was designed to provide a fast, low cost alternative to paper or cassette tape in program development and control applications using the 6800 or 6809 microprocessors. Program development and system software support for the LFD-400/800 includes TEXT EDITORS, TEXT PROCESSORS, various ASSEMBLERS, and an EXTENDED BASIC Interpreter. In addition numerous programs are available from other software suppliers which may be adapted for use with the LFD-400 or LFD-800 systems.

Although the LFD-400/800 disk controller was designed specifically for use with 10-Sector HARD-SECTORED mini-diskettes, it is equally capable of reading SOFT-SECTORED mini-diskettes as well. Consequently, programs which may only be available on SOFT-SECTORED mini-diskettes may be read and used by the LFD-400.

The LFD-400/800 controller circuit card is a SINGLE DENSITY mini-disk controller capable of controlling up to four (4) SINGLE or DUAL-HEADED mini-disk drives. The controller circuit card is available in two (2) forms:

1. The 'EX' version is designed for use with the Motorola EXORcisor (tm) bus or with computing systems using the EXORcisor bus concept such as the Motorola MICROMODULEs and the MEK6800D1, MEK6800D2, and MICRO-CHROMA evaluations 'kits'.

   The 'EX' version disk controller may also be used with 6800 EXORcisor bus based microcomputer modules produced by other manufacturers such as Creative MicroSystems and Percom Data.

2. The 'SS-50' version is designed for use with the 6800 'hobby' computer produced by SouthWest Technical Products in San Antonio, Tex.

The circuit design of both disk controllers is nearly identical. However, the 'EX' version includes a 1k read/write memory (RAM) which is not available on the 'SS-50' version. Both versions include provision for three 2708 EPROMs and are capable of controlling either SINGLE or DUAL-HEADED mini-disk drives.

Both the LFD-400 and LFD-800 use 5-1/4" 10-Sector, HARD-SECTORED mini-diskettes. Each diskette sector includes:

A 10 byte Header
A 256 byte block of data
A 2 byte CRC check code

Thus the LFD-400 which uses 40 track mini-disk drives provides a usable storage capacity of 102,400 bytes/diskette.

(256 Bytes/Sector x 10 Sectors/Track x 40 Tracks)

Since the LFD-400 disk drives include dual INDEX pulse and WRITE PROTECT sensors, the diskettes may be 'flipped over' permitting another 102.4K bytes of data to be stored on the back side of each diskette.

The LFD-800 uses 77 track mini-disk drives manufactured by Micropolis. Despite the decreased spacing between tracks on the Micropolis drives we have found them to every bit as reliable as the 40 track drives. The LFD-800 uses the same controller circuit card as the LFD-400. The usable storage capacity of the LFD-800 is 197,120 bytes/diskette.

(256 Bytes/Sector x 10 Sectors/Track x 77 Tracks)

Unlike the LFD-400, the LFD-800 permits data to be stored on only one side of the diskette.

2.0 DISK OPERATING SYSTEM SUPPORT

Operating system support for the LFD-400/800 is available in four levels:

MINIDOS: Designed for those applications in which the primary function of the disk is to LOAD and SAVE PROGRAMS or DATA as quickly as possible. It permits very direct access to the disk and requires very little additional hardware or software overhead. MINIDOS is supplied on a 2708 EPROM which plugs into the first ROM socket on the LFD-400/800 controller circuit card. MINIDOS may be the only Disk Operating System required in most control applications or feasibility evaluation systems where memory resources are limited. An assembled listing of MINIDOS is contained at the end of this manual.

MPX: A supplement to MINIDOS which permits disk files to be accessed and manipulated by FILE NAME. Its demand on system memory resources is somewhat higher than MINIDOS but is more convenient than MINIDOS for program development. MPX is supplied on a 2708 EPROM which is used with the MINIDOS ROM and plugs into the second ROM socket on the LFD-400/800 controller circuit card. The assembler source of MPX is contained on the SYSTEM DISKETTE supplied with each LFD-400/800 disk system.

DFM: The Disk File Manager (DFM) is a supplement to MPX which permits system and application programs to be easily linked to the MPX operating system. The DFM manages disk file allocation and permits character stream I/O with the disk. The DFM is available on a 2708 EPROM which plugs into the third ROM socket on the LFD-400/800 controller circuit card.

Since most systems programs sold by Percom contain their own DFM, the DFM ROM will be required in very few applications.

INDEX: Is a highly sophisticated, INTERRUPT DRIVEN, I/O DEVICE INDEPENDENT, DYNAMICALLY ALLOCATED Disk Operating System. While programmers accustomed to the Disk Operating Systems on larger computers may find the MPX operating system to be unduly restrictive, they should be well satisfied with the power and convenience of INDEX. INDEX can only be used on systems with more than one disk drive and at least 16k bytes of Read/Write memory (RAM), 8K bytes of which must be dedicated exclusively to the Disk Operating System. Very few control applications or feasibility evaluations will require the power of INDEX, however it is especially convenient for program development and data/text processing applications.

MINIDOS and MPX are described in more detail later in this manual.

3.0 SYSTEM REQUIREMENTS

3.1 DISK CONTROLLER MEMORY MAP

The LFD-400/800 disk controller card occupies a 4k block of the computer's memory space beginning at address $C000 thru address $CFFF ('$' is used to indicate a Hexadecimal value).

ADDRESS RANGE	FUNCTION
CFF8 - CFFF	DISK CONTROLLER I/O (EX VERSION)
CC00 - CFF7	MPX RAM (EX VERSION)
CC00 - CC07	DISK CONTROLLER I/O (SS-50 VERSION)
C800 - CBFF	RESERVED FOR DFM ROM (ROM #3)
C400 - C7FF	RESERVED FOR MPX ROM (ROM #2)
C000 - C3FF	RESERVED FOR MINIDOS ROM (ROM #1)

Additional RAM memory required by the various Disk Operating Systems must be supplied by the user.

A100 - BFFF	RAM FOR 'INDEX' DOS
A080 - A3FF	RAM FOR MPX (SS-50 VERSION)
0020 - 1FFF	RAM FOR MPX AND INDEX TRANSIENT COMMANDS
0000 - 001F	RAM FOR DISK PARAMETERS

The SWTP computer may require some modifications to locate RAM memory at $A000 and $B000. Technical Memo TM-LFD-400-08 contains the necessary modification instructions.

Most of the Motorola MICROMODULES and 'evaluation kits' decode the address of the various ROM, RAM, and I/O components on the module such that they appear at more than one location in the computer memory space. Any address 'collisions' with the memory space used by the LFD-400/800 disk controller must be resolved for proper operation of the disk. The appendices at the back of this manual provide suggestions for resolving the address 'collisions' with various of the more common MICROMODULES and 'evaluation kits'.

3.2 PROCESSOR CLOCK FREQUENCY REQUIREMENTS

For proper operation of the LFD-400/800 mini-disk system the processor clock frequency must be NO LESS THAN 890 kHZ NOR MORE THAN 1.1 mHZ.

3.21 SWTP MP-A2 Processor card

The Resistor/Capacitor network (R1,C1) controlling the clock frequency of the SWTP MP-A2 processor Card should be replaced. Replace R1 with a jumper. Replace C1 with a 4 mHZ crystal (available from Percom). Since some crystals tend to oscllate on 3rd overtone (3 times fundamental frequency), it is wise to connect a 32 pF capacitor in parallel with the crystal to suppress the overtone.

3.22 MEK6800D2 Evaluation Kit

The 614.4 kHZ MC6871B clock module on the MEK6800D2 evaluation kit MUST be replaced with a 1 mHZ version of the same module (available from Percom).

3.23 MEK6800D1 Evaluation Kit

The clock generator one-shots on the MEK6800D1 evaluation kit MUST be carefully adjusted to produce a symmetrical 1 mHz clock.

3.3 HOST SYSTEM MONITOR COMPATIBILITY

Both MINIDOS and MPX operating systems communicate with the operator Data Terminal by calling the I/O subroutines in the Data Terminal ROM Monitor (EXbug, MINIbug, MIKbug, etc.). Since the I/O subroutine call addresses in the various ROM Monitors are different, you MUST make sure the version of the MINIDOS and MPX ROMs on the Disk Controller circuit card match the ROM Monitor with which the disk is used. Versions of MINIDOS and MPX are available for the more popular ROM monitors. Refer to Appendix D. Furthermore, the assembly source code for both ROMs is contained on the SYSTEM DISKETTE for users wishing to alter the code for specific applications or unsupported monitors.

For the same reason it may be necessary to modify the I/O subroutine call vectors in the MPX disk based utility commands.

The SOFTWARE SERVICES division of Percom can supply custom versions of MINIDOS and MPX (as well as other Percom software) for a nominal fee. To request a quotation, submit your specifications and other requirements in writing to:

Percom Data Co. Inc.
Software Services Division
211 N. Kirby
Garland, Tx 75042

4.0 INSTALLATION PROCEDURES

****** WARNING ******

During installation all power should be removed from the computer and connecting peripheral devices to avoid damage to the computer and the LFD-400/800. The LFD-400/800 power cord must be connected to 117 VOLTS 50/60 HZ AC 3-WIRE GROUNDED outlets. --DO NOT DEFEAT the safety prong on the power cord---. In addition to the operator safety provided by the 3-WIRE power connection, the safety ground also shields the low level read electronics in the disc drive from error producing noise pickup.

******* YOU MUST ALSO SAFETY GROUND THE COMPUTER CHASSIS *******

The SWTP 6800 computer is supplied with a 2-wire power cord. this is UNACCEPTABLE and must be replaced with a 3-wire power cord. Connect the 3-wire power cord WHITE wire to solder lug 'A' on terminal strip TS-1. Connect the GREEN wire to solder lug 'B'. Connect the 'BLACK' wire to solder lug 'C'. 3-wire power cords are available from most hardware stores or may be purchased from PERCOM.

******* THE ABOVE PROCEDURE IS VERY IMPORTANT *******

LFD-400 systems are supplied with PERTEC or SIEMENS 40 track mini-disk drives. These disk drives have the capability to 'write' and 'read' on BOTH sides of the diskette. To use this capability, refer to SECTION 5.2 later in this manual.

The LFD-800 uses 77-track Micropolis mini-disk drives which have greater On-Line capacity than the 40-track drives but can only record on one side of a diskette.

4.1 PROGRAMMING MINI-DISK DRIVES

The MINI-DISK drives require no circuit modifications to work in the LFD-400/800 systems. However the drives need to be programmed to permit the drive select signals to selectively enable the appropriate drive.

4.11 Programming the PERTEC drive

Refer to Figure 1 and orient the drive so the component side of the printed circuit board is facing you. Find the DIP switch located near the edge connector. To program the drive for Drive #1, 2, 3, or 4 place the corresponding DIP switch in the 'ON' position. Only one switch should be in the 'ON' position at any time, this is to insure that the drive will only respond to one select line.

As a practical matter when using only one drive, you will want to program the drive as 'drive #1' (switch #1 in 'ON' position).

To install additional disk drives in the system you will need a drive ribbon cable with sufficient connectors to accommodate the extra drives. Furthermore the added drives must be programmed to respond to desired selection. Configure the programming DIP switch as described earlier.

Remove the TERMINATOR PACK (U2) from all drives except the drive which is at the end of the ribbon cable most distant from the controller card. This will normally be Drive #1. The drive at the end of the ribbon cable must have the TERMINATOR PACK (U2) and there must not be a TERMINATOR PACK in any of the other drives.

No two drives can be allowed to respond to the same drive select number, consequently the drive select switch on each drive must be different than any other drive in the system.

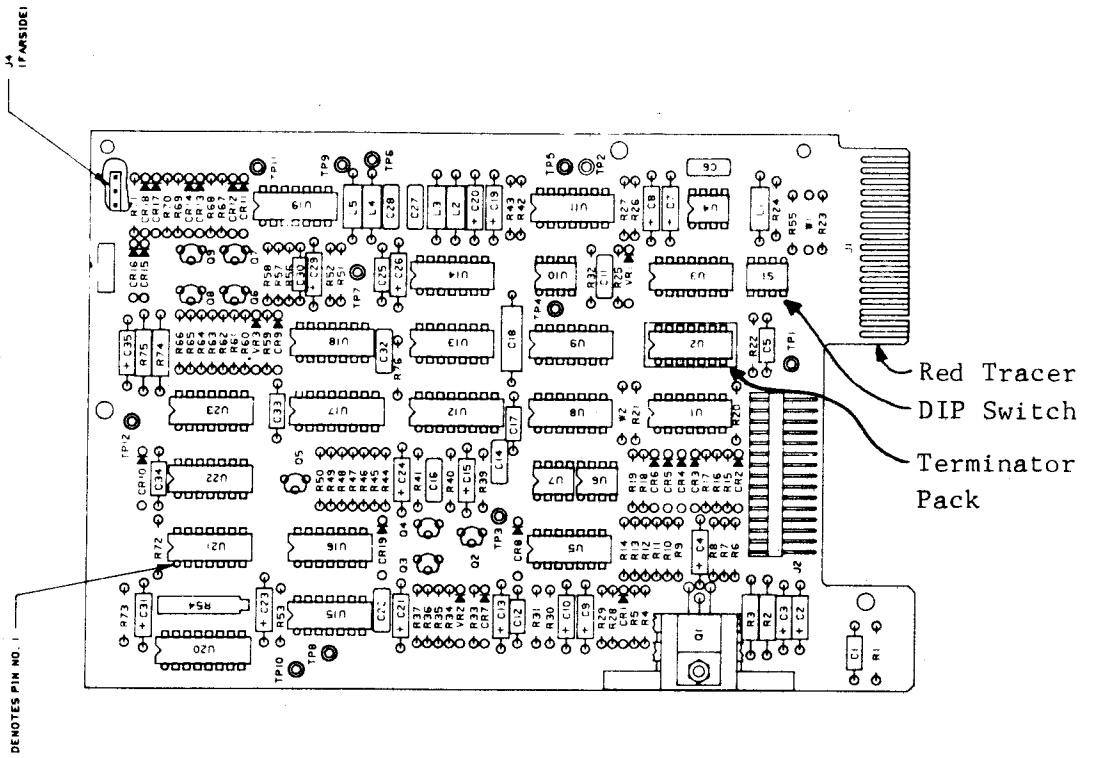


FIGURE 1.

4.12 Programming the SIEMENS drive

Refer to Figure 2 and orient the drive so that the edge connectors are nearest you and so the drive belt and pulleys are on the bottom of the drive. Near the bottom left corner of the disk drive circuit card you will notice what looks like an IC made of jumpers in a socket. Remove this jumper block with a small screwdriver.

* To program the drive as Drive #1, bend outward the jumper block pins which go to terminals DS2, DS3, and MUX. Do not break off the pins.

* To program the drive as Drive #2, bend outward the jumper block pins which go to terminals DS1, DS3, and MUX.

* To program the drive as Drive #3, bend outward the jumper block pins which go to terminals DS1, DS2, and MUX.

* To program the drive as Drive #4 requires the installation of a special jumper on the drive circuit card. Refer to the manual supplied with the drive for instructions.

Re-install the jumper block into its socket. Double check the jumper block connection. Pins should only be inserted into the socket in the 'HS' and desired drive positions. There should be no pins in the 'HM' position.

To install additional disk drives in the system you will need a drive ribbon cable with sufficient connectors to accommodate the extra drives. Furthermore the added drives must be programmed to respond to desired selection.

Configure the programming jumper block as described earlier.

Remove the TERMINATOR PACK (1E) from all drives except the drive which is at the end of the ribbon cable most distant from the controller card. This will normally be Drive #1. The drive at the end of the ribbon cable must have the TERMINATOR PACK (1E) and there must not be a TERMINATOR PACK in any of the other drives.

Jumper block pins should only be inserted into the jumper block socket (1F) in the 'HS' and desired drive positions. There should be no pins in the 'HM' position. No two drives can respond to the same drive select number, consequently the drive select jumper in each drive must be different than any other drive in the system.

4.13 Programming the 77-track Micropolis drive

While referring to Figure 3 locate the 8-pin DIP socket just above the ribbon cable connector. Insert a single jumper wire across the socket to program the drive select. The left-most position is for Drive #1. The TERMINATOR PACK is managed the same as with the PERTEC or SIEMENS drive.

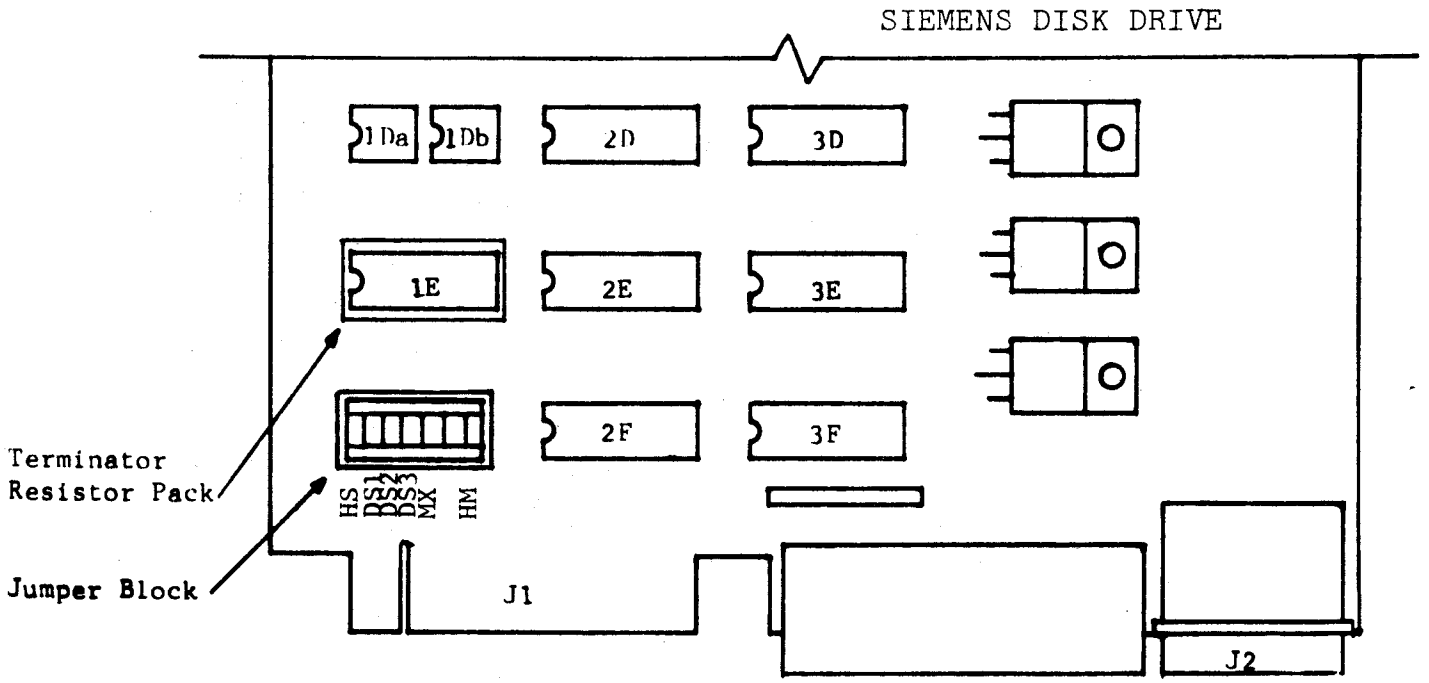


FIGURE 2.

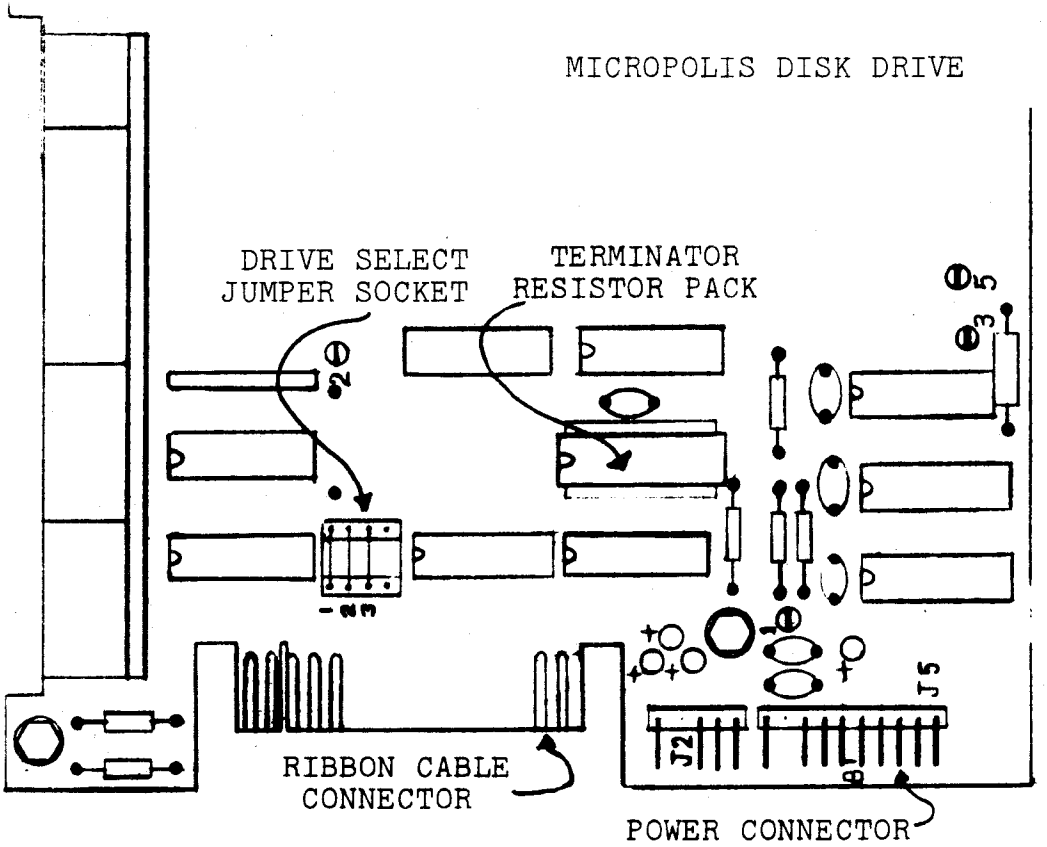


FIGURE 3.

4.2 ASSEMBLY INSTRUCTIONS

1. Plug the 'WHITE' nylon power supply output connector into the mating socket on the rear of the disk drive circuit card. The connector and its mate are 'keyed' and can be put together only one way. Be sure the connector is pushed all the way in.

2. Push the grounding spade on the end of the 'GREEN' wire from the power supply onto the 'FASTON' lug at the rear of the disk drive.

3. Fasten the power supply to the chassis pan with two 6-32 x 1/4" machine screws through the bottom of the chassis pan.

4. Fasten the disk drive to the chassis pan with two 6-32 x 3/8" machine screws through the bottom of the chassis pan. The drive should be mounted with the front panel Drive Select Indicator LED down.

5. Push the connector at one end of the ribbon cable onto the circuit card edge connector at the rear of the disk drive circuit card. Orient the cable so the 'RED' tracer on the edge of the cable is closest to pin #1 of the edge connector. If installed correctly, the ribbon should exit out and away from the drive circuit card.

6. Mount the enclosure cover over the disk drive and power supply making sure the ribbon cable exits from the rear of the enclosure and is not pinched. Secure the cover with four 6-32 x 1/4" machine screws. Do not overtighten.

7. Plug the disk controller circuit card into the computer. Make certain the card is installed correctly.

**** WARNING ****

Severe damage to the computer and interface will occur if the circuit card is not plugged in properly. Double check installation.

8. Push the free end of the ribbon cable connector onto the top of the controller circuit card. The 'RED' tracer on the ribbon must be to the right (as viewed from the front of the circuit card) and the ribbon should exit from the rear of the connector.

5.0 OPERATING PROCEDURES

5.1 DISKETTE CARE AND HANDLING

The diskettes used by both the LFD-400 and LFD-800 are 5-1/4" 10-Sector mini-diskettes. Diskettes must be handled very carefully to insure reliable operation. Creases, bends, scratches, dust and oil contamination will cause data errors and may damage the drive read/write head.

1. When not in use, the diskette should be stored vertically in its protective jacket.

2. Never bend, flex, or snap the diskette. When sending a diskette through the mails pack the diskette in a rigid carton to prevent the diskette from bending. Mark the parcel MAGNETIC SENSITIVE MATERIAL and hope postal employees can read!

3. Never touch the magnetic media.

4. Diskette temperature must not exceed 10-52 degrees (C).

5. Keep the diskette away from transformers, speakers, motors, and other magnetic fields. Keep in mind that many steel objects and appliances carry residual magnetization which may be destructive to the data on a diskette.

6. Never clean the magnetic media. The diskette jacket is designed to perform this function automatically.

7. Do not write on the diskette with a pencil or pen. Use a felt-tip or other soft marker and write only on the label area of the diskette jacket.

8. To protect a diskette from the possibility of accidental erasure or undesired recording, apply a 'gummed tab' over the WRITE PROTECT notch along the edge of the diskette jacket. If the WRITE PROTECT notch is covered the diskette is protected.

9. NEVER-EVER switch power to ANY part of the computing system 'on' or 'off' with a diskette mounted in a drive. The power 'transient' may write 'garbage on the diskette even if the diskette is 'write protected'.

5.2 INSERTING A DISKETTE INTO THE LFD-400 DRIVE

The LFD-400 disk drives permit data to be stored on both sides of the diskette. Although the diskette manufacturers sell diskettes designed for two-sided recording (flippy-disks), nearly all 'single-sided' diskettes work just as well and are less expensive. Since the LFD-400 drives contains dual 'index' and 'write protect' sensors it is not necessary to modify or punch holes in the diskette. If you have punched additional 'index' and 'write protect' holes in the jacket of your diskettes, these holes must be covered for the diskette to function properly in the LFD-400.

Diskettes are inserted into the drive with the 'long oval' cutout entering the drive first. To read and write on SIDE A (normal or front side) insert the diskette into the drive with the label AWAY from the Drive Select Indicator LED. To read and write on SIDE B (back side) insert the diskette into the drive with the diskette label on the same side of the diskette as the Drive Select Indicator LED.

Some users have reported difficulities with diskettes binding when diskettes are inserted in new drives. When inserting a diskette into a drive make sure the diskette is 'free' before closing the drive door. When closing the drive door, press the door latch until it 'bottoms' but not hard enough to latch closed. Release the pressure on the door slightly then press the door latch again to lock the door closed. This permits the centering hole in the diskette to 'walk' up the diskette centering hub without binding. The problem diminishes with time as the diskette centering hub is 'polished' by repeated use.

5.3 INSERTING A DISKETTE INTO THE LFD-800 DRIVE

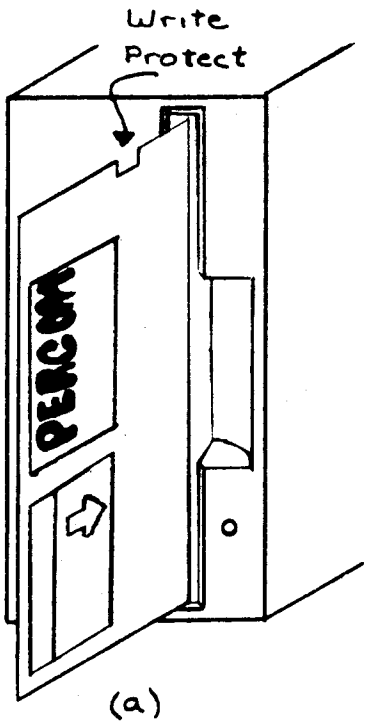
NEVER insert a diskette into the LFD-800 drive unless disk drive power is ON.

A diskette is inserted into the drive with the LOAD ACTUATOR in the OPEN position (see Figure 4a). Push the diskette into the drive until an audible 'click' is heard. This means the diskette is properly located. To load the diskette, firmly and slowly 'squeeze' the LOAD ACTUATOR as far as it will go. See Figure 4b. The LOAD ACTUATOR should lock in the LOADED position.

If the diskette is missing or is not properly inserted into the drive, it is not possible to depress the LOAD ACTUATOR. This protects the diskette from damage if not inserted properly.

5.4 REMOVING A DISKETTE FROM THE LFD-800 DRIVE

To remove a diskette from the LFD-800, 'squeeze' the LOAD ACTUATOR the ~~save~~ same as when loading a diskette then allow it to spring to the OPEN position. To eject the diskette, place the tip of a forefinger or thumb under the LOAD ACTUATOR and push outward. This will unlatch the diskette interlock and eject the diskette into your hand.

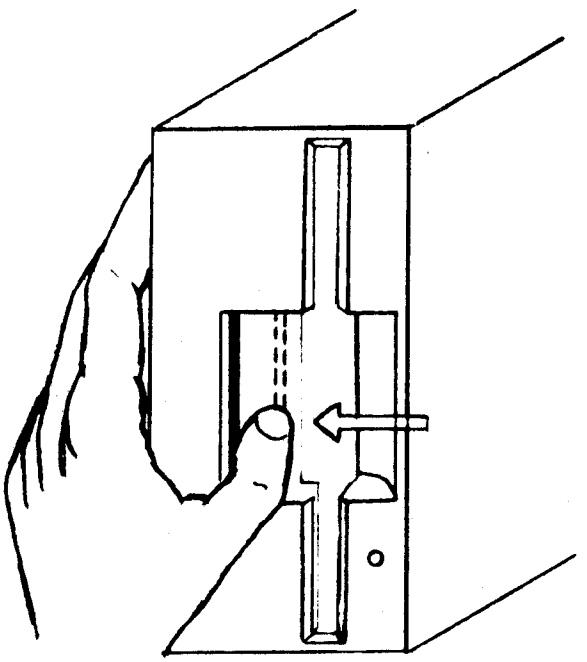


(a)

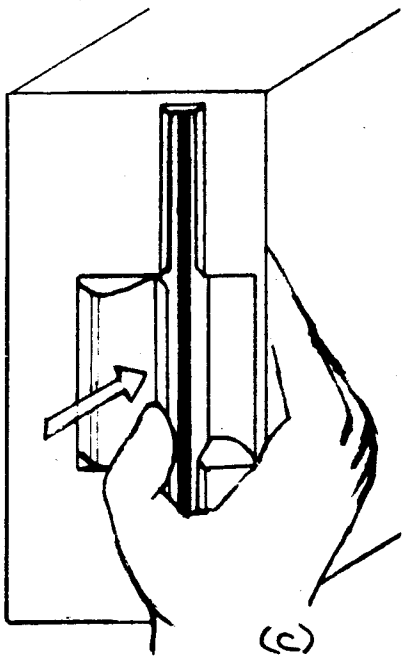
Insert the diskette into the drive with the Write Protect notch up. Push the diskette in until an audible 'click' is heard.

Load the diskette by 'squeezing' the LOAD ACTUATOR until it locks.

Unload the diskette the same way.



(b)



(c)

Eject the diskette by pushing the LOAD ACTUATOR outward.

FIGURE 4.

6.0 INITIAL SYSTEM CHECKOUT

*** WARNING ***

NEVER switch power to the DISK DRIVE or ANY part of the computing system 'on' or 'off' with a diskette mounted in a drive. The power 'transient' may write 'garbage' on the diskette even if the diskette is 'write protected'. It is sufficient to open the disk door and withdraw the diskette an inch or so when switching power to the system 'on' or 'off'.

The LFD-400/800 does not require any specially placed RAM memory when using the MINIDOS(tm) operating system. Specially placed RAM memory IS required when using one of the 'named file' disk operating systems such as MPX or INDEX. MINIDOS(tm) uses memory locations $0000 thru $001F for temporary and sector header storage which by Motorola convention has been reserved for floppy disk operation.

1. Remove any diskettes and power up the drive and computer.
2. Disk drive motor(s) may turn on. The drive activity LED indicator must be 'off'. The drive motor (if on) will turn off in 5-10 seconds. If the motor runs continuously and the activity LED stays on, the ribbon cable may be installed incorrectly.
3. Examine computer memory location $C000: it must be '7E'.
4. If you are using the SS-50 bus system store a '40' in memory location $CC03 (memory location $CFFB if you are using the EX controller). The computer will respond with a question mark (?) and return to the monitor. This selects drive #1 but the drive activity LED will not turn on until the next step.
5. Examine memory location $CC05 (~~$CFFC~~ $CFFD on the EX controller). The drive motor and drive #1 activity LED should turn 'on' and remain on for several seconds. This demonstrates the disk controller is responding (at least partially) to command.
6. Read the sections of this manual describing the procedure for loading a program using MINIDOS or MPX.
7. Load and run the memory test program contained on the SYSTEM DISKETTE . If you have difficulty loading the program, re-insert the diskette into the drive and try again. If you continue to have difficulty refer to the section entitled IN CASE OF DIFFICULTY.
8. Since most problems attributed to the disk have been traced to defective memory, run the memory test over your entire memory. Perform the test over 4K of memory at a time. Incidently, the ROBIT and MEMCON memory tests supplied by SWTP are virtually worthless!

7.0 MINIDOS (ROM #1)

The MINIDOS ROM occupies the first ROM socket on the disk controller card (address $C000 - $C3FF). The ROM contains the software drivers and other useful subroutines which directly control the disk. A Jump table at the beginning of the ROM permits the various driver subroutines to be called by higher level programs and Operating Systems.

The MINIDOS ROM also contains a PRIMITIVE Disk Operating System which permits blocks of memory or programs to be saved to or loaded from disk in applications which do not require a more sophisticated Disk Operating System. Consequently the MINIDOS ROM will be the only DOS required in many control applications and feasibility evaluations where the purpose of the disk is to SAVE and LOAD essential programs or STORE and RETRIEVE collected data.

The primitive DOS within MINIDOS communicates with the operator Data Terminal by calling subroutines in the Data Terminal ROM Monitor (EXBUG, MIKBUG, TVBUG, etc.). Since the I/O subroutine call addresses in the various ROM monitors are different, you must make sure the version of the MINIDOS ROM installed on the disk controller circuit card matches the ROM Monitor with which it will be used. Versions of the MINIDOS ROM are available for use with the more popular ROM Monitors. Furthermore, the assembly source of MINIDOS is contained on the LFD-400/800 SYSTEM DISKETTE for those users wishing to alter the code for specific applications or unsupported Monitors. A printed listing of the MIKBUG version of MINIDOS is at the back of this manual.

7.1 MINIDOS OPERATING PROCEDURES

7.11 SAVING A PROGRAM

To illustrate the following procedure we will assume you wish to save a program so that it may be loaded back into the computer at a later time. The program could be any program (or data) you wish to save. We will use SWTP 8K BASIC to illustrate the procedure.

You must decide where on the disk you wish to begin saving the program. The diskette contains 400 (or 770) blocks or sectors which, for the purpose of this discussion, are numbered from 000 to 399 (or 769). Avoid using sectors 000-009. These sectors are used by the PERCOM MPX operating system for directory storage. Even if you are not using MPX to save and load data, the format in which the data is stored on diskette by MINIDOS is upward compatible. Since each block (sector) will hold 256 bytes of data, an 8K program will require 32 sectors. To keep track of what is stored on a disk and where it is stored, write the name of the program and the first and last block used on the jacket used to store the diskette.

First load the program you wish to save by whatever means you have been using before (keyboard, paper tape, cassette, etc.).

USING SWTP 8K BASIC AS AN EXAMPLE:

The first byte of the 8K BASIC program is at address $0100. We will call this the `BEGINNING ADDRESS` (BEGA).

The last byte of an 8K program is at address $1FFF. We will call this the `ENDING ADDRESS` (ENDA).

To commence program execution you would cause the computer program counter to go to address $0100. We will call this the program `TRANSFER ADDRESS` (XFER).

*** NOTE *** the `TRANSFER ADDRESS` does not have to be the same as the `BEGINNING ADDRESS`. Many programs begin execution at some point other than the beginning address.

Assume you wish to save the program on drive 1 beginning at block (sector) 25, Jump or Go to MINIDOS at address $C000.

Now type the following information:

```
S 0100 1FFF 0100 1025
- ---- ---- ---- ----
```

DRIVE 1, BLOCK 25
TRANSFER ADDRESS (PROGRAM START)*
ENDING ADDRESS
BEGINNING ADDRESS
SAVE COMMAND

*IF YOU DO NOT WISH TO SAVE THE TRANSFER ADDRESS ON DISKETTE, TYPE `FFFF` INSTEAD.

In a couple of seconds the computer will report:

```
LAST SECTOR=XXX
```

and return to the system monitor.

Write the last sector number on the storage jacket of the diskette together with the program name and starting sector for future reference.

FOR EXAMPLE:

```
FILE   BEGA ENDA XFER FRST LAST

BASIC  0100 1FFF 0100 X010 X041
```

If something goes wrong (DISK MISSING, INVALID BLOCK NUMBER, DISK PROTECTED) MINIDOS(tm) will report:

```
***ERROR X
```

Refer to APPENDIX A for ERROR codes.

7.0 MINIDOS (ROM #1)

The MINIDOS ROM occupies the first ROM socket on the disk controller card (address $C000 - $C3FF). The ROM contains the software drivers and other useful subroutines which directly control the disk. A Jump table at the beginning of the ROM permits the various driver subroutines to be called by higher level programs and Operating Systems.

The MINIDOS ROM also contains a PRIMITIVE Disk Operating System which permits blocks of memory or programs to be saved to or loaded from disk in applications which do not require a more sophisticated Disk Operating System. Consequently the MINIDOS ROM will be the only DOS required in many control applications and feasibility evaluations where the purpose of the disk is to SAVE and LOAD essential programs or STORE and RETRIEVE collected data.

The primitive DOS within MINIDOS communicates with the operator Data Terminal by calling subroutines in the Data Terminal ROM Monitor (EXBUG, MIKBUG, TVBUG, etc.). Since the I/O subroutine call addresses in the various ROM monitors are different, you must make sure the version of the MINIDOS ROM installed on the disk controller circuit card matches the ROM Monitor with which it will be used. Versions of the MINIDOS ROM are available for use with the more popular ROM Monitors. Furthermore, the assembly source of MINIDOS is contained on the LFD-400/800 SYSTEM DISKETTE for those users wishing to alter the code for specific applications or unsupported Monitors. A printed listing of the MIKBUG version of MINIDOS is at the back of this manual.

7.1 MINIDOS OPERATING PROCEDURES

7.11 SAVING A PROGRAM

To illustrate the following procedure we will assume you wish to save a program so that it may be loaded back into the computer at a later time. The program could be any program (or data) you wish to save. We will use SWTP 8K BASIC to illustrate the procedure.

You must decide where on the disk you wish to begin saving the program. The diskette contains 400 (or 770) blocks or sectors which, for the purpose of this discussion, are numbered from 000 to 399 (or 769). Avoid using sectors 000-009. These sectors are used by the PERCOM MPX operating system for directory storage. Even if you are not using MPX to save and load data, the format in which the data is stored on diskette by MINIDOS is upward compatible. Since each block (sector) will hold 256 bytes of data, an 8K program will require 32 sectors. To keep track of what is stored on a disk and where it is stored, write the name of the program and the first and last block used on the jacket used to store the diskette.

First load the program you wish to save by whatever means you have been using before (keyboard, paper tape, cassette, etc.).

*** NOTE ***

DO NOT TRY TO SAVE MEMORY FROM ADDRESS $0000 THRU $001F. THIS SPACE IS USED BY THE DISK DRIVER SOFTWARE TO KEEP TRACK OF DRIVE OPERATION AND IS NOT AVAILABLE FOR PROGRAM USE.

7.12 LOADING A PROGRAM FROM THE DISK INTO THE COMPUTER

Data on the disk may be loaded into the computer memory to the same address from which it was originally saved (PRIMARY ADDRESS) or it may be loaded to an `ALTERNATE ADDRESS`. If the information on the disk is an executable program if will normally be loaded into the same memory location from which it was originally saved. This may not be true if the disk file is a relocatable program or if it is a data file to be processed by another program.

ASSUME WE WISH TO LOAD THE 8K BASIC PROGRAM SAVED EARLIER:

Jump or Go to the ENTRY ADDRESS of the MINIDOS ROM ($C000).

Now type the following information:

```
L 1025 FFFF
- ---- ----
 \    \    \_____ TARGET ADDRESS - READ THE FOLLOWING
  \    \          DISCUSSION FOR MORE INFORMATION
   \    \_______ DRIVE 1, BLOCK 25
    \____________ LOAD COMMAND
```

In a couple of seconds the computer will respond with the system monitor prompt command (* or $) indicating the program is loaded and ready.

If something goes wrong (DISK MISSING, INVALID BLOCK NUMBER, READ ERROR, etc.) MINIDOS(tm) will report:

```
***ERROR X
```

Refer to Appendix A for error codes.

In the above example we entered `FFFF` as the TARGET ADDRESS to cause the program to be loaded back into the same memory location from which it was saved. In the header information preceding the data in each block stored on the diskette is recorded the address of the first byte of data in the block. We call this the `PRIMARY TARGET` address.

If you wish to load the data into memory beginning at some address other than the primary target address (for example a relocatable program), enter the `ALTERNATE TARGET` address instead of `FFFF`.

*** SPECIAL NOTE ***

DO NOT TRY TO LOAD DATA INTO MEMORY LOCATIONS $0000 THRU $001F. THIS SPACE IS USED BY THE DISK DRIVER SOFTWARE TO KEEP TRACK OF THE DRIVE OPERATION. LOADING A PROGRAM INTO THIS SPACE WILL DESTROY ESSENTIAL DISK DRIVER INFORMATION. DO NOT TRY TO STORE DATA IN THE `STACK`. THIS WILL `CRASH` THE STACK AND PRODUCE UNPREDICTABLE RESULTS.

7.13 TRACK SELECTION

In the above procedures nothing was said about selecting the disk track. The disk SAVE and LOAD routines convert the decimal block (sector) number supplied by the user into physical track and sector information internally. Since there are 10 sectors per track, the two most significant digits of the block number identify the track.

7.14 DRIVE SELECTION

The desired drive number is the first digit of the four digit number you entered to SAVE or LOAD a file. In a single drive system there is only one drive; drive 1. In a multiple drive system the drives number from 1 to 3. Selecting drive 0 is a drive deselect and will result in an error message.

7.2 MINIDOS DRIVERS AND SUBROUTINES

The various disk driver subroutines contained in the MINIDOS(tm) PROM may be readily linked to existing software. In addition to the self standing SAVE and LOAD routines used to save and load programs without the need for a more complicated DOS, the PROM contains a number of disk driver and driver support subroutines.

MINIDOS(tm) contains a jump table at the beginning of the PROM. This jump table links to the various subroutines in the PROM and may be expanded in future revisions of MINIDOS.

For purposes of linking to existing software there are only three entry points of major interest:

INITIALIZE DISKS	($C027)
READ A SECTOR	($C00C)
WRITE A SECTOR	($C00F)

These three routines take care of selecting drives, starting motors, seeking tracks, etc. Obviously to read or write a sector of data, you must supply several parameters. Refer to the description of these three subroutines for additional information.

There are other routines which may be useful in some programs but are not absolutely necessary for proper disk operation. Some of these routines are used by the sector read and write subroutines, others provide the console interaction with MINIDOS(tm). There may be other useful routines within the MINIDOS(tm) PROM which are not reached through the jump table but which may be useful to you. Be aware that the entry point to these cannot be guaranteed in future versions of the MINIDOS(tm) PROM. You must study the listing for the operation and use of these subroutines.

7.21 ERROR REPORTING:

In subroutines in which error conditions occur, the CARRY status bit will be set upon return from the subroutine. Accumulator A will contain a code identifying the error. If the CARRY bit is clear, no error condition exists.

7.22 JUMP TABLE ENTRY POINTS:

MINIDOS: ($C000): THIS IS THE ENTRY POINT FOR THE MINIDOS(TM) COMMAND PROCESSOR. THIS IS NOT A SUBROUTINE BUT A SELF STANDING PROGRAM WHICH RETURNS TO THE SYSTEM MONITOR. IT USES VARIOUS SUBROUTINES IN BOTH THE MONITOR AND MINIDOS ROMS.

THE COMMAND PROCESSOR HAS AN AUTOMATIC EXPANSION FEATURE. UPON ENTRY INTO MINIDOS(TM), THE CONTENT OF MEMORY LOCATION $C400 IS EXAMINED. IF A $7E IS IN THIS LOCATION, THE PROGRAM JUMPS TO ADDRESS $C400 OTHERWISE IT EXECUTES THE INSTRUCTIONS IN MINIDOS(TM). $C400 IS THE FIRST LOCATION IN THE SECOND ROM ON THE DISK CONTROLLER CARD. IF A ROM IS NOT PLUGGED INTO THE SOCKET PROVIDED, THE PROGRAM WILL READ $FF AT ADDRESS $C400 AND WILL NOT JUMP TO THE SECOND ROM. IF YOU WISH TO PUT A ROM IN THE SECOND ROM SOCKET BUT DO NOT WANT MINIDOS(TM) TO JUMP TO THE ROM, THE FIRST BYTE IN THE SECOND ROM MUST NOT BE $7E. INCIDENTLY, THE PERCOM MPX PROM IS DESIGNED TO PLUG INTO THE SECOND ROM SOCKET ON THE DISK CONTROLLER CARD. MPX IS A SIMPLE NAMED FILE DISK OPERATING SYSTEM WHICH SUPPORTS UP TO 31 NAMED FILES. YOU MAY WISH TO DESIGN YOUR OWN DOS OR SYSTEM UTILITIES INTO 2708 EPROMS TO BE PLUGGED INTO THE SECOND OR THIRD PROM SOCKETS.

ENTER:	SELF CONTAINED
EXIT:	DIRECT JUMP TO MONITOR CONTROL
ERROR:	PROCESSED AND PRINTED ON SYSTEM CONSOLE
FUNCTION:	PROGRAM/DATA SAVE/LOAD FROM/TO MAIN MEMORY. PROVISION FOR AUTOMATIC FUNCTION EXPANSION.

CVTDTS: ($C003) USED BY MINIDOS TO CONVERT THE OPERATOR SUPPLIED DRIVE AND SECTOR NUMBER (DSSS) TO THE PHYSICAL ALTERNATE SECTOR REQUIRED BY MINIDOS(TM). THIS IS VERY USEFUL WHEN LINKING MINIDOS(TM) TO EXISTING PROGRAMS.

ENTER:	4-DIGIT BCD NUMBER IN 'X' REGISTER		
EXIT:	DRIVE*	$0000	(MS 2 BITS)
	TRACK	0001	
	SECTOR	0002	
ERROR:	INVALID NUMBERS, CARRY SET, CODE IN A		
FUNCTION:	CONVERT LOGICAL DSSS TO PHYSICAL DRV, TRK, SEC		

RESTORE: ($C006) SENDS PREVIOUSLY SELECTED DRIVE TO TRACK 00

EXIT: CURRENT TRACK REGISTER CLEAR ($000F)

SEEK: ($C009) SEEK DESIRED TRACK

ENTER:	DESIRED TRACK	0001
	CURRENT TRACK	000F
EXIT:	CURRENT TRACK UPDATED	

RDSEC: ($C00C) READ A SECTOR

ENTER:	DESIRED DRIVE*	$0000	(MS 2 BITS)
	DESIRED TRACK	0001	
	DESIRED SECTOR	0002	
	MEMORY TARGET	0016-0017	
	IF MEMORY TARGET IS $FFFF, ROUTINE WILL PLACE DATA INTO ADDRESS READ FROM THE DISK HEADER.		
EXIT:	SECTOR HEADER AND CRC CODE IN LOCATIONS $0000-$000E		
ERROR:	CARRY SET, ERROR CODE IN A		

WTSEC: ($C00F) WRITE A SECTOR

ENTER:	DESIRED DRIVE*	$0000	(MS 2 BITS)
	DESIRED TRACK	0001	
	DESIRED SECTOR	0002	
	BACK LINK TRACK	0003	
	BACK LINK SECTOR	0004	
	FORWARD LINK TRACK	0005	
	FORWARD LINK SECTOR	0006	
	BYTE COUNT	0007	
	TARGET ADDRESS(HI BYTE)	0008	
	TARGET ADDRESS (LO BYTE)	0009	
	FILE TYPE	000A	
	DATA SOURCE (HI BYTE)	0014	(TA)
	DATA SOURCE (LO BYTE)	0015	
ERROR:	CARRY SET, ERROR CODE IN A		

SLTDRV: ($C012) SELECT DESIRED DRIVE, START MOTOR, CHECK IF DISK IS MISSING, SEEK TRACK, CHECK DRIVE PARAMETERS

ENTER:	DESIRED DRIVE*	$0000	(MS 2 BITS)
	DESIRED TRACK	0001	
EXIT:	CARRY SET, ERROR CODE IN A		

SRTMTR: ($C015) TURN ON DRIVE MOTORS, DELAY FOR STARTUP

SAVE: ($C018) SAVE A MEMORY IMAGE FILE

ENTER:	DESIRED DRIVE*	$0000	(MS 2 BITS)
	DESIRED TRACK	0001	
	DESIRED SECTOR	0002	
	FILE TYPE	000A	
	DATA SOURCE (HI BYTE)	0014	
	DATA SOURCE (LO BYTE)	0015	
	ENDING ADDRESS (HI BYTE)	001E	
	ENDING ADDRESS (LO BYTE)	001F	
ERROR:	CARRY SET, ERROR CODE IN A		

LOAD: ($C01B) LOAD A MEMORY IMAGE FILE

ENTER	DESIRED DRIVE*	$0000	(MS 2 BITS)
	DESIRED TRACK	0001	
	DESIRED SECTOR	0002	
	MEMORY TARGET	0016-0017	
EXIT	CARRY SET, ERROR CODE IN A		

TYPERR: ($C01E) TYPE ERROR MESSAGES TO CONSOLE

ENTER	ERROR CODE IN A	

FWDCAL: ($C021) CALCULATE FORWARD SECTOR LINK

ENTER	CURRENT TRACK	0005
	CURRENT SECTOR	0006
EXIT	FORWARD LINK TRACK	0005
	FORWARD LINK SECTOR	0006

LENGTH: ($C024) CALCULATE SECTOR BYTE COUNT LENGTH. THIS ROUTINE IS NOT NORMALLY USED BY PROGRAMS OTHER THAN MINIDOS. STUDY THE LISTING IF YOU HAVE USE FOR THIS ROUTINE.

INITRK: ($C027) INITIALIZE DRIVE TRACK REGISTERS. THIS IS THE EQUIVALENT OF INITIALIZING THE DRIVES. THIS ROUTINE STORES $FF IN THE CURRENT TRACK REGISTER AND IN EACH OF THE TRACK HISTORY REGISTERS INDICATING THE DRIVES ARE OFF-LINE AND NEED TO BE INITIALIZED BEFORE USE. THE RDSEC AND WTSEC ROUTINES NOTE THIS CONDITION AND PERFORM DRIVE INITIALIZATION IF NECESSARY.

PRTSEC: ($C31F) TYPE LAST SECTOR USED ON SYSTEM CONSOLE. USEFUL REPORTING FUNCTION WHEN LINKING MINIDOS TO EXISTING PROGRAMS.

ENTER:		
	LAST TRACK USED	0001
	LAST SECTOR USED	0002

PCRLF: ($C363) TYPE CARRIAGE RETURN, LINE FEED.

* DESIRED DRIVE IS THE MOST SIGNIFICANT 2 BITS IN MEMORY LOCATION $0000. THE REMAINING 6 BITS ARE CURRENTLY IGNORED. FOR EXAMPLE, DRIVE 1 IS 01XXXXXX, DRIVE 2 IS 10XXXXXX, DRIVE 3 IS 11XXXXXX.

7.23 FILE TYPES:

FILE TYPE ASSIGNMENTS HAVE BEEN SOMEWHAT ARBITRARY, HOWEVER THE FOLLOWING TYPE CODES ARE IN CURRENT USE.

- 00 - A FILE CREATED BY MINIDOS
- 0B - BASIC PROGRAM FILE (MEMORY IMAGE OF SWTP 2.0 OR 2.2)
- 0C - TSC ASSEMBLER OBJECT CODE FILE
- 0E - TSC TEXT EDITOR FILE (MEMORY IMAGE WITH BCD LINE NUMBERING)
- BA - BASIC PROGRAM FILE
- BD - BASIC DATA FILE
- D0 - MINIDOS-PLUS DIRECTORY FILE
- E1 - EDIT68 (HEMENWAY) TEXT FILE
- ED - TOUCHUP EDITOR TEXT FILE

IN THE FUTURE THE FOLLOWING GENERAL TYPE ASSIGNMENTS WILL APPLY:

- 0X - FILES CREATED BY MINIDOS AND MEMORY IMAGE FILES
- 10 - 9F (UNASSIGNED)
- AX - ASSEMBLER CREATED FILES (OBJECT, LISTING, ETC.)
- BX - BASIC CREATED FILES (PROGRAM, DATA)
- CX - COMPILER CREATED FILES
- DX - DIRECTORY FILES
- EX - TEXT EDITOR CREATED FILES
- FX - (UNASSIGNED)

8.0 MPX OPERATING SYSTEM (ROM #2)

The MPX ROM (sometimes referred to as MINIDOS-PLUSX) occupies the second ROM socket on the LFD-400/800 disk controller card (address $C400 - $C7FF). MPX is a simple ROM based Disk Operating System which permits disk files to be access and manipulated by 6 character file names.

MPX should be adequate for all applications except those requiring an extremely sophisticated operating system.

A Jump table at the beginning of the MPX ROM permits some of the MPX subroutines to be called and used by other programs.

The MPX operating systems has two types of commands:

ROM RESIDENT
DISK RESIDENT (Transient)

The ROM resident commands are:

S(ave) <NAME> <BEGA> <ENDA> [<EXEC>]
L(oad) <NAME>
D(elete) <NAME>
F(iles)
I(nitialize)
R(ename) <THIS> <THAT>
A(llocate)
J(ump) <ADDRESS>
X(exit)
M(inidos)

Each of these commands will be described in more detail later.

ROM resident commands consist of single letters. If more than one character is entered, the MPX directory is searched for a file with the name of the given command. If such a file is found, it will be loaded into memory and executed just as if it were a command. Such DISK RESIDENT commands are called 'TRANSIENT' because they are called into memory only when needed. The SYSTEM DISKETTE is supplied with a number of TRANSIENT commands which will be described in detail later.

The Disk Resident Transient command feature of MPX permits the operating system to be expanded indefinitely or customized to specific system requirements.

Like MINIDOS, the MPX operating system communicates with the operator Data Terminal by calling I/O subroutines in the Data Terminal ROM Monitor (MIKBUG, EXBUG, MICROBUG, etc.). Since the I/O subroutine call addresses in the various ROM Monitors are different, you must make sure the version of MPX installed on the LFD-400/800 controller circuit card matches the ROM Monitor with which it will be used. Versions of the MPX ROM are available for use with the more popular ROM monitors. The assembly source of MPX is contained on the SYSTEM DISKETTE for those users wishing to alter the code for specific applications or unsupported Monitors.

8.01 MPX Procedures

Enter MPX using the computer ROM monitor by J(umping) or G(oing) to address $C000. If the MPX ROM is installed in the Disk controller card the automatic 'look-ahead' feature of MINIDOS will transfer control to MPX. MPX outputs a '>' whenever is is waiting for a command:

```
>
```

MPX commands are entered into a 32 character buffer and no action is taken until the receipt of a RETURN code. This permits errors to be edited during entry.

```
Control-H will BACKSPACE in the command line
Control-X will CANCEL the line and reprompt (>)
RETURN terminates the command input
```

SPACES (blanks) are delimiters between entries in the command where more than one parameter is required.

Drives other than DRIVE #1 may be selected by preceding the command with the drive number and a SLASH (/). If no drive number is specified DRIVE#1 is assumed.

```
2/L NAME
1/L NAME
L NAME
```

To PROTECT a file, make the first character in the file name '@'. To UNPROTECT a protected file rename the file without the '@'.

```
@NAME                   Protected file
R @NAME NAME            File is now unprotected
R NAME @NAME            File protected again
```

8.1 MPX ROM RESIDENT COMMANDS

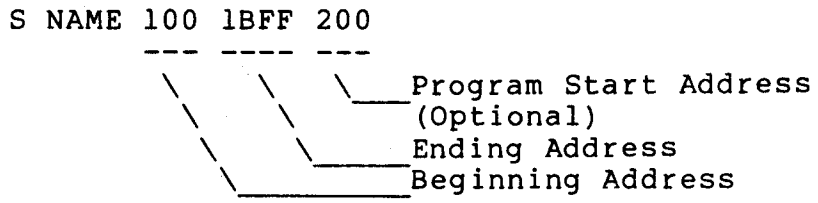
8.11 Initializing the Disk ('I' command)

The disk directory must be initialized before the disk can be used by MPX. The INITIALIZE command destroys any information previously stored on the first two (2) sectors of Track #0. The remaining eight (8) sectors of Track #0 are reserved for directory expansion.

```
I
2/I
```

8.12 Saving a Program or File ('S' command)

The SAVE command is used to create a new file or to 'overwrite' an existing file by the same name. The file name must be 6 characters or less. The first character in the file name must be a letter (A - Z) or '@'. '@' is used to protect a file. If an old file is being 'overwritten', system protection features prevent destruction of adjacent files (Allocation Error).



Leading Zeros in the addresses are not required.

8.13 Allocating Extra Space ('A' command)

If you expect a new file to eventually require more space than it uses when first saved, you can reserve 10 extral blocks (sectors) with the ALLOCATE command. This command may be invoked repeatedly to reserve multiples of 10 blocks. Remember 256 bytes or characters may be stored in each block.

8.14 Loading a File ('L' command)

The LOAD command causes a program to be loaded into memory but does not begin execution of the program. This is useful if you wish to make changes in a program before it is executed. To 'LOAD' and 'GO' simply enter the program name without the 'L' prefix. The file will be located, loaded, and executed if a program Start address was supplied when the program was saved to diskette.

L BASIC	This caused the program named 'BASIC' to be loaded but not executed. After loading, system control returns to MPX.
BASIC	This causes the program named 'BASIC' to be located, loaded, and executed.

Obviously a separate 'RUN' command is not required.

The MPX LOAD (and GO) command may also be used to load programs from diskettes which do not contain a directory (MINIDOS files). Enter the 4 digit drive and sector number (DSSS) of the desired file in place of a file name.

L 1025	Loads the program on drive #1 beginning at sector 25.
1025	Loads and executes the program on drive #1 at sector 25

8.14 Listing the Directory ('F' command)

The FILES command causes the disk directory to be listed on the Data Terminal. The following information is supplied for each file:

FILE NAME	(FILE)
FIRST BLOCK USED	(FST)
LAST BLOCK USED	(LST)
BEGINNING ADDRESS	(BEGA)
ENDING ADDRESS	(ENDA)
PROGRAM START ADDRESS	(STRT)

Addresses are given in Hexadecimal values. If the program Start address is FFFF, a program start address was not supplied when the file was saved and automatic program execution is inhibited.

```
F
2/F
```

8.15 Renaming a File ('R' command)

The RENAME command allows a file name to be changed. It also permits a 'protected' file to be 'unprotected' and vice-versa.

```
R THIS THAT
2/R THIS THAT
```

8.16 Deleting a file ('D' command)

This command should be used sparingly because the space occupied by a deleted file cannot be recovered unless the diskette is REPACKED (a relatively hazardous function). An alternative procedure is to rename obsolete files with distinctive names such as DUMMY1 or DEAD2. Later the dead space may be reclaimed by renaming and overwriting. System protections prevent allocation errors from destroying adjacent files.

```
D NAME
2/D NAME
```

8.17 Jump to Address ('J' command)

This command permits the user to Jump to a specified address to begin program execution. It simply avoids the necessity of returning to the System Monitor to do the same thing.

```
J 100
```

8.18 Exit to System Monitor ('X' command)

This command return control to the System Monitor.

```
X
```

8.19 Exit to MINIDOS ('M' command)

The automatic 'Look-Ahead' feature of MINIDOS causes a jump into the MPX ROM (if it is installed) thereby bypassing the primitive SAVE and LOAD functions of MINIDOS. This command permits you to utilize these primitive functions if necessary.

8.2 MPX ERROR CODES

Error codes 0 thru 5 are the same as MINIDOS

```
6  FILE NAME NOT FOUND
7  DIRECTORY FULL
8  COMMAND NOT UNDERSTOOD
9  ALLOCATION ERROR
A  BAD ADDRESS
```

8.3 CHARACTERISTICS WORTH NOTING

1. The first character of a file name must be a letter or '@'.
2. If '0000' is given as the program start (EXEC) address, MPX stores $FFFF in the Directory. This is no problem since no program should begin execution at address '0000' since this would destroy essential temporary disk parameters.
3. If the beginning and/or ending addresses are not supplied in the SAVE command, '0000' ia assumed.
4. If the ending address is not supplied in a SAVE command, ENDA = BEGA.
5. Using the RENAME command, a file may be given the same name as another existing file. Only the first occurrence of a file name is recognized.
6. Disk space occupied by a deleted file is made available to the file preceding it in the directory.
7. The ALLOCATE command reserves 10 sectors following the last file entered in the directory for use by this file.

8.4 DISK RESIDENT (TRANSIENT) COMMANDS

Following is a collection of the Disk Resident commands included on the SYSTEM DISKETTE. The assembler source for each command is contained on the SYSTEM DISKETTE.

All of the utility programs use the same ERROR codes as MINIDOS or MPX. ERROR 8 (What?) indicates an error in parameters.

BACKUP
AN MPX UTILITY PROGRAM

THIS TRANSIENT MPX COMMAND WILL BACK UP A DISKETTE USING A FAST, EFFICIENT METHOD OF COPYING WITH FULL VERIFICATION OF EVERY SECTOR WRITTEN.

THE PROGRAM WILL ASK THE USER TO ENTER THE DRIVE TO COPY FROM, THE DRIVE TO COPY TO, AND THE RANGE OF SECTORS TO BE COPIED. ALL ENTRIES HAVE DEFAULT VALUES WHICH ARE SELECTED BY RESPONDING WITH RETURN TO THE PROMPT. IF AN ERROR IS MADE DURING ENTRY, SIMPLY CONTINUE TO TYPE UNTIL THE ENTRY IS CORRECT. ONLY THE LAST DIGIT ENTERED IS USED FOR DRIVE # ENTRY WHILE THE LAST 3 DIGITS ENTERED ARE USED FOR SECTORS. ANY NON-NUMERIC ENTRY WILL BE FOLLOWED BY '?' AND IGNORED.

IF AN ERROR OCCURS DURING THE BACKUP, IT WILL BE DISPLAYED ALONG WITH THE DRIVE AND SECTOR ON WHICH IT OCCURED. CONTROL WILL THEN RETURN TO MPX.

NOTE THAT ERROR 3 (BLANK SECTOR) IS NOT CONSIDERED AN ERROR DURING THE READ PHASE OF THE BACKUP. IF A BLANK SECTOR IS READ, A SECTOR CONTAINING ONLY A SINGLE 'EOT' WILL BE WRITTEN IN ITS PLACE ON THE DESTINATION DRIVE. THUS IT IS POSSIBLE TO BACKUP A RANGE OF SECTORS CONTAINING EMBEDDED EMPTY SECTORS.

ONCE THE BACKUP HAS SUCCESSFULLY COMPLETED, IT WILL DISPLAY A MESSAGE INDICATING THAT FACT AND RETURN TO MPX.

SINCE THE BACKUP COMMAND RESIDES IN RAM BEGINNING AT ADDRESS $0100 IT CAN NOT BE CALLED FROM OTHER PROGRAMS SUCH AS SUPER BASIC.

COPY
AN MPX UTILITY PROGRAM

THE COPY TRANSIENT COMMAND PROVIDES A METHOD FOR COPYING FILES OF ANY TYPE FROM ONE DISKETTE TO ANOTHER, OR FROM ONE PART OF A DISKETTE TO ANOTHER. IT ALSO ALLOWS COPYING A FILE WHICH IS NOT IN THE MPX DIRECTORY (DSSS) TO A NAMED FILE, THUS ESTABLISHING IT IN THE DIRECTORY.

THE FORMAT OF THE COPY COMMAND IS AS FOLLOWS:

```
N/COPY <SOURCE> <DESTINATION>
N/COPY <FILE 1> <FILE 2>       COPY NAMED FILES
N/COPY <DSSS> <FILE 2>         COPY UNNAMED FILES
```

THE 'N' IS THE OPTIONAL DRIVE NUMBER WHERE THE COPY TRANSIENT COMMAND PROGRAM RESIDES. FILE NAMES MAY BE PRECEDED BY AN OPTIONAL DRIVE NUMBER FOLLOWED BY A SLASH, AS '2/EDITOR'. IF NO DRIVE IS PRESENT DRIVE 1 IS ASSUMED. THE DESTINATION FILE MUST ALWAYS BE NAMED, I.E. IT IS NOT POSSIBLE TO COPY DSSS TO DSSS.

EXAMPLES:

```
2/COPY 1010 TEST
2/COPY TEST 2/TEST
COPY COPY 2/COPY
```

THE FILE TO BE COPIED TO MUST NOT EXIST AT THE TIME THE COPY COMMAND IS INVOKED. IF IT DOES ERROR 9 (ALLOCATION ERROR) WILL BE RETURNED. THE FIRST LETTER OF A FILE NAME MUST BE ALPHABETIC OR '@', OR ERROR 8 (WHAT?) WILL BE DISPLAYED.

COPY RETURNS TO MPX WHEN COMPLETED. BECAUSE OF THE LARGE AMOUNT OF MEMORY USED FOR BUFFERS, THE COPY COMMAND RESIDES IN USER RAM AND THUS MAY NOT BE CALLED FROM OTHER PROGRAMS SUCH AS BASIC.

CREATE
AN MPX UTILITY PROGRAM

THIS DISK TRANSIENT PROGRAM IS USED TO CREATE DATA FILES ON DISK. THE FORMAT FOR USING IT IS AS FOLLOWS:

```
N/CREATE <FILE> <SIZE>
```

'N' IS THE OPTIONAL DRIVE NUMBER OF THE DISK WHICH CONTAINS THE CREATE COMMAND. <FILE> IS THE FILE NAME WHICH IS 1 TO 6 CHARACTERS. THE FIRST CHARACTER MUST BE ALPHABETIC OR '@'. THE DRIVE NUMBER THAT THE FILE IS TO BE CREATED ON IS PLACED IN FRONT OF THE NAME WITH A SLASH AS IN THESE EXAMPLES:

```
2/TEST          CREATE 'TEST' ON DRIVE 2
DUMMY           CREATE 'DUMMY' ON DRIVE 1
```

NOTE THAT IF NO DRIVE IS GIVEN OR IF IT IS INVALID DRIVE 1 IS ASSUMED.

THE <SIZE> ENTRY IS THE DECIMAL SIZE OF THE FILE IN SECTORS. IF IT IS MISSING OR ZERO A DEFAULT OF 10 IS ASSUMED.

ALL MPX ERROR CODES APPLY TO CREATE. NOTE THAT IF AN ATTEMPT IS MADE TO CREATE A FILE WHICH ALREADY EXISTS, ERROR 9 (ALLOCATION) IS GIVEN. ALL PARAMETER ERRORS RECEIVE ERROR 8 (WHAT?).

WHEN A FILE IS CREATED, EVERY SECTOR IS WRITTEN WITH A SINGLE 'EOT' ($04) CHARACTER. ALL SECTORS ARE LINKED. THE DIRECTORY ENTRY WILL SHOW 0 FOR THE START ADDRESS, THE NUMBER OF SECTORS ALLOCATED FOR THE END ADDRESS, AND $FFFF FOR EXECUTION ADDRESS.

SINCE THE CREATE UTILITY RESIDES ENTIRELY IN RAM BEGINNING AT $A400, THIS UTILITY MAY BE CALLED AND USED WITHIN OTHER PROGRAMS SUCH AS THE PERCOM TEXT EDITOR OR SUPER BASIC.

PACK
AN MPX UTILITY PROGRAM

THE PACK TRANSIENT COMMAND ALLOWS THE SPACE FREED UP BY FILE DELETION TO BE RECLAIMED FOR REUSE. IT DOES THIS BY COPYING THE FILES ON THE DISKETTE TO 'CLOSE UP' THE GAPS BETWEEN THEM.

INVOKING THE PACK COMMAND IS A SIMPLE PROCEDURE AS FOLLOWS:

```
N/PACK D                PACK DISK ON DRIVE # D
```

'N' IS THE OPTIONAL NUMBER OF THE DRIVE CONTAINING THE PACK PROGRAM. THE DRIVE NUMBER 'D' IS NOT OPTIONAL AND MUST BE BETWEEN 1 AND 3. IF NO DRIVE IS GIVEN ERROR 8 (WHAT?) WILL BE DISPLAYED. IF THE DRIVE NUMBER IS ILLEGAL, ERROR 2 WILL BE GIVEN.

THE SEQUENCE OF OPERATIONS DURING PACK REQUIRES MODIFICATION OF THE DISK DIRECTORY. CONSEQUENTLY GREAT CARE MUST BE TAKEN TO INSURE THAT THE PACK COMMAND COMPLETES BEFORE DOING ANYTHING TO THE DISK. FAILURE TO OBSERVE THIS PRECAUTION WILL RESULT IN A DISK WITH A CORRUPT DIRECTORY, AND VERY PROBABLY SOME DATA LOSS. IT IS RECOMMENDED THAT THE BACKUP COMMAND BE USED TO MAKE A DUPLICATE OF ANY DISK TO BE PACKED SO THAT NOTHING WILL BE LOST SHOULD THE PACK COMMAND FAIL.

FOLLOWING COMPLETION PACK RETURNS TO MPX. DUE TO THE BUFFER REQUIREMENTS OF PACK, THIS COMMAND RESIDES IN USER RAM BEGINNING AT LOCATION $100. PACK, THEREFORE, MAY NOT BE INVOKED BY OTHER PROGRAMS USING THIS RAM, SUCH AS SUPER BASIC.

PRINT DISK DIRECTORY (PDIR)
AN MPX UTILITY PROGRAM

THIS UTILITY PERMITS THE MPX DIRECTORY TO BE PRINTED ON THE SYSTEM PRINTER. THE COMMAND IS IN THE FORM:

```
N/PDIR <DRIVE>
```

'N' REFERS TO THE DRIVE CONTAINING THE PDIR UTILITY. DEFAULT VALUE IS DRIVE #1. <DRIVE> REFERS TO THE DRIVE WHOSE DIRECTORY YOU WISH TO PRINT. DEFAULT VALUE IS DRIVE #1.

EXAMPLES:

```
PDIR
2/PDIR
2/PDIR 2
```

THIS UTILITY ASSUMES AN ASCII SERIAL OR PARALLEL PRINTER IS CONNECTED TO AN MP-S (SERIAL) OR MP-L (PARALLEL) INTERFACE IN PORT #7 OF THE SWTP COMPUTER.

APPENDIX A
MINIDOS and MPX ERROR CODES

0 - DISK MISSING, DISK GATE OPEN
1 - DISK PROTECTED, WRITE NOT PERMITTED
2 - INVALID SECTOR NUMBER, IMPROPER ENTRY
3 - BLANK SECTOR, or SEEK ERROR
4 - DISK OVERRUN, EXCEEDED 400 (770) SECTORS
5 - DISK READ ERROR, TRIED 9 TIMES WITHOUT SUCCESS

6 - FILE NAME NOT FOUND
7 - DIRECTORY FULL
8 - COMMAND NOT UNDERSTOOD (WHAT?)
9 - ALLOCATION ERROR
A - BAD ADDRESS

APPENDIX B
LOW MEMORY LOCATIONS USED BY DISK SOFTWARE

ADDRESS	FUNCTION
0000	DRIVE NUMBER (MS 2 BITS)
0001	TRACK NUMBER
0002	SECTOR NUMBER
0003	BACK LINK TRACK
0004	BACK LINK SECTOR
0005	FORWARD LINK TRACK
0006	FORWARD LINK SECTOR
0007	BYTE COUNT (0 = 256)
0008	TARGET ADDRESS (HI BYTE)
0009	TARGET ADDRESS (LO BYTE)
000A	FILE TYPE CODE
000B	CRC (HI BYTE)
000C	CRC (LO BYTE)
000D	POSTAMBLE (HI BYTE)
000E	POSTAMBLE (LO BYTE)
000F	CURRENT TRACK NUMBER
0010	TRACK FOR DRIVE #0
0011	TRACK FOR DRIVE #1
0012	TRACK FOR DRIVE #2
0013	TRACK FOR DRIVE #3
0014	CONTINUATION ADDRESS (TA) (2 BYTES)
0016	ALTERNATE TARGET ADDRESS (TW) (2 BYTES)
0018	TEMPORARY INDEX REGISTER STORAGE (2 BYTES)
001A	READ RETRY COUNTER
001A	PROGRAM START ADDRESS (EXEC) (2 BYTES)
001C	TEMPORARY STORAGE (2 BYTES)
001E	ENDING ADDRESS (ENDA) (2 BYTES)

APPENDIX C
MPX ROM RESIDENT COMMAND SUMMARY

MPX is an extension of the Percom MINIDOS operating system. It permits disk files to be manipulated by files names of 6 characters or less. Up to 31 files are supported. The resident commands are:

```
S(AVE) <NAME> <BEGIN> <END> [EXEC]
L(OAD) <NAME>              LOAD FILE INTO MEMORY
D(ELETE) <NAME>            DELETE FILE FROM DIRECTORY
F(ILES)                    LISTS FILES ON TERMINAL
I(NIT)                     INITIALIZES DISK DIRECTORY
R(ENAME) <THIS> <THAT>
A(LLOCATE)                 RESERVES 10 BLOCKS FOLLOWING
                           LAST FILE IN DIRECTORY
J(UMP) <ADDRESS>           JUMP TO SPECIFIED ADDRESS
X(IT)                      EXIT TO MONITOR
M(INIDOS)                  ESCAPE TO MINIDOS
```

Commands consist of single letters. If more than one character is entered, the directory will be searched for a file with the name of the given command. If such a file is found, it will be loaded and executed.

Spaces (blanks) are delimiters between entries.

Disk space occupied by a deleted file is made available to the file preceding it in the directory.

To PROTECT a file, make the first character in the file name '@'. For example: @NAME

To UNPROTECT the file, RENAME the file without the '@'.

The ALLOCATE command reserves 10 sectors following the last file for use by the last file.

Drives other than drive #1 may be selected by preceding the command with the drive number and a slash (/). For example, to load a file from drive #2, type: 2/L <NAME>

Control-X is used to cancel a command line
Control-H is used to backspace in command line
RETURN terminates command line

APPENDIX D
SYSTEM MONITOR I/O SUBROUTINE ENTRY ADDRESSES

MINIDOS, MPX, and the MPX disk resident commands depend on the system ROM Monitor (MIKBUG, EXBUG, etc.) I/O subroutnes for operator Data Terminal I/O. Since the I/O subroutine address vectors differ for the different Monitors, the user must make sure the I/O subroutine vectors in the MINIDOS and MPX ROMs match the Monitor with which the LFD-400/800 disk system is used.

The MPX Disk Resident Utility commands are supplied with I/O subroutine vectors set for MIKBUG. If the computing system uses a Monitor other than MIKBUG these I/O subroutine vectors must be changed. The MPX Utility descriptions in Section 8.4 of this manual identify the affected subroutine vector locations and describe the procedure for modification.

The following table is a summary of the I/O subroutine entry points used by MINIDOS and MPX for the System Monitors supported by the Percom LFD-400/800. All addresses are Hexadecimal values.

	MIKBUG	MICROBUG	MINIBUG II	MINIBUG III	TVBUG	EXBUG (2.1)	SYSMON (CMS)
INCH	E1AC	FD33	E11F	E133	F800	F015	F9D1
OUTCH	E1D1	FD26	E108	E126	F803	F018	F9E8
OUTS	E0CC	FD9A	E180	E19A	F9B4	F02A	F8EB
OUTHR	E06B	FD1C	E0FE	E11C	F986	F0D0*	F881
OUT4HS	E0C8	FD96	E17C	E196	F80F	F01E	F8E7
PDATA1	E07E	FD4B	E130	E14B	F806	F027	F894
CONTRL	E0E3	FC65	E040		F821	F564	F81C
BADDR	E047	FCF8	E0D9	E0F8	F809	F00F*	F864
I/O TYPE	PIA*	ACIA	ACIA	ACIA	PIA*	ACIA	ACIA
ADDRES	8004	8408	8008	8008	F404	FCF4	E3C0

NOTES:

1. F0D0 is not a guaranteed entry point in EXBUG. Consequently the entry may not be compatible with other versions of EXBUG.
2. F00F is the EXBUG 'XINADD' routine. To duplicate the BADDR function of the other Monitors, the subroutine call must be followed by an "LDX 0,X" instruction.
3. MIKBUG uses a PIA to 'Bit-Bang' serial data to and from the Data Terminal.
4. TVBUG uses a PIA for keyboard input only. Output is to a color CRT or TV.

APPENDIX E

SUBJECT: MODIFYING THE SWTP MP-A PROCESSOR CIRCUIT CARD AND 4K RAM MEMORY CIRCUIT CARD TO PERMIT A 4K RAM MEMORY CARD TO BE LOCATED AT ADDRESS $A000.

THE SWTP MP-A PROCESSOR CIRCUIT CARD CONTAINS A 128 BYTE RAM MEMORY AT ADDRESS $A000. THE MANNER IN WHICH THE MP-A CIRCUIT IS CONFIGURED PREVENTS LOCATING A LARGER MEMORY IN THE $A000-$BFFF ADDRESS ZONE UNLESS THE PROCESSOR CARD IS MODIFIED. FORTUNATLY THE MODIFICATION IS QUITE SIMPLE INVOLVING ONLY ONE CIRCUIT TRACE CUT AND ONE JUMPER. IF YOU ARE USING THE NEWER MP-A2 PROCESSOR CIRCUIT CARD NO MODIFICA- TION OF THE PROCESSOR CARD IS REQUIRED.

THE 4K RAM MEMORY CARD WAS NOT DESIGNED TO RESPOND TO MEMORY ADDRESSES HIGHER THAN 32K. WITH MINOR MODIFICATION, THE 4K RAM MEMORY CARD WILL OPERATE AT ADDRESSES ABOVE 32K BUT ONCE MODIFIED IT WILL NO LONGER OPERATE BELOW 32K WITHOUT REMODIFICATION.

MODIFYING THE MP-A PROCESSOR

1. ON THE COMPONENT (TOP) SIDE OF THE CIRCUIT CARD LOCATE THE CIRCUIT TRACE CONNECTING IC16 PIN 10 TO IC16 PIN 13. CUT THE TRACE WITH A SMALL KNIFE (XACTO KNIFE IS IDEAL) NEAR IC16 PIN 13.
2. ON THE SOLDER (BOTTOM) SIDE OF THE CIRCUIT CARD CONNECT A PIECE OF #24 OR #26 INSULATED HOOKUP WIRE FROM IC16 PIN 13 TO IC2 PIN 14. DOUBLE CHECK THE CONNECTION. MAKE SURE YOU HAVE IDENTIFIED IC2 PIN 14.

MODIFYING THE 4K RAM MEMORY CARD

1. ON THE SOLDER (BOTTOM) SIDE OF THE CIRCUIT CARD LOCATE THE CIRCUIT TRACE CONNECTING IC22 PIN 6 TO IC24 PIN 1. CUT THE TRACE NEAR IC22 PIN 6.
2. ON THE COMPONENT (TOP) SIDE OF THE CIRCUIT CARD LOCATE THE CIRCUIT TRACE CONNECTING IC22 PIN 4 TO EDGE CONNECTOR PIN A15. CUT THE TRACE NEAR IC22 PIN 4.
3. ON THE SOLDER (BOTTOM) SIDE OF THE CIRCUIT CARD CONNECT A PIECE OF #24 OR #26 INSULATED HOOKUP WIRE FROM IC22 PIN 4 TO IC24 PIN 2.
4. ALSO ON THE SOLDER SIDE OF THE CIRCUIT CARD CONNECT A PIECE OF #24 OR #26 INSULATED HOOKUP WIRE FROM IC22 PIN 6 TO EDGE CONNECTOR PIN A15.
5. PROGRAM THE MEMORY CARD FOR ADDRESS $A000 BY CONNECTING THE ADDRESS SELECT PROGRAMMING JUMPER TO THE `THIRD` CONTACT (HOLE #2).

IN CASE OF DIFFICULTY
RE-READ THE INSTRUCTION MANUAL!

Clean the Mother-Board contact pins in SS-50 bus computers with a 'Pink-Pearl' eraser. Move the disk controller circuit card to a different slot (a connector contact in the Mother-Board may be intermittent or poorly soldered). Remove and reinstall all circuit cards in the computer. Upend the computer and shake out the dust and trash.

DRIVE MOTORS NEVER TURN ON

Turn on the Disk Drive power switch, make sure the ribbon cable is firmly connected to the drive and the controller card, make sure the drive power supply is firmly plugged into the drive power connector.

DRIVE MOTORS RUN CONTINUOUSLY

Drive ribbon cable connector may be reversed at either the drive or the controller card.

ERROR #0

Make sure you are selecting the correct drive and that a diskette is correctly mounted. Double check the drive select programming jumpers. Diskette may be binding in drive, remove and re-install or try another diskette. Disk drive belt may have come off pulley (this is usually caused by a binding diskette).

ERROR #1

Diskette is write protected. Writing on disk is not permitted unless you remove the Write Protect tape. Ribbon cable connector may not be installed securely.

ERROR #2

Make sure you are providing proper parameters. This problem will also occur if you do not have memory in location $0000 - $001F or if the memory is defective.

ERROR #3

The Operating System could not find anything written in the specified sector or the sector header information did not match the expected values (Seek Error). Make sure you have given MINIDOS proper directions. Make sure the MPX directory has been initialized. Try reading a diskette known to have data in a specific sector. Disk data may have been destroyed by a stray magnetic field. Check the hole in the center of the diskette to see if it has been 'crimped' or distorted causing the diskette to rotate 'off-center'.

ERROR #4

You may have tried to put too much data on the diskette! PACK the diskette to recover deleted file space. Relocate the data elsewhere on the diskette or use another diskette.

ERROR #5

The LFD-400/800 performs a very comprehensive check of the disk data validity AFTER it is read into memory. Consequently memory defects may also report as disk read errors.

1. Make sure you have RAM memory at the locations into which the disk is storing data.
2. Run the memory diagnostic (MEMTST) across the memory in question. Remember the disk controller uses RAM memory from address $0000 thru $001F. This memory must also function properly. Do not try to store data in this region of memory because important disk parameters will be destroyed producing unpredictable results.
3. Do not try to store data in the program STACK. This will CRASH the STACK and produce unpredictable results. STACK locations vary from program to program and system to system.

To separate DISK problems from MEMORY problems, save the data in the MINIDOS ROM ($C000 - $CFFF) to disk, then read back into memory to the same address as the ROM. Since the ROM is not altered by data read from the disk, and since the error check is made using a code stored on the disk, a satisfactory read virtually clears the disk of suspicion.

If you are not able to resolve your problem by performing the above checks, describe in writing as clearly as possible the problem you are having and return the entire disk system for checkout and repair. If possible include a diskette exhibiting the problem with a complete description.

DO NOT ATTEMPT TO ADJUST OR REPAIR THE DISK DRIVE UNIT. Special equipment and tools are required and considerable (expensive) damage can be done by attempting to work on these units without proper training.

OUT-OF-WARRANTY repairs are performed for a labor charge plus parts and shipping.

	LABOR CHARGE
CONTROLLER CARD AND CABLE	$17.50
DRIVE POWER SUPPLY	15.00
MINI-DISK DRIVES	39.50

If we find that a circuit card, drive, power supply, cable, or complete unit is functioning properly as received and does not require any service, the CHECKOUT CHARGE is $15.00 plus return shipping and insurance. Do not enclose any payment. The unit(s) will be returned C.O.D. for authorized repairs and shipping.

When returning a unit for repair, pack it in a large carton with at least 3" of padding on all sides. We will not attempt to service any unit if there is shipping damage until the claim is settled (a real hassle). Ship prepaid by UPS or INSURED PARCEL POST to:

Percom Data Co.
Service Dept.
211 N. Kirby
Garland, Tx 75042

We try to turn most repairs around within one week.

PERCOM DATA CO. INC.
211 N. Kirby
Garland, TX 75042
(214) 272-3421

NOTICE

All COMPUTER PROGRAMS sold or distributed by PERCOM DATA CO. INC. are sold or distributed on an AS-IS basis WITHOUT WARRANTY.

PERCOM DATA CO. INC. shall have no LIABILITY or responsibility to customers, or any other person or entity with respect to any LIABILITY, LOSS, OR DAMAGE caused or alleged to be caused directly or indirectly by equipment or computer programs sold by PERCOM DATA CO. INC. including but not limited to any interruption of service, loss of business or anticipatory profits or consequential damages resulting from the use or operation of such equipment or computer programs.

Good data processing procedure dictates that the user test the program, run and test sample sets of data, and run the system in parallel with the system previously in use for a period of time adequate to insure that results of operation of the computer or program are satisfactory.

PERCOM

PERCOM DATA COMPANY, INC.

ATTENTION - NOTICE - WARNING

This program is designed for use with a Southwest Technical Products M-6800 Computer with the following configuration:

MP-C or MP-S Console Interface in Port #1 ($8004)
MP-L or MP-LA Printer Interface in Port #7 ($801C)
Percom LFD-400 Disk System with MINIDOS (V1.4)
MIKBUG(tm) or SWTBUG(tm) ROM monitor
MP-A or MP-A2 processor card operating with 890KHZ to 1.0mHZ clock.
RAM memory sufficient to run the software.

Although information for adapting to other configurations may be available, we do not guarantee nor can we support operation of the program on differing configurations. If you wish to have us convert this program for a different configuration or do any custom software work, we will be happy to submit a quotation in response to your written specification. Please address such requests to:

Percom Consulting
P. O. Box 40598
Garland, TX 75040

A great deal of time and effort has gone into the creation of this program. In spite of this effort, it is possible (even likely) there are bugs. If you discover a problem or have difficulty, *please do not contact us by phone.* We have found that phone calls are not very productive. There is no guarantee someone knowledgeable will be available when you call. Furthermore, it is very difficult to describe software problems on the telephone and even more difficult to solve them while you wait.

Instead describe the problem in writing. Include examples and enough information to permit us to recreate the problem. We then can examine all pertinent facts and listings and respond to your problem in a more knowledgeable and timely fashion.

PERCOM DATA COMPANY

PERCOM SOFTWARE PROBLEM REPORT NUMBER _____

DATE: _____ RECEIVED BY: _____

CUSTOMER NAME: _____

CUSTOMER ADDRESS: _____

CUSTOMER PHONE: _____

PROBLEM DESCRIPTION

SOFTWARE PRODUCT _____ VERSION _____

HARDWARE CONFIGURATION _____

_____

PROBLEM DESCRIPTION: _____

_____

_____

_____

_____

PROBLEM RESOLUTION

DATE ANSWERED: _____ REPLY BY: _____

REPLY VIA: _____

PROBLEM SOLUTION: _____

_____

_____

_____

_____

```
                        NAM   MINIDOS
                       * MINIDOS (TM) 6800 REV 1.4
                       * COPYRIGHT 1978 PERCOM DATA CO.
                       * WRITTEN BY H.A. MAUCH
                       * REVISED 8-20-78

   (C000)              BASE   EQU   $C000
                       * MONITOR LINKS
   (A07F)              STACK  EQU   $A07F      STACK LOCATION
   (A048)              XFER   EQU   $A048      MONITOR TRANSFER ADDRESS
   (E047)              BADDR  EQU   $E047      BUILD ADDRESS
   (E06B)              OUTHR  EQU   $E06B      PRINT RIGHT HEX DIGIT
   (E07E)              PDATA1 EQU   $E07E      PRINT CHARACTER STRING
   (E0CC)              OUTS   EQU   $E0CC      PRINT SPACE CHARACTER
   (E0E3)              MONITR EQU   $E0E3      MONITOR RE-ENTRY POINT
   (E1AC)              INEEE  EQU   $E1AC      INPUT A CHARACTER
                       * INPUT PORTS ASSIGNMENT
   (CC00)              STATUS EQU   $CC00      CONTROLLER STATUS
   (CC01)              RDDTA  EQU   $CC01      RECEIVED DATA
   (CC02)              SECTOR EQU   $CC02      SECTOR COUNTER
   (CC03)              DVSTAT EQU   $CC03      DRIVE STATUS
   (CC04)              RRSRT  EQU   $CC04      RECEIVER RESTART PULSE
   (CC05)              MOTON  EQU   $CC05      MOTOR ON PULSE
   (CC06)              MOTOFF EQU   $CC06      MOTOR OFF PULSE
                       * OUTPUT PORTS ASSIGNMENT
   (CC00)              SYNC   EQU   $CC00      SYNC WORD PORT
   (CC01)              WRTDTA EQU   $CC01      WRITE DATA PORT
   (CC02)              FILL   EQU   $CC02      FILL WORD PORT
   (CC03)              DRVSLT EQU   $CC03      DRIVE AND TRACK SELECT
   (CC04)              WRTPLS EQU   $CC04      WRITE PULSE
                       * DRIVE STATUS BITS (DVSTAT)
   (0001)              WTPBIT EQU   $1         WRITE PROTECT BIT
   (0002)              TRKBIT EQU   $2         TRACK ZERO BIT
   (0004)              MTRBIT EQU   $4         MOTOR TEST BIT
   (0008)              WRTBIT EQU   $8         WRITE GATE BIT
   (0010)              SECBIT EQU   $10        SECTOR PULSE BIT
   (0020)              IDXBIT EQU   $20        INDEX PULSE BIT
                       * DRIVE LIMITS
   (0022)              TRKLMT EQU   34         USE 76 FOR MICROPOLIS DRIVES
   (003F)              TRKMSK EQU   $3F        USE $7F FOR     "         "
   (0009)              SECLMT EQU   9          USE 15 IF DOUBLE DENSITY

   (0000)                     ORG   0
                       * SECTOR HEADER TEMPORY STORAGE
   (0000)              HEADER EQU   *
0000                   DRV    RMB   1         DESIRED DRIVE (MS 2 BITS)
   (0001)              TRKSEC EQU   *
0001                   TRK    RMB   1         DESIRED TRACK
0002                   SCTR   RMB   1         DESIRED SECTOR
0003                   BAKLNK RMB   2         BACKWARD LINK
0005                   FWDLNK RMB   2         FORWARD LINK
0007                   BYTCNT RMB   1         SECTOR BYTE COUNT
0008                   ADDRES RMB   2         DATA ADDRESS VECTOR
000A                   FILTYP RMB   1         FILE TYPE CODE
```

```
000B            CKSUM   RMB    2          CHECKSUM
000D            POSTAM  RMB    2          POSTAMBLE

000F            CRNTRK  RMB    5          CRNT TRK AND TRK HISTORY
0014            TA      RMB    2          CONTINUATION ADDRESS
0016            TW      RMB    2          ALTERNATE TARGET ADDRESS
0018            XTEMP   RMB    2          INDEX REGISTER STORAGE
  (001A)        RETRY   EQU    *          RETRY COUNTER
001A            EXEC    RMB    2          PROG STRT ADDRESS
001C            TEMP    RMB    2          TEMP STORAGE
001E            ENDA    RMB    2          END ADDRESS

  (C000)                ORG    BASE
                *INTERNAL FUNCTION JUMP TABLE
C000 7E C39F            JMP    MINDOS     MINIDOS CMD PROCESSOR
C003 7E C36B            JMP    CVTDTS     CONVERT DSSS TO PHYSICAL SECTOR
C006 7E C267  RESTORE   JMP    GTKOX      GO TO TRACK ZERO
C009 7E C23F  SEEK      JMP    SEEKX      SEEK TRACK
C00C 7E C034  RDSEC     JMP    RDSECX     READ SECTOR
C00F 7E C115  WTSEC     JMP    WTSECX     WRITE SECTOR
C012 7E C16F  SLTDRV    JMP    DCHECK     SLT DRV, START MTR, CK DISK, SEEK TRK
C015 7E C1D2  STMTR     JMP    START      START MOTOR (DELAY IF NECESSARY)
C018 7E C2DC  SAVE      JMP    SAVEX      SAVE A MEMORY IMAGE FILE
C01B 7E C2CB  LOAD      JMP    LOADX      LOAD A MEMORY IMAGE FILE
C01E 7E C353  TYPERR    JMP    TYPERX     TYPE ERROR MESSAGES
C021 7E C2B3  FWDCAL    JMP    FWDCLX     CALCULATE FORWARD LINK
C024 7E C3DC  LENGTH    JMP    LNTH       CALCULATE BLOCK LENGTH
C027 86 FF    INITRK    LDA A  #$FF       INITIALIZE TRACK REGISTERS
C029 97 0F              STA A  CRNTRK
C02B 97 10              STA A  CRNTRK+1    DRIVE 0 TRACK
C02D 97 11              STA A  CRNTRK+2    DRIVE 1 TRACK
C02F 97 12              STA A  CRNTRK+3    DRIVE 2 TRACK
C031 97 13              STA A  CRNTRK+4    DRIVE 3 TRACK
C033 39       RDRTN     RTS

                *READ SECTOR-3X3 RETRIES WITH JOGS
C034 BD C16F  RDSECX    JSR    DCHECK     CHECK PARAMETERS
C037 25 FA              BCS    RDRTN      RETURN IF ERROR
C039 8D 19              BSR    RD3
C03B 24 F6              BCC    RDRTN      RETURN IF OK
C03D 27 F4              BEQ    RDRTN      RETURN IF DISK MISSING
C03F BD C27F            JSR    TKIN       JOG HEAD
C042 BD C27B            JSR    TKOT
C045 BD C25D            JSR    SETTLE
C048 8D 0A              BSR    RD3
C04A 24 E7              BCC    RDRTN      RETURN IF OK
C04C 27 E5              BEQ    RDRTN      RETURN IF DISK MISSING
C04E BD C267            JSR    GTKOX      HOME HEAD
C051 BD C23F            JSR    SEEKX      SEEK TRACK

                *READ THREE TIMES
C054 86 03    RD3       LDA A  #3
C056 97 1A              STA A  RETRY      SETUP RETRY CNTR
C058 86 FB              LDA A  #$FB       SET SYNC CODE
C05A B7 CC00            STA A  SYNC
C05D BD C20C  READ      JSR    GTSEC      GET SCTR
```

```
C060 25 D1            BCS   RDRTN     BRANCH IF DISK MISSING (A=0)
C062 36               PSH A           SAVE CCR
C063 86 1E            LDA A #30       SHORT DELAY (180-200 USEC)
C065 4A       R1    1- DEC A
C066 26 FD          2 BNE   R1
C068 B6 CC04          LDA A RRSRT     START RCVR
C06B 8D 62            BSR   IN        GET SYNC CHAR
C06D 24 45            BCC   RDERR     BRANCH IF EMPTY SCTR
C06F 8D 5E            BSR   IN        GET DRV & TRK
C071 16               TAB             SAVE A COPY
C072 98 01            EOR A TRK       CK FOR PROPER TRK
C074 84 3F            AND A #TRKMSK
C076 26 3C            BNE   RDERR     BR IF SEEK ERR (C IS SET)
C078 D7 01            STA B TRK
C07A 8D 53            BSR   IN
C07C 91 02            CMP A SCTR      CK FOR PROPER SCTR
C07E 0D               SEC             SET ERROR FLAG
C07F 26 33            BNE   RDERR     BR IF SEEK ERR
C081 CE 0003          LDX   #BAKLNK   GET REST OF HDR
C084 C6 08            LDA B #8
C086 8D 5C            BSR   INPUT
C088 DE 16            LDX   TW        CHK FOR ALT TRGT
C08A 8C FFFF          CPX   #$FFFF
C08D 26 02            BNE   R2
C08F DE 08            LDX   ADDRES
C091 D6 07    R2      LDA B BYTCNT
C093 8D 4F            BSR   INPUT     INPUT DATA
C095 DF 14            STX   TA        SAVE LAST ADDRESS
C097 CE 000B          LDX   #CKSUM    INPUT CKSUM & POSTAMBLE
C09A C6 04            LDA B #4
C09C 8D 46            BSR   INPUT
C09E 32               PUL A           RESTORE CCR (TURN ON INT)
C09F 06               TAP
C0A0 DE 16            LDX   TW        CHK FOR ALT TRGT
C0A2 8C FFFF          CPX   #$FFFF
C0A5 26 02            BNE   R3
C0A7 DE 08            LDX   ADDRES
C0A9 8D 49    R3      BSR   CRC       CHECK CRC
C0AB 9C 0B            CPX   CKSUM
C0AD 0D               SEC
C0AE 26 06            BNE   RDERRX    BRANCH IF ERROR
C0B0 8D 13            BSR   TWTA
C0B2 0C               CLC
C0B3 39               RTS
C0B4 32       RDERR   PUL A           RESTORE CCR
C0B5 06               TAP
C0B6 7A 001A  RDERRX  DEC   RETRY
C0B9 26 A2            BNE   READ
C0BB 8D 08            BSR   TWTA
C0BD 86 05            LDA A #5        READ ERROR (A=5)
C0BF 25 03            BCS   RDERTN
C0C1 86 03            LDA A #3        EMPTY SECTOR (A=3)
C0C3 0D               SEC
C0C4 39       RDERTN  RTS

C0C5 DE 16    TWTA    LDX   TW
```

```
C0C7 8C FFFF          CPX   #$FFFF
C0CA 27 02            BEQ   RS
C0CC DE 14            LDX   TA
C0CE 39         RS    RTS

C0CF B6 CC00    IN    LDA A STATUS
C0D2 46               ROR A
C0D3 25 0B            BCS   IN1
C0D5 B6 CC02          LDA A SECTOR    STILL IN THIS SCTR?
C0D8 84 0F            AND A #$0F
C0DA 91 02            CMP A SCTR
C0DC 27 F1            BEQ   IN        BR IF YES
C0DE 0C               CLC
C0DF 39               RTS
C0E0 B6 CC01    IN1   LDA A RDDTA     (CARRY IS SET)
C0E3 39               RTS
C0E4 B6 CC00    INPUT LDA A STATUS
C0E7 46               ROR A
C0E8 24 FA            BCC   INPUT
C0EA B6 CC01          LDA A RDDTA
C0ED A7 00            STA A 0,X
C0EF 08               INX
C0F0 5A               DEC B
C0F1 26 F1            BNE   INPUT
C0F3 39               RTS

                *CALCULATE CRC CODE - TAKES 8 MSEC (1MHZ CLOCK)
C0F4 D6 07      CRC   LDA B BYTCNT
C0F6 4F               CLR A
C0F7 97 19            STA A XTEMP+1
C0F9 8D 0C            BSR   CX
C0FB CE 0001          LDX   #TRK
C0FE C6 0A            LDA B #$A
C100 8D 05            BSR   CX
C102 97 18            STA A XTEMP
C104 DE 18            LDX   XTEMP
C106 39               RTS
C107 A8 00      CX    EOR A 0,X
C109 48               ASL A
C10A 79 0019          ROL   XTEMP+1
C10D 24 01            BCC   C1
C10F 4C               INC A
C110 08         C1    INX
C111 5A               DEC B
C112 26 F3            BNE   CX
C114 39               RTS

                *WRITE A SECTOR
C115 BD C16F    WTSECX JSR  DCHECK    SELECT AND CHECK DRIVE
C118 25 40            BCS   WTERR
C11A 86 01            LDA A #1
C11C B5 CC03          BIT A DVSTAT    TEST WRITE PROTECT
C11F 26 02            BNE   WT1       BRANCH IF NOT PROTECTED
C121 0D               SEC             WRITE PROTECTED (A=1)
C122 39               RTS             RETURN IF DISK PROTECTED
C123 DE 14      WT1   LDX   TA
```

```
C125 8D CD            BSR   CRC
C127 DF 0B            STX   CKSUM
C129 86 FF            LDA A #$FF
C12B B7 CC02          STA A FILL        SET FILL WORD
C12E BD C20C          JSR   GTSEC       GET SCTR
C131 25 27            BCS   WTERR       BRANCH IF DISK MISSING (A=0)
C133 36               PSH A             SAVE CONDITION CODE REGISTER
C134 B7 CC04          STA A WRTPLS      TURN ON WRITE
                    *WRITE THE LEADER
C137 C6 10            LDA B #16         WRITE 16 NULL CODES
C139 4F               CLR A
C13A 8D 28          WS3   BSR   OUT
C13C 5A               DEC B
C13D 26 FB            BNE   WS3
                    *WRITE THE SECTOR HEADER
C13F 86 FB            LDA A #$FB        SYNC BYTE
C141 CE 0000          LDX   #HEADER
C144 C6 0B            LDA B #$B
C146 8D 15            BSR   OUTPUT+2
                    *WRITE THE DATA
C148 DE 14            LDX   TA
C14A D6 07            LDA B BYTCNT      BYTE COUNT
C14C 8D 0D            BSR   OUTPUT
C14E DF 14            STX   TA
                    *WRITE THE POSTAMBLE
C150 CE 000B          LDX   #CKSUM
C153 C6 04            LDA B #4
C155 8D 04            BSR   OUTPUT
C157 32               PUL A             RESTORE CCR
C158 06               TAP
C159 0C               CLC
C15A 39             WTERR RTS

C15B A6 00          OUTPUT LDA A 0,X
C15D 8D 05            BSR   OUT
C15F 08               INX
C160 5A               DEC B
C161 26 F8            BNE   OUTPUT
C163 39               RTS
C164 36             OUT   PSH A
C165 B6 CC00          LDA A STATUS
C168 2A FB            BPL   OUT+1
C16A 32               PUL A
C16B B7 CC01          STA A WRTDTA
C16E 39               RTS

                    *CHECK DISK PARAMETERS
C16F 86 02          DCHECK LDA A #2
C171 D6 02            LDA B SCTR        SECTOR = [0,S]
C173 C1 09            CMP B #9
C175 22 0A            BHI   DCKER2      BRANCH IF ERROR
C177 D6 01            LDA B TRK         TRACK = [0,T]
C179 C4 3F            AND B #TRKMSK     MASK OFF DRIVE BITS
C17B C1 22            CMP B #TRKLMT
C17D 23 0D            BLS   DRIVE       BRANCH IF OK
C17F 4C             DCKER1 INC A        DISK OVERRUN (A=4)
```

```
C180 4C              INC A
C181 0D        DCKER2 SEC              INVALID PARAMETER (A=2)
C182 39               RTS

C183 0C        POS    CLC
C184 59               ROL B
C185 59               ROL B
C186 59               ROL B
C187 08        P1     INX
C188 5A               DEC B
C189 2A FC            BPL   P1
C18B 39               RTS

                      *SELECT DRIVE, START MOTOR, CHECK DISK, SEEK TRACK
C18C 86 C0     DRIVE  LDA A #$C0       DRIVE BITS MASK
C18E 16               TAB
C18F B4 CC03          AND A DVSTAT      GET CURRENT DRIVE #
C192 D4 00            AND B DRV         GET DESIRED DRIVE #
C194 11               CBA               COMPARE
C195 26 07            BNE   D0          BRANCH IF NOT SAME
C197 96 0F            LDA A CRNTRK      CHECK IF DRIVE IS INITIALIZED
C199 43               COM A
C19A 27 24            BEQ   D1          BRANCH IF NOT
C19C 20 2E            BRA   D2
C19E 37        D0     PSH B
C19F 37               PSH B
C1A0 36               PSH A
C1A1 CE 000F          LDX   #CRNTRK
C1A4 A6 00            LDA A 0,X         GET CURRENT TRK
C1A6 33               PUL B             FIND PROPER PIDGEON HOLE
C1A7 8D DA            BSR   POS
C1A9 A7 00            STA A 0,X         STORE CURRENT TRK
C1AB CE 000F          LDX   #CRNTRK     RETRIEVE TRK FOR NEW DRV
C1AE 33               PUL B
C1AF 8D D2            BSR   POS
C1B1 A6 00            LDA A 0,X
C1B3 97 0F            STA A CRNTRK
C1B5 B6 CC03   DA     LDA A DVSTAT
C1B8 85 08            BIT A #WRTBIT     CHECK WRITE GATE
C1BA 27 F9            BEQ   DA          WAIT UNTIL GATE TURNS OFF
C1BC 32               PUL A             SELECT NEW DRIVE
C1BD B7 CC03          STA A DRVSLT
C1C0 8D 24     D1     BSR   DRVTST      DISK MISSING TEST
C1C2 25 0D            BCS   D3          BRANCH IF DISK MISSING
C1C4 96 0F            LDA A CRNTRK      CHECK IF DRIVE IS ON LINE
C1C6 43               COM A
C1C7 26 03            BNE   D2          BRANCH IF ON LINE
C1C9 BD C267          JSR   GTKOX       RESTORE DRIVE
C1CC 8D 04     D2     BSR   START       START MOTOR
C1CE 8D 6F            BSR   SEEKX       SEEK TRACK
C1D0 0C               CLC
C1D1 39        D3     RTS

                      * START MOTOR (DELAY IF NECESSARY)
C1D2 B6 CC03   START  LDA A DVSTAT      TEST MOTOR BIT
C1D5 85 04           BIT A #MTRBIT
```

```
C1D7 27 09           BEQ   START1    BRANCH IF ALREADY ON
C1D9 7D CC05         TST   MOTON     TRIGGER MOTOR ONE-SHOT
C1DC CE 03E8         LDX   #1000     SET UP ONE SEC DELAY
C1DF BD C2A8         JSR   DELAY
C1E2 7D CC05 START1  TST   MOTON     RETRIGGER MOTOR
C1E5 39              RTS

                     * CHECK IF DISK MISSING
C1E6 8D EA     DRVTST BSR   START    START MOTOR
C1E8 CE 0000         LDX   #0        SET TIME LIMIT (1 SEC)
C1EB C6 0B           LDA B #11       SECTORS TO SYNC COUNTER
C1ED 86 10           LDA A #SECBIT   SECTOR BIT MASK
C1EF B5 CC03 DVTST1  BIT A DVSTAT    SECTOR PULSE=0?
C1F2 27 05           BEQ   DVTST2
C1F4 09              DEX             CHECK TIME LIMIT
C1F5 26 F8           BNE   DVTST1
C1F7 20 08           BRA   DVTST3    BRANCH IF TIME LIMIT
C1F9 B5 CC03 DVTST2  BIT A DVSTAT    SECTOR PULSE=1?
C1FC 26 09           BNE   DVTST4    BRANCH IF YES
C1FE 09              DEX             CHECK TIME LIMIT
C1FF 26 F8           BNE   DVTST2
C201 86 FF     DVTST3 LDA A #$FF     DISK MISSING
C203 97 0F           STA A CRNTRK    FLAG DRIVE OFF LINE
C205 43              COM A           THIS CLEARS A AND SETS CARRY
C206 39              RTS
C207 5A        DVTST4 DEC B          END OF TEST?
C208 26 E5           BNE   DVTST1    BRANCH IF NOT
C20A 0C              CLC             DRIVE IS READY
C20B 39              RTS

                     * LOCATE DESIRED SECTOR AND MASK INTERRUPT
C20C CE 6FFF   GTSEC LDX   #$6FFF    SET TIME LIMIT (1 SEC)
C20F 20 05           BRA   GT2
C211 32        GT1   PUL A           RESTORE CCR
C212 06              TAP
C213 09              DEX             DISK MISSING TIME OUT
C214 27 EB           BEQ   DVTST3    DISK MISSING
C216 07        GT2   TPA             SET INTERRUPT MASK
C217 36              PSH A
C218 01              NOP
C219 0F              SEI
C21A 86 10           LDA A #SECBIT   IS ONE-SHOT OFF?
C21C B5 CC03         BIT A DVSTAT
C21F 27 F0           BEQ   GT1       BRANCH IF NOT
C221 96 02           LDA A SCTR      GET DESIRED SECTOR
C223 4A              DEC A           SET TO N-1
C224 2A 02           BPL   GT3
C226 86 09           LDA A #9        FORCE SECTOR 9
C228 F6 CC02 GT3     LDA B SECTOR    ARE WE IN SECTOR N-1?
C22B C4 0F           AND B #$0F
C22D 11              CBA
C22E 26 E1           BNE   GT1       BRANCH IF NOT
C230 86 10           LDA A #SECBIT   IS ONE-SHOT STILL ON?
C232 B5 CC03         BIT A DVSTAT
C235 27 DA           BEQ   GT1       BRANCH IF NOT
C237 B5 CC03 GT4     BIT A DVSTAT    ARE WE IN SECTOR N?
```

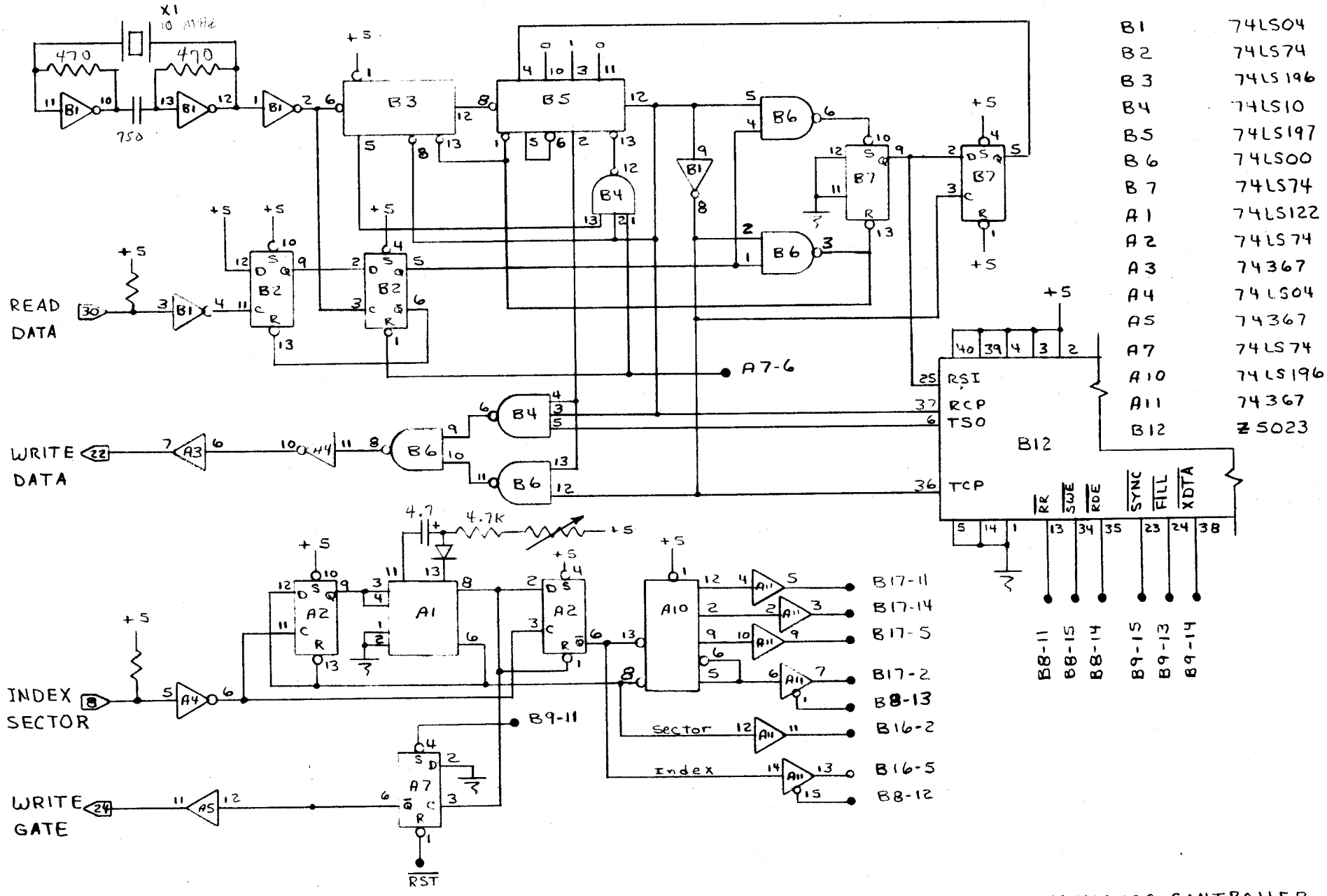
```
C23A 26 FB            BNE   GT4         LOOP UNTIL WE ARE
C23C 32               PUL A
C23D 0C               CLC
C23E 39               RTS

                      * LOCATE DESIRED TRACK
C23F D6 01     SEEKX  LDA B TRK
C241 C4 3F            AND B #TRKMSK
C243 D1 0F            CMP B CRNTRK
C245 27 1F            BEQ   S2
C247 23 0B            BLS   STPOUT
C249 8D 34     STPIN  BSR   TKIN
C24B 7C 000F          INC   CRNTRK
C24E D1 0F            CMP B CRNTRK
C250 26 F7            BNE   STPIN
C252 20 09            BRA   SETTLE
C254 8D 25     STPOUT BSR   TKOT
C256 7A 000F          DEC   CRNTRK
C259 D1 0F            CMP B CRNTRK
C25B 26 F7            BNE   STPOUT
C25D DF 18     SETTLE STX   XTEMP
C25F CE 001E          LDX   #30          30 MSEC DLY
C262 8D 44            BSR   DELAY
C264 DE 18            LDX   XTEMP
C266 39        S2     RTS

                      *GO TO TRK 0 (HOME-RESTORE HEAD)
C267 8D 16     GTKOX  BSR   TKIN
C269 8D 14            BSR   TKIN
C26B 8D 12            BSR   TKIN
C26D 8D 0C     GO     BSR   TKOT
C26F B6 CC03          LDA A DVSTAT
C272 46               ROR A
C273 46               ROR A
C274 25 F7            BCS   GO
C276 7F 000F          CLR   CRNTRK
C279 20 E2            BRA   SETTLE
C27B 8D 1F     TKOT   BSR   RDRV
C27D 20 04            BRA   T1
C27F 8D 1B     TKIN   BSR   RDRV
C281 8A 10            ORA A #$10         SET DIR BIT
C283 B7 CC03   T1     STA A DVSTAT
C286 DF 18            STX   XTEMP
C288 8D 1B            BSR   DEL
C28A 8A 20            ORA A #$20         SET STP BIT
C28C B7 CC03          STA A DVSTAT
C28F 01               NOP                TIME DELAY
C290 01               NOP
C291 01               NOP
C292 84 DF            AND A #$DF         RESET STP BIT
C294 B7 CC03          STA A DVSTAT
C297 8D 0C            BSR   DEL
C299 DE 18            LDX   XTEMP
C29B 39               RTS
C29C 7D CC05   RDRV   TST   MOTON        RETRIGGER MOTOR
C29F B6 CC03          LDA A DVSTAT
```

```
C2A2 84 CF            AND A #$CF       GET DRV
C2A4 39               RTS

C2A5 CE 0014  DEL     LDX   #20         20 MSEC DELAY
              *DELAY ONE MSEC PER INC OF X REG
C2A8 36       DELAY   PSH A
C2A9 86 94    DLY1    LDA A #148       ADJUST FOR CPU CLOCK
C2AB 4A       DLY2    DEC A
C2AC 26 FD            BNE   DLY2
C2AE 09               DEX
C2AF 26 F8            BNE   DLY1
C2B1 32               PUL A
C2B2 39               RTS

              *CALCULATE FWD LNK
C2B3 96 06    FWDCLX  LDA A FWDLNK+1   GET SCTR
C2B5 81 08            CMP A #8
C2B7 27 0C            BEQ   F2         BRANCH IF SCTR 8
C2B9 22 04            BHI   F1         BR IF SCTR 9
C2BB 8B 02            ADD A #2
C2BD 20 08            BRA   F3
C2BF 7C 0005  F1      INC   FWDLNK     INC FWD TRK
C2C2 4F               CLR A
C2C3 20 02            BRA   F3
C2C5 86 01    F2      LDA A #1
C2C7 97 06    F3      STA A FWDLNK+1   UPDATE FWD SCTR
C2C9 0C       F4      CLC
C2CA 39       F5      RTS

              *LOAD A MEMORY IMAGE FILE
C2CB BD C034  LOADX   JSR   RDSECX     LOAD A FILE
C2CE DF 16            STX   TW
C2D0 25 F8            BCS   F5
C2D2 DE 05            LDX   FWDLNK     CK FOR LST BLK
C2D4 27 F4            BEQ   F5         (CARRY IS CLEAR)
C2D6 DF 01            STX   TRKSEC     SETUP NEXT TRACK AND SECTOR
C2D8 DE 16            LDX   TW         POTENTIAL DINOSAUR
C2DA 20 EF            BRA   LOADX

              *SAVE A MEMORY IMAGE FILE
C2DC CE 0000  SAVEX   LDX   #0         CLEAR BACK LINK
C2DF DF 03            STX   BAKLNK     BK LNK
C2E1 DF 0D            STX   POSTAM     CLEAR POSTAMBLE
C2E3 DE 01            LDX   TRKSEC
C2E5 DF 05            STX   FWDLNK
C2E7 8D CA    W3      BSR   FWDCLX     CALC FWD LNK
C2E9 DE 14            LDX   TA
C2EB DF 08            STX   ADDRES     TARGET ADDRESS
C2ED BD C3DC          JSR   LNTH       CALCULATE BLK LNTH
C2F0 BD C115  W4      JSR   WTSECX     WRITE SECTOR
C2F3 25 D5            BCS   F5         BRANCH IF ERROR
C2F5 DE 14            LDX   TA
C2F7 09               DEX
C2F8 9C 1E            CPX   ENDA
C2FA 27 CD            BEQ   F4         BRANCH IF YES
C2FC DE 01            LDX   TRKSEC     SETUP BACK LINK
```

```
C2FE DF 03              STX   BAKLNK
C300 DE 05              LDX   FWDLNK     SETUP NXT TRK/SEC
C302 DF 01              STX   TRKSEC
C304 20 E1              BRA   W3

   (C306)               ORG   BASE+$306
C306 8D 55     SAV      BSR   IN4HS      GET BEG ADD
C308 DF 14              STX   TA         INIT CONT ADD
C30A 8D 51              BSR   IN4HS      GET END ADD
C30C DF 1E              STX   ENDA
C30E 8D 4D              BSR   IN4HS      GET EXEC ADD
C310 DF 1A              STX   EXEC
C312 8D 55              BSR   INDTS      GET DRIVE & SECTOR
C314 25 3D              BCS   TYPERX
C316 7F 000A            CLR   FILTYP     INIT HEADER
C319 8D 48              BSR   CRLF
C31B 8D BF              BSR   SAVEX
                  * REPORT LAST SECTOR USED
C31D 25 34              BCS   TYPERX
C31F CE C3F7   RPTSEX   LDX   #LSTSEC
C322 8D 42              BSR   PD
C324 D6 01              LDA B TRK        GET TRK
C326 C4 3F              AND B #TRKMSK    MASK OFF DRV
C328 4F                 CLR A
C329 20 01              BRA   LS1        CONVERT BINARY TO DEC
C32B 4C        LS2      INC A
C32C C0 0A     LS1      SUB B #10
C32E 2A FB              BPL   LS2
C330 CB 0A              ADD B #10
C332 BD E06B            JSR   OUTHR
C335 17                 TBA
C336 BD E06B            JSR   OUTHR
C339 96 02              LDA A SCTR
C33B 44                 LSR A            CONVERT ALT SCT TO SEQ SEC
C33C 24 02              BCC   LS3
C33E 8B 05              ADD A #5
C340 BD E06B   LS3      JSR   OUTHR
C343 39        LS4      RTS

C344 2A        ER       FCC   /***ERROR /
C345 2A 2A
C347 45 52
C349 52 4F
C34B 52 20
C34D 04                 FCB   4
C34E 0D        MCRLF    FCB   $0D,$0A,0,0,4
C34F 0A 00
C351 00 04

                  *TYPE OUT ERROR MESSAGES
C353 36        TYPERX   PSH A
C354 CE C344            LDX   #ER
C357 8D 0D              BSR   PD
C359 32                 PUL A
C35A 7E E06B            JMP   OUTHR
```

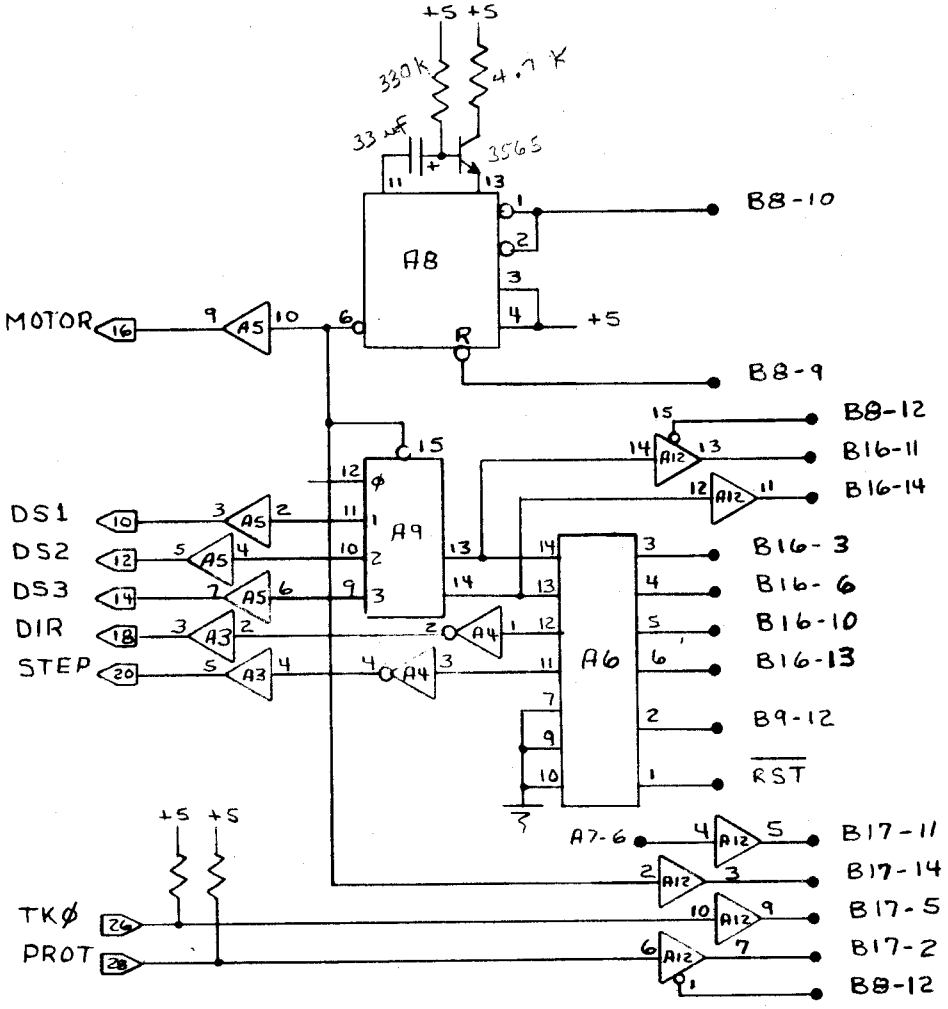
```
C35D BD E0CC  IN4HS  JSR    OUTS
C360 7E E047         JMP    BADDR

                     * OUTPUT CR-LF
C363 CE C34E  CRLF   LDX    #MCRLF     POINT TO CRLF MSSG
C366 7E E07E  PD     JMP    PDATA1

                     *INPUT & CONVERT DTS
C369 8D F2    INDTS  BSR    IN4HS
C36B DF 01    CVTDTS STX    TRKSEC     SAVE DSSS IN TRK-SEC TEMPORARILY
C36D 96 02           LDA A  SCTR       GET LS SEC DIG
C36F 16              TAB              SAVE COPY
C370 C4 0F           AND B  #$F       ISOLATE REAL SECTOR
C372 C1 09           CMP B  #9        CK FOR INVALID SECTOR
C374 22 25           BHI    INVSEC
                     *CONVERT SEQ SCTR TO ALT SCTR
C376 C1 04           CMP B  #4        LESS THAN 4?
C378 23 06           BLS    W1        DO IT DIFFERENTLY
C37A C0 05           SUB B  #5
C37C 58              ASL B            MULT BY 2
C37D 5C              INC B            ADD 1
C37E 20 01           BRA    W2
C380 58       W1     ASL B            MULT BY 2
C381 D7 02    W2     STA B  SCTR       STORE SECTOR
C383 44              LSR A            GET NEXT DIGIT
C384 44              LSR A            INTO POSITION
C385 44              LSR A
C386 44              LSR A
C387 81 09           CMP A  #9        CK FOR INVALID NUMBER
C389 22 10           BHI    INVSEC
C38B D6 01           LDA B  TRK       GET DRV & MS SCTR DIG
C38D 58              ASL B            DEC TO BIN CONVERT
C38E 58              ASL B            4B+A+4B+2B=10B+A
C38F D7 00           STA B  DRV       STORE DRIVE NUMBER
C391 C4 3C           AND B  #$3C
C393 1B              ABA
C394 1B              ABA
C395 54              LSR B
C396 1B              ABA
C397 97 01           STA A  TRK
C399 0C              CLC
C39A 39              RTS
C39B 86 02    INVSEC LDA A  #2         INVALID SCTR MSG
C39D 0D              SEC
C39E 39              RTS

                     * MINIDOS COMMAND PROCESSOR
C39F 8E A07F  MINDOS LDS    #STACK
C3A2 B6 C400         LDA A  $C400      CHK FOR ROM EXTENSION
C3A5 81 7E           CMP A  #$7E
C3A7 26 03           BNE    MINO
C3A9 BD C400         JSR    $C400      JUMP TO IT
C3AC BD C027  MINO   JSR    INITRK     INITIALIZE TRACK REGISTERS
C3AF 8D 03           BSR    MIN2
C3B1 7E E0E3  MIN1   JMP    MONITR
C3B4 BD E1AC  MIN2   JSR    INEEE      INPUT COMMAND
```

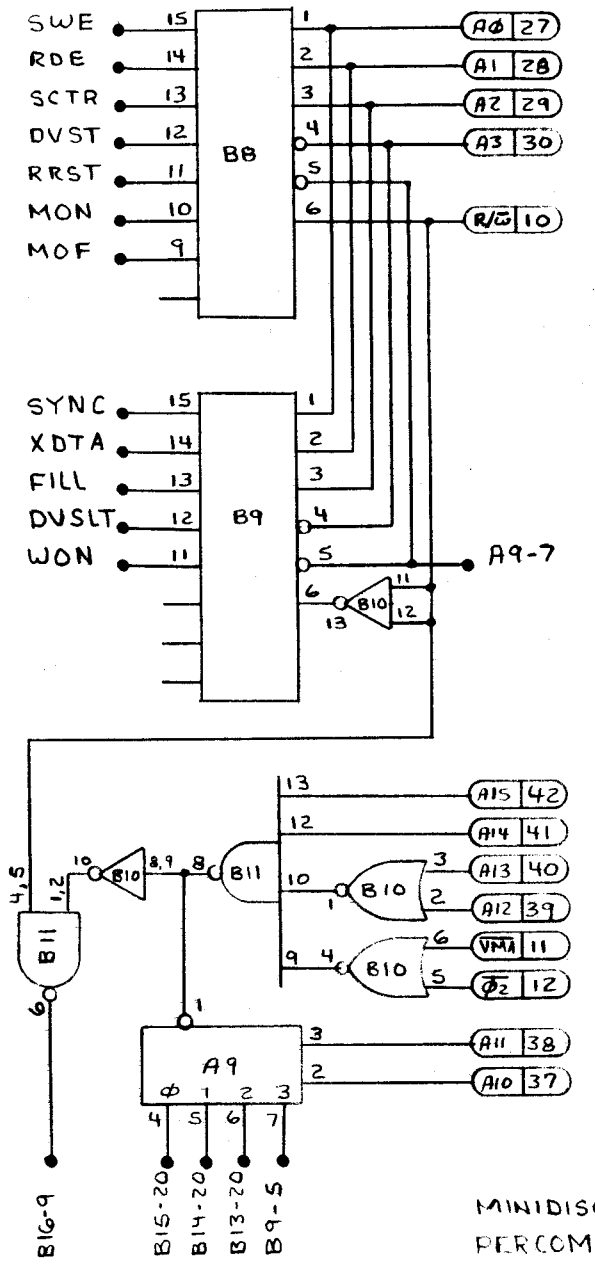
```
C3B7 81 4C            CMP A #'L
C3B9 27 07            BEQ   LOD
C3BB 81 53            CMP A #'S
C3BD 26 F2            BNE   MIN1
C3BF 7E C306          JMP   SAV
C3C2 8D A5     LOD    BSR   INDTS      GET DISK LOCATION
C3C4 25 8D            BCS   TYPERX
C3C6 8D 95            BSR   IN4HS      GET TARGET ADDRESS
C3C8 DF 16            STX   TW
C3CA 8D 97            BSR   CRLF
C3CC BD C2CB          JSR   LOADX      LOAD BINARY FILE
C3CF 25 82            BCS   TYPERX
C3D1 DE 0D            LDX   POSTAM
C3D3 8C FFFF          CPX   #$FFFF
C3D6 27 03            BEQ   MIN3
C3D8 FF A048          STX   XFER
C3DB 39        MIN3   RTS

               *CALCULATE BLOCK LENGTH
C3DC 96 1F     LNTH   LDA A ENDA+1
C3DE 90 09            SUB A ADDRES+1   TARGET ADDRESS LSB
C3E0 D6 1E            LDA B ENDA
C3E2 D2 08            SBC B ADDRES     TARGET ADDRESS MSB
C3E4 27 04            BEQ   L1         LAST BLK IF B=0
C3E6 4F               CLR A
C3E7 97 07            STA A BYTCNT     BYTE COUNT
C3E9 39               RTS
C3EA 4C        L1     INC A
C3EB 97 07            STA A BYTCNT
C3ED 4F               CLR A            CLEAR FWD LNK
C3EE 97 05            STA A FWDLNK
C3F0 97 06            STA A FWDLNK+1
C3F2 DE 1A            LDX   EXEC       GET EXECTION ADDRESS
C3F4 DF 0D            STX   POSTAM     SINCE THIS IS LAST BLK
C3F6 39               RTS

C3F7 4C        LSTSEC FCC   /LST SEC=/ LAST SECTOR MESSAGE
C3F8 53 54
C3FA 20 53
C3FC 45 43
C3FE 3D
C3FF 04               FCB   4
                      END
      00  ERROR(S) DETECTED
```



B1	74LS04
B2	74LS74
B3	74LS196
B4	74LS10
B5	74LS197
B6	74LS00
B7	74LS74
A1	74LS122
A2	74LS74
A3	74367
A4	74LS04
A5	74367
A7	74LS74
A10	74LS196
A11	74367
B12	Z5023

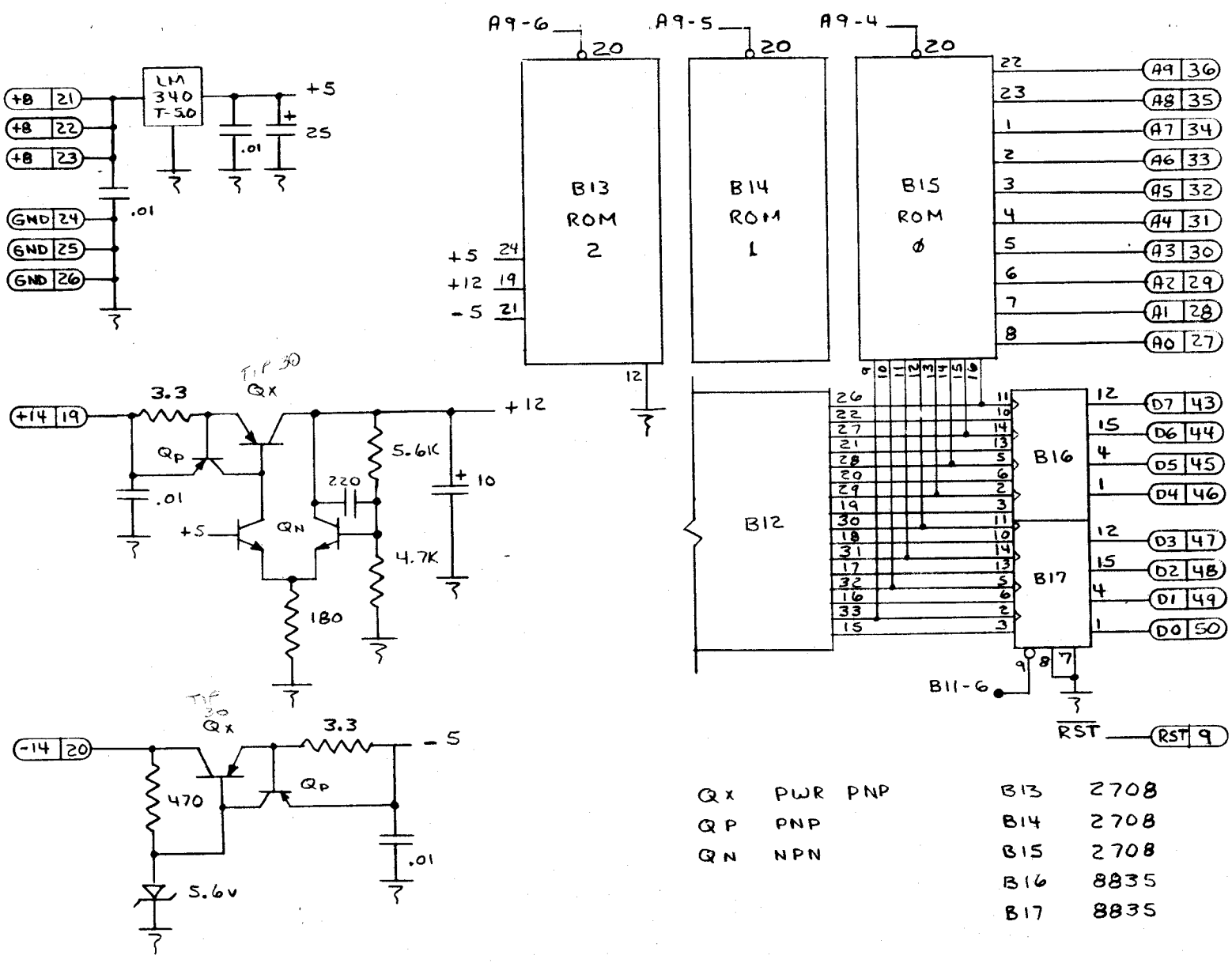
MINI DISC CONTROLLER
PERCOM DATA CO.
COPYRIGHT 1977
Sheet 1 of 3
REV D 12-8-78 HM





A4	74LS04
A5	74 367
A6	74LS161
A8	74LS122
A9	74LS139
A12	74 367
B8	74LS138
B9	74LS138
B10	74LS02
B11	74LS20

MINIDISC CONTROLLER
PERCOM DATA CO.
COPYRIGHT 1977
SHEET 2 of 3
REV D 12-8-78 HM



Qx	PWR PNP	B13	2708
Qp	PNP	B14	2708
QN	NPN	B15	2708
		B16	8835
		B17	8835

MINI DISC CONTROLLER
PERCOM DATA CO.

SHEET 3 of 3
REV D 10-10-77 HM
5-24-78 HM

PERCOM DATA CO.

TECHNICAL MEMO

TM-LFD-400-01
LFD-400 FLOPPY DISK SYSTEM
JANUARY 3, 1978
REVISED APRIL 29,1978 1.1

SUBJECT: INTERFACING THE PERCOM LFD-400 FLOPPY DISK WITH SWTPC 8K BASIC VERSION 2.0 (ALSO 2.2)

THE ENCLOSED LISTING IS A SOFTWARE PATCH FOR SWTPC 8K BASIC VERSION 2.0 TO PERMIT USER PROGRAMS TO BE SAVED TO AND LOADED FROM THE PERCOM DISK. THE PATCH RETAINS THE NORMAL CASSETTE SAVE AND LOAD FUNCTIONS. IT ALSO INCLUDES A "TAIL END" APPEND WHICH SHOULD PROVE USEFUL FOR APPENDING DATA STATEMENTS TO AN EXISTING PROGRAM.

TO IMPLEMENT THE PATCH, FIRST LOAD 8K BASIC IN ANY MANNER AVAILABLE TO YOU (CASSETTE, PAPER TAPE, DISC). EXAMINE MEMORY LOCATION $014E. IF YOUR VERSION OF BASIC IS THE SAME AS OURS, IT SHOULD CONTAIN 1E AND THE NEXT LOCATION ($014F) SHOULD CONTAIN AF.* THE PATCH IS SHORT ENOUGH THAT IT MAY BE ENTERED BY HAND USING THE LISTING. THE ADDED CODE BEGINS AT $1EF0 AND RUNS TO $1F6F . BE SURE TO PICK UP THE PATCHES AT ADDRESSES $014E, $02EF, $02F6, AND $02FF.

AFTER MAKING THE PATCHES, YOU WILL WANT TO MAKE A COPY OF THE MODIFIED PROGRAM. SAVE MEMORY FROM $0100 TO $1F6F. THE PROGRAM START ADDRESS IS $0100.

SAVING A PROGRAM

TO SAVE A PROGRAM YOU HAVE ENTERED IN BASIC, TYPE `SAVE` AND THE DRIVE AND SECTOR TO WHICH YOU WISH THE PROGRAM TO BE SAVED. WHEN THE PROGRAM HAS BEEN SAVED, MINIDOS(TM) WILL REPORT THE LAST SECTOR USED AND WILL RETURN TO BASIC CONTROL (#).

EXAMPLE:

```
#SAVE 1045<CR>          THIS WILL SAVE THE PROGRAM ON DRIVE ONE
LAST SECTOR=047         BEGINNING AT SECTOR 45 ENDING AT SECTOR 47
```

IF AN ERROR OCCURS (DISK MISSING, DISK OVERRUN, ETC) MINIDOS(TM) WILL REPORT THE ERROR...NOTICE...MINIDOS(TM) ERROR REPORTING IS DIFFERENT THAN BASIC ERROR REPORTING AND THE TWO SHOULD NOT BE CONFUSED. MINIDOS(TM) ERROR MESSAGES ARE PRECEDED BY THREE ASTERISKS.

EXAMPLE:

```
#SAVE 1350<CR>
***ERROR 04                 DISK OVERRUN-EXCEEDED 350 SECTORS
```

THE PROCEDURE TO SAVE A PROGRAM TO CASSETTE REMAINS THE SAME AS BEFORE THIS PATCH WAS ADDED.

*8K BASIC VERSION 2.2 WILL CONTAIN $E2 AT LOCATION $Ø14F.

LOADING A PROGRAM

TO LOAD A PROGRAM SIMPLY TYPE `LOAD` AND THE DRIVE AND SECTOR AT WHICH IT BEGINS. WHEN LOADED CONTROL WILL BE RETURNED TO BASIC.

EXAMPLE:

```
#LOAD 1045<CR>          THIS WILL LOAD THE PROGRAM SAVED EARLIER
                        ON DRIVE ONE SECTOR 45.
```

ERROR REPORTING IS HANDLED IN THE MANNER DESCRIBED EARLIER.

EXAMPLE:

```
#LOAD 1045<CR>
***ERROR 00             DISK MISSING-DISK GATE OPEN
```

THE PROCEDURE TO LOAD A PROGRAM FROM CASSETTE REMAINS THE SAME AS BEFORE THIS PATCH WAS ADDED.

APPENDING TO THE PROGRAM

THIS FUNCTION IS A "TAIL END" APPEND IN THAT IT DOES NOT "SORT IN" THE ADDED LINES. INSTEAD THEY ARE SIMPLY APPENDED TO THE END OF WHATEVER PROGRAM IS ALREADY IN MEMORY...NOTICE...THE LINE NUMBERS OF THE APPENDED LINES MUST BE GREATER THAN THE HIGHEST LINE NUMBER IN THE PROGRAM ALREADY IN MEMORY.

EXAMPLE:

```
#APPEND 1067<CR>        THIS APPENDS THE FILE ON DRIVE ONE:
                        SECTOR 67 TO THE PROGRAM ALREADY IN
                        MEMORY.
```

DATA FILES

AT THE TIME THIS MEMO IS BEING WRITTEN WE ARE WORKING ON A WAY TO IMPLEMENT BASIC DATA FILES IN SWTP 8K BASIC. THIS WILL GREATLY INCREASE THE POWER OF 8K BASIC IN MANY APPLICATIONS. WHEN READY PERCOM WILL ISSUE A SUPPLEMENT TO THIS TECHNICAL MEMO. IN THE MEAN TIME YOU MAY SIMULATE DATA FILES BY APPENDING DATA STATEMENTS TO THE MAIN PROGRAM AS DESCRIBED EARLIER.

SUPPLEMENTAL APRIL 29, 1978

DISK DATA FILES HAVE NOW BEEN IMPLEMENTED ON SWTP 8K BASIC. THE EXTENSION IS DESCRIBED IN TECHNICAL MEMO TM-LFD-400-05. SINCE NOT ALL USERS REQUIRE DISK BASIC DATA FILES AND SINCE PERCOM DATA MUST PAY ROYALTIES TO THE PROGRAMMER WHO WROTE THE EXTENSION, A DISKETTE AND LISTING OF THE EXTENSION ARE AVAILABLE FOR A MODERATE FEE ($15).

PERCOM DATA CO.

TECHNICAL MEMO

TM-LFD-400-01
LFD-400 FLOPPY DISK SYSTEM
JANUARY 3, 1978
REVISED APRIL 29,1978 1.1

SUBJECT: INTERFACING THE PERCOM LFD-400 FLOPPY DISK WITH SWTPC 8K BASIC VERSION 2.0 (ALSO 2.2)

THE ENCLOSED LISTING IS A SOFTWARE PATCH FOR SWTPC 8K BASIC VERSION 2.0 TO PERMIT USER PROGRAMS TO BE SAVED TO AND LOADED FROM THE PERCOM DISK. THE PATCH RETAINS THE NORMAL CASSETTE SAVE AND LOAD FUNCTIONS. IT ALSO INCLUDES A "TAIL END" APPEND WHICH SHOULD PROVE USEFUL FOR APPENDING DATA STATEMENTS TO AN EXISTING PROGRAM.

TO IMPLEMENT THE PATCH, FIRST LOAD 8K BASIC IN ANY MANNER AVAILABLE TO YOU (CASSETTE, PAPER TAPE, DISC). EXAMINE MEMORY LOCATION $014E. IF YOUR VERSION OF BASIC IS THE SAME AS OURS, IT SHOULD CONTAIN 1E AND THE NEXT LOCATION ($014F) SHOULD CONTAIN AF.* THE PATCH IS SHORT ENOUGH THAT IT MAY BE ENTERED BY HAND USING THE LISTING. THE ADDED CODE BEGINS AT $1EF0 AND RUNS TO $1F6F . BE SURE TO PICK UP THE PATCHES AT ADDRESSES $014E, $02EF, $02F6, AND $02FF.

AFTER MAKING THE PATCHES, YOU WILL WANT TO MAKE A COPY OF THE MODIFIED PROGRAM. SAVE MEMORY FROM $0100 TO $1F6F. THE PROGRAM START ADDRESS IS $0100.

SAVING A PROGRAM

TO SAVE A PROGRAM YOU HAVE ENTERED IN BASIC, TYPE `SAVE` AND THE DRIVE AND SECTOR TO WHICH YOU WISH THE PROGRAM TO BE SAVED. WHEN THE PROGRAM HAS BEEN SAVED, MINIDOS(TM) WILL REPORT THE LAST SECTOR USED AND WILL RETURN TO BASIC CONTROL (#).

EXAMPLE:

```
#SAVE 1045<CR>          THIS WILL SAVE THE PROGRAM ON DRIVE ONE
LAST SECTOR=047         BEGINNING AT SECTOR 45 ENDING AT SECTOR 47
```

IF AN ERROR OCCURS (DISK MISSING, DISK OVERRUN, ETC) MINIDOS(TM) WILL REPORT THE ERROR...NOTICE...MINIDOS(TM) ERROR REPORTING IS DIFFERENT THAN BASIC ERROR REPORTING AND THE TWO SHOULD NOT BE CONFUSED. MINIDOS(TM) ERROR MESSAGES ARE PRECEDED BY THREE ASTERISKS.

EXAMPLE:

```
#SAVE 1350<CR>
***ERROR 04                DISK OVERRUN-EXCEEDED 350 SECTORS
```

THE PROCEDURE TO SAVE A PROGRAM TO CASSETTE REMAINS THE SAME AS BEFORE THIS PATCH WAS ADDED.

*8K BASIC VERSION 2.2 WILL CONTAIN $E2 AT LOCATION $Ø14F.

```
                        NAM   BASIC     PATCH 1.2
                      * WRITTEN BY H. A. MAUCH
                      * COPYRIGHT 1978 PERCOM DATA CO.
                      * ALL RIGHTS RESERVED
                      * MODIFIED FOR MINIDOS VER 1.4  9-27-78
                      ****************************************
                      * PATCHES FOR SWTPC 8K BASIC VERSION 2.0 AND 2.2
                      * TO ADAPT IT TO THE PERCOM LFD-400 MINI-DISC
                      * SYSTEM.  REQUIRES THE PERCOM MINIDOS(TM) 1.4
                      *
                      * THIS PATCH IMPLEMENTS SIMPLE DISK `SAVE` AND
                      * `LOAD` FUNCTIONS.  FOR USERS NEEDING A MORE
                      * COMPREHENSIVE RANDOM ACCESS DATA FILE CAP-
                      * ABILITY IN BASIC, THE PERCOM `BASIC BAND-AID`
                      * IS SUGGESTED.
                      ****************************************

                      * PATCH LOCATIONS IN BASIC
   (0CC7)             TSAVE  EQU   $0CC7 D41  ENTRY TO TAPE SAVE ROUTINE
   (0D0E)             TLOAD  EQU   $0D0E D85  ENTRY TO TAPE LOAD ROUTINE
   (0BAB)             BASIC  EQU   $0BAB      ENTRY TO BASIC CONTROL
   (0ABF)             SKPSP  EQU   $0ABF      ROUTINE TO SKIP OVER SPACES
   (063A)             NUM    EQU   $063A      ROUTINE TO IDENTIFY A NUMERIC
   (0B71)             INTFIL EQU   $0B71      ROUTINE TO INITIALIZE FILE
   (0032)             HILINE EQU   $32        LOCATION OF HIGHEST LINE #
   (0034)             LINPTR EQU   $34        LOCATION OF LINE BUFFER POINTER
   (002E)             BOF    EQU   $2E        BEGINNING OF BASIC FILE
   (002A)             EOF    EQU   $2A        END OF BASIC FILE
                      * PATCH LOCATIONS IN MINIDOS(TM) 1.4
   (C319)             SAVEX  EQU   $C319      ENTRY INTO SAVE ROUTINE
   (C3C8)             LOADX  EQU   $C3C8      ENTRY INTO LOAD ROUTINE
   (C36B)             CVTDTS EQU   $C36B      ROUTINE TO CONVERT DRV/SCTR
   (0000)             DSKHDR EQU   0           DISK HEADER
   (0014)             TA     EQU   $14        TEMP ADDRESS STORAGE
   (001A)             EXEC   EQU   $1A        EXECUTION ADDRESS
   (001C)             WRTDTS EQU   $1C        TEMP DRV/SCTR STORAGE
   (001E)             ENDA   EQU   $1E        END POINT ADDRESS

   (1EF0)                    ORG   $1EF0
                      * SAVE A PROGRAM WRITTEN IN BASIC
1EF0 8D 43            SAVE   BSR   GETDTS     GET DRV/SCTR FROM LINE BUFFER
1EF2 24 03                   BCC   S1         BRANCH IF DISK SAVE
1EF4 7E 0CC7                 JMP   TSAVE      JUMP TO TAPE SAVE
1EF7 DE 2E            S1     LDX   BOF        GET BEGINNING OF FILE ADDRESS
1EF9 9C 2A                   CPX   EOF        CHECK FOR EMPTY FILE
1EFB 27 35                   BEQ   JMPBSC     QUIT IF EMPTY
1EFD DF 14                   STX   TA
1EFF DE 2A                   LDX   EOF        GET END OF FILE
1F01 09                      DEX
1F02 DF 1E                   STX   ENDA
1F04 DE 32                   LDX   HILINE     GET HIGH LINE NUMBER
1F06 DF 1A                   STX   EXEC
1F08 86 0B                   LDA A #$B        IDENTIFY AS A BASIC PROGRAM
1F0A 97 0A                   STA A DSKHDR+10
1F0C BD C319                 JSR   SAVEX      GO SAVE TO DISK
```

```
1F0F 20 21          BRA   JMPBSC    JMP TO BASIC CONTROL

                    * LOAD OR APPEND A PROGRAM WRITTEN IN BASIC
1F11 BD 0BD9 LOAD   JSR   INTFIL    CLEAR FILE BUFFER
1F14 8D 1F   APPEND BSR   GETDTS    GET DRV/SCTR FROM LINE BUFFER
1F16 24 03          BCC   L1        BRANCH IF LOAD FROM DISK
1F18 7E 0D0E 85     JMP   TLOAD     JMP IF LOAD FROM TAPE
1F1B DE 2A   L1     LDX   EOF       GET LOAD ADDRESS
1F1D BD C3C8        JSR   LOADX     ACTIVATE DISK AND LOAD
1F20 DE 14          LDX   TA        GET NEXT AVAILABLE ADDRESS
1F22 DF 2A          STX   EOF       SET END OF FILE
1F24 FE A048        LDX   $A048     FIX HILINE NUMBER
1F27 DF 32          STX   HILINE
1F29 CE 0103        LDX   #$0103    RESTORE WARM START ADDRESS
1F2C FF A048        STX   $A048
1F2F 01             NOP
1F30 01             NOP
1F31 01             NOP
1F32 7E 0BAB JMPBSC JMP   BASIC     JMP TO BASIC CONTROL


1F35 DE 34   GETDTS LDX   LINPTR
1F37 BD 0ABF        JSR   SKPSP
1F3A BD 063A        JSR   NUM
1F3D 25 13          BCS   G1        BR IF NOT NUMERIC
1F3F 8D 12          BSR   GET2
1F41 25 0F          BCS   G1
1F43 97 1C          STA A WRTDTS
1F45 8D 0C          BSR   GET2
1F47 25 09          BCS   G1
1F49 97 1D          STA A WRTDTS+1
1F4B DF 34          STX   LINPTR
1F4D DE 1C          LDX   WRTDTS
1F4F BD C36B        JSR   CVTDTS    CONVERT DTS
1F52 39      G1     RTS

1F53 8D 0D   GET2   BSR   GETN
1F55 25 0A          BCS   G2
1F57 48             ASL A
1F58 48             ASL A
1F59 48             ASL A
1F5A 48             ASL A
1F5B 16             TAB
1F5C 8D 04          BSR   GETN
1F5E 25 01          BCS   G2
1F60 1B             ABA
1F61 39      G2     RTS

1F62 A6 00   GETN   LDA A 0,X
1F64 BD 063A        JSR   NUM
1F67 25 04          BCS   G3
1F69 84 0F          AND A #$0F      REMOVE ASCII OFFSET
1F6B 08             INX
1F6C 0C             CLC
1F6D 39      G3     RTS
   (1F6E)   PATEND EQU   *          END OF PATCH
```

```
   (014E)               ORG    $14E
014E 1F 6E              FDB    PATEND

   (02F6) 0311          ORG    $2F6     0D41
02F6 1E F0              FDB    SAVE

   (02EF) 030A          ORG    $2EF
02EF 1F 11              FDB    LOAD     0D85

   (02FF) 0319          ORG    $2FF
02FF 1F 14              FDB    APPEND   0D88
                        END
      00  ERROR(S) DETECTED
```

SYMBOL TABLE:

APPEND	1F14	BASIC	0BAB	BOF	002E	CVTDTS	C36B
DSKHDR	0000	ENDA	001E	EOF	002A	EXEC	001A
G1	1F52	G2	1F61	G3	1F6D	GET2	1F53
GETDTS	1F35	GETN	1F62	HILINE	0032	INTFIL	0B71
JMPBSC	1F32	L1	1F1B	LINPTR	0034	LOAD	1F11
LOADX	C3C8	NUM	063A	PATEND	1F6E	S1	1EF7
SAVE	1EF0	SAVEX	C319	SKPSP	0ABF	TA	0014
TLOAD	0D0E	TSAVE	0CC7	WRTDTS	001C		

PERCOM DATA CO.

TECHNICAL MEMO

TM-LFD-400-02
LFD-400 FLOPPY DISK SYSTEM
MARCH 31, 1978 REV 5/30/78

SUBJECT: INTERFACING THE PERCOM LFD-400 FLOPPY DISK WITH THE TSC (TECHNICAL SYSTEMS CONSULTANTS) TEXT EDITOR.

THE TSC TEXT EDITOR IS A VERY POWERFUL EDITOR FOR THE 6800. IT IS EASY TO LEARN AND USE AND THE PRICE IS RIGHT. THE ENCLOSED LISTING IS A SOFTWARE PATCH FOR THE TSC EDITOR WHICH PERMITS THE TEXT FILES CREATED USING THE EDITOR TO BE SAVED TO AND LOADED FROM THE PERCOM DISK. THE DISK `SAVE`, `READ`, AND `WRITE` FUNCTIONS REPLACE THE TAPE FUNCTIONS. OPERATION OF THESE FUNCTIONS IS THE SAME AS BEFORE WITH THE ADDITION OF A DISK TARGET SPECIFICATION.

TO IMPLEMENT THIS PATCH, FIRST LOAD THE TSC EDITOR IN ANY MANNER AVAILABLE TO YOU. TSC SOFTWARE IS AVAILABLE ON PAPER TAPE, KC STANDARD CASSETTE, AND COMPLETE LISTING. THE KC STANDARD CASSETTES CAN BE LOADED WITH THE PERCOM CIS-30 CASSETTE INTERFACE. THE PATCH IS SHORT ENOUGH THAT IT MAY BE EASILY ENTERED BY HAND USING THE ATTACHED LISTING. THE MODIFIED CODE BEGINS AT ADDRESS $13D3 AND RUNS TO $145F. BE SURE TO PICK UP THE PATCHES AT $139D, $0358, $02D9, AND $0272. AFTER MAKING THE PATCHES, YOU WILL WANT TO MAKE A COPY OF THE MODIFIED PROGRAM. SAVE MEMORY FROM $00B0 THROUGH $145F. THE PROGRAM START ADDRESS IS $0200.

SAVING A TEXT FILE

TO SAVE A FILE CREATED USING THE TSC EDITOR, TYPE `SAVE` AND THE DRIVE AND SECTOR TO WHICH YOU WISH THE FILE TO BE SAVED. WHEN THE FILE HAS BEEN SAVED, MINIDOS(TM) WILL REPORT THE LAST SECTOR USED AND WILL RETURN TO THE EDITOR CONTROL (#).

EXAMPLE:

```
#SAVE 1045<CR>      THIS WILL SAVE THE FILE ON DRIVE ONE BEGINNING
LAST SECTOR=068     AT SECTOR 45 ENDING AT SECTOR 68
```

THE WRITE COMMAND WORKS THE SAME WAY. SIMPLY APPEND A DISK TARGET TO THE COMMAND STRING.

READING A FILE

TO LOAD A FILE PREVIOUSLY CREATED BY THE EDITOR SIMPLY APPEND THE DISK TARGET OF THE BEGINNING OF THE FILE TO THE `READ` COMMAND.

EXAMPLE:

```
#READ 1045<CR>      THIS WILL LOAD THE FILE SAVED EARLIER ON
                    DRIVE ONE SECTOR 45
```

IN ALL INSTANCES ERRORS WHICH MAY OCCUR WHEN ACCESSING THE DISK ARE REPORTED BY MINIDOS(TM) AND CONTROL IS RETURNED TO THE EDITOR.

EXAMPLE:

```
#SAVE 1350<CR>
***ERROR 4          DISK OVERRUN-EXCEEDED 350 SECTORS
```

DATA SAVED ON DISK IS A BINARY MEMORY IMAGE OF THE EDITOR BUFFER. THIS MUST BE CONSIDERED IF THE DISK FILE WILL BE USED BY SOME OTHER SOFTWARE SUCH AS AN ASSEMBLER OR A TEXT PROCESSOR. PERCOM TECHNICAL MEMO TM-LFD-400-03 DESCRIBES MODIFICATIONS TO THE TSC ASSEMBLER WHICH PERMITS SOURCE FILES CREATED BY THE MODIFIED TSC EDITOR AND STORED ON DISK TO BE ASSEMBLED.

FINAL NOTE: THE TAPE `SAVE`, `LOAD`, AND `GAP` FUNCTIONS HAVE BEEN ELIMINATED. THE MODIFIED TSC EDITOR WAS USED TO PREPARE THIS TECHNICAL MEMO. THE RESULTING TEXT WAS SAVED ON DISK FOR FUTURE REVISION.

THIS SOFTWARE PATCH IN NOW AVAILABLE ON DISKETTE ALONG WITH PATCHES FOR THE TSC ASSEMBLER, SWTP 8K BASIC AND OTHER USEFUL ROUTINES.

TSC SOFTWARE IS AVAILABLE FROM MANY LOCAL COMPUTER STORES. THE ADDRESS IF YOU ARE UNABLE TO OBTAIN THE SOFTWARE LOCALLY IS:

TECHNICAL SYSTEMS CONSULTANTS
P.O. BOX 2574
WEST LAFAYETTE, IN 47906
(317) 742-7509

```
                        NAM   TSC       EDITOR PATCH 1.2
                      * WRITTEN BY H.A. MAUCH
                      * COPYRIGHT 1978 PERCOM DATA CO.
                      * ALL RIGHTS RESERVED
                      * MODIFIED FOR MINIDOS VER 1.4  9-27-78
                      ****************************************
                      * PATCHES FOR TSC EDITOR TO ADAPT IT TO
                      * THE PERCOM LFD-400 MINI-DISC SYSTEM.
                      * WRITTEN FOR PERCOM MINIDOS(TM) 1.4
                      ****************************************

                      * LINK POINTS TO TSC EDITOR
   (0040)             TEMP   EQU   $40
   (0044)             BUFPNT EQU   $44
   (0058)             SPCPT1 EQU   $58
   (005A)             SPCPT2 EQU   $5A
   (0083)             CHKFLG EQU   $83
   (0090)             NUMBER EQU   $90
   (0097)             FILBEG EQU   $97
   (0099)             FILEND EQU   $99
   (0212)             MEMEND EQU   $0212
   (0441)             ERROR  EQU   $0441
   (0483)             PSTRNG EQU   $0483
   (0492)             SKIPSP EQU   $0492
   (0663)             TSTEND EQU   $0663
   (0698)             GETNUM EQU   $0698
   (06B0)             RENUM2 EQU   $06B0
   (0755)             BCDCON EQU   $0755
   (07A3)             CLRNUM EQU   $07A3
   (07F0)             BAKONE EQU   $07F0
   (07F6)             BAKON2 EQU   $07F6
   (0990)             BOTTO1 EQU   $0990
                      * LINK POINTS TO MINIDOS(TM) 1.2
   (0000)             WRTHDR EQU   0
   (0014)             TA     EQU   $14
   (001A)             EXEC   EQU   $1A
   (001E)             ENDA   EQU   $1E
   (C319)             SAVEX  EQU   $C319
   (C3C8)             LOADX  EQU   $C3C8
   (C363)             CRLF   EQU   $C363
   (C36B)             CVTDTS EQU   $C36B

   (139D)                    ORG   $139D
139D 01                      NOP                REMOVE CR-EOL TEST
139E 01                      NOP
139F 01                      NOP
   (13D3)                    ORG   $13D3
                      * SAVE TEXT FILE TO DISK
13D3 BD C363       RECORD    JSR   CRLF
13D6 8D 21                   BSR   GETDTS       GET DRV/SCT FROM LINE BUFFER
13D8 25 32                   BCS   ERR
13DA 86 0E                   LDA A #$0E         IDENTIFY AS EDITOR TEXT
13DC 97 0A                   STA A WRTHDR+10
13DE DE 58                   LDX   SPCPT1       BEGINNING OF FILE ADDRESS
13E0 DF 14                   STX   TA
```

```
13E2 DE 5A          LDX   SPCPT2    END OF FILE ADDRESS
13E4 9C 97          CPX   FILBEG    CHECK FOR EMPTY FILE
13E6 26 06          BNE   SAV1
13E8 CE 1454        LDX   #NOFILE
13EB 7E 0483        JMP   PSTRNG
13EE 09       SAV1  DEX
13EF DF 1E          STX   ENDA
13F1 CE FFFF        LDX   #$FFFF    KILL EXECUTION ADDRESS
13F4 DF 1A          STX   EXEC
13F6 7E C319        JMP   SAVEX     GO SAVE TO DISK

13F9 DE 44    GETDTS LDX  BUFPNT    GET LINE BUFFER POINTER
13FB BD 0492        JSR   SKIPSP    SKIP SPACES
13FE BD 0663        JSR   TSTEND    IS IT CR OR EOL?
1401 27 15          BEQ   GET2      BRANCH IF YES
1403 BD 0755        JSR   BCDCON    GET DRV/SEC NUMBER
1406 86 F0          LDA A #$F0      CHECK FOR VALID DRV NUMBER
1408 95 90          BIT A NUMBER
140A 26 03          BNE   GET1      BRANCH IF VALID DRIVE
140C 7E 0441  ERR   JMP   ERROR
140F DF 44    GET1  STX   BUFPNT    SAVE BUFFER POINTER
1411 DE 90          LDX   NUMBER
1413 BD C36B        JSR   CVTDTS    CONVERT DRV/SEC
1416 0C             CLC
1417 39             RTS
1418 0D       GET2  SEC
1419 39             RTS

                    * READ TEXT FILE FROM DISK
141A 8D DD    READ  BSR   GETDTS
141C 25 EE          BCS   ERR
141E BD 07A3        JSR   CLRNUM
1421 DE 99          LDX   FILEND
1423 DF 40          STX   TEMP
1425 9C 97          CPX   FILBEG
1427 27 06          BEQ   READ1
1429 BD 07F0        JSR   BAKONE
142C BD 0698        JSR   GETNUM
142F DE 99    READ1 LDX   FILEND
1431 BD C3C8        JSR   LOADX
1434 DE 14          LDX   TA
1436 5F       READ4 CLR B
1437 BD 07F6        JSR   BAKON2
143A DF 99    READ5 STX   FILEND
143C 7C 0083        INC   CHKFLG
143F 9C 40          CPX   TEMP
1441 27 05          BEQ   READ6
1443 DE 40          LDX   TEMP
1445 BD 06B0        JSR   RENUM2
1448 7E 0990  READ6 JMP   BOTTO1
144B BC 0212  STORE CPX   MEMEND
144E 27 03          BEQ   STOR1
1450 A7 00          STA A 0,X
1452 08             INX
1453 39       STOR1 RTS
1454 46       NOFILE FCC  'FILE EMPTY'
```

```
1455 49 4C
1457 45 20
1459 45 4D
145B 50 54
145D 59
145E 04                 FCB    4

   (145F)       BEGPNT  EQU    *
   (0358)               ORG    $0358
0358 CE 145F            LDX    #BEGPNT    CHANGE BEGINNING POINT

   (02D9)               ORG    $02D9
02D9 14 1A              FDB    READ CHANGE READ VECTOR

   (0272)               ORG    $0272
0272 04 41              FDB    ERROR CHANGE GAP VECTOR
                        END
      00  ERROR(S) DETECTED
```

SYMBOL TABLE:

BAKON2	07F6	BAKONE	07F0	BCDCON	0755	BEGPNT	145F
BOTTO1	0990	BUFPNT	0044	CHKFLG	0083	CLRNUM	07A3
CRLF	C363	CVTDTS	C36B	ENDA	001E	ERR	140C
ERROR	0441	EXEC	001A	FILBEG	0097	FILEND	0099
GET1	140F	GET2	1418	GETDTS	13F9	GETNUM	0698
LOADX	C3C8	MEMEND	0212	NOFILE	1454	NUMBER	0090
PSTRNG	0483	READ	141A	READ1	142F	READ4	1436
READ5	143A	READ6	1448	RECORD	13D3	RENUM2	0680
SAV1	13EE	SAVEX	C319	SKIPSP	0492	SPCPT1	0058
SPCPT2	005A	STOR1	1453	STORE	144B	TA	0014
TEMP	0040	TSTEND	0663	WRTHDR	0000		

PERCOM DATA CO.

TECHNICAL MEMO

TM-LFD-400-03
LFD-400 FLOPPY DISK SYSTEM
REVISED SEPTEMBER 30, 1978
FOR MINIDOS VER 1.4

SUBJECT: ADAPTING THE TSC 6800 MNEMONIC ASSEMBLER FOR USE WITH THE PERCOM LFD-400 FLOPPY DISK SYSTEM.

THE ATTACHED LISTING IS A SOFTWARE PATCH FOR THE TSC (TECHNICAL SYSTEMS CONSULTANTS) 6800 ASSEMBLER TO PERMIT PROGRAM SOURCE FILES TO BE READ FROM DISK AND ASSEMBLED. ANY RESULTING OBJECT CODE MAY ALSO BE STORED ON THE DISK. THE TSC ASSEMBLER HAS BEEN MODIFIED TO PERMIT THE ASSEMBLY OF SOURCE FILES WHICH ARE LARGER THAN THE AVAILABLE MEMORY. THE TSC ASSEMBLER WAS CHOSEN FOR THIS MODIFICATION SIMPLY BECAUSE A SOURCE LISTING IS READILY AVAILABLE.

IN ITS ORIGINAL FORM, THE TSC ASSEMBLER REQUIRED THE USER TO SET A NUMBER OF 'ASSEMBLER DATA POINTERS' (TSC ASSEMBLER MANUAL PAGES 12-14). THIS REQUIREMENT HAS BEEN ELIMINATED UNLESS YOU WISH TO INCREASE THE SIZE OF THE AVAILABLE LABEL STORAGE AREA. MORE ABOUT THIS LATER.

WE HAVE ELIMINATED THE MEMORY OPTION (MEM) SINCE THIS IS OF LITTLE VALUE IN DISK BASED SYSTEMS AND HAVE USED THE MEMORY SPACE FOR THE DISK I/O ROUTINES. THE TAPE OPTION (TAP-NOT) NOW CONTROLS THE GENERATION OF AN OBJECT FILE ON THE DISK DURING PASS 2. THE TAPE ROUTINES STILL EXIST BUT THE OUTPUT HAS BEEN REDIRECTED TO THE DISK I/O. IF YOU WISH TO RESTORE TAPE OUTPUT SIMPLY CHANGE THE JUMP INSTRUCTION AT $0323.

IMPLEMENTING THE PATCH

LOAD THE TSC ASSEMBLER BY ANY MEANS AVAILABLE TO YOU. PAPER TAPE, KC STANDARD CASSETTE, AND COMPLETE LISTINGS ARE AVAILABLE FROM TSC. THE KC STANDARD CASSETTES MAY BE READ USING THE PERCOM CIS-30 CASSETTE INTERFACE.

YOU MAY ENTER THE PATCHES BY HAND BUT SINCE THE PATCHES ARE NOW INCLUDED ON PERCOM 'SOFTWARE DISKETTE #1' IT WILL BE EASIER TO OVERLAY THE ASSEMBLER PATCH FILE USING THE HEX LOADER PROGRAM (HEXLDR) ON THE DISKETTE AND DESCRIBED IN PERCOM TECHNICAL MEMO TM-LFD-400-04. AFTER MAKING THE PATCHES, YOU WILL WANT TO SAVE A COPY OF THE MODIFIED ASSEMBLER. SAVE MEMORY FROM $0300 TO $176F. THE PROGRAM START ADDRESS IS $0300.

PROCEDURE FOR USING THE ASSEMBLER

1. CREATE THE SOURCE FILE USING THE TSC EDITOR MODIFIED FOR USE WITH THE LFD-400. REFER TO PERCOM TECHNICAL MEMO TM-LFD-400-02.
2. SAVE THE SOURCE FILE ON THE DISK NOTING THE BEGINNING AND ENDING DISK LOCATIONS.
3. LOAD THE MODIFIED ASSEMBLER AND TYPE 'G'. THE ASSEMBLER WILL RESPOND WITH A PASS REQUEST TO WHICH YOU MUST ENTER 1, 2, OR 3

EXAMPLE:

PASS? 1

PASS 1 - CLEARS THE SYMBOL TABLE AND BUILDS A SYMBOL TABLE FROM THE SOURCE FILE.
PASS 2 AND 3 - ARE AS DESCRIBED ON PAGE 8 AND 9 OF THE TSC ASSEMBLER MANUAL. PASS 1 MUST BE RUN BEFORE EITHER PASS 2 OR 3. YOU SHOULD ALSO BE AWARE THAT THE SYMBOL TABLE IS DESTROYED BY THE SYMBOL TABLE SORT WHICH OCCURS AT THE END OF PASS 2.

4. IF YOU ENTER PASS 1 THE ASSEMBLER WILL NEXT REQUEST THE 'SOURCE'. SIMPLY ENTER THE DRIVE AND SECTOR OF THE BEGINNING OF THE SOURCE FILE.

EXAMPLE:

PASS? 1 SOURCE? 1045<CR>

THIS STARTS A PASS 1 SYMBOL SORT ON THE DISK IN DRIVE #1 BEGINNING AT SECTOR 45 AND WILL CONTINUE TO THE END OF THE FILE. THE DISK DRIVE WILL TURN ON AND REMAIN ON UNTIL THE PASS IS COMPLETED. WHEN FINISHED, THE ASSEMBLER WILL AGAIN REQUEST 'PASS?'

5. IF YOU ENTER PASS 2 OR 3, THE ASSEMBLER WILL REQUEST 'OBJECT?'. ENTER THE DRIVE AND SECTOR WHERE YOU WISH THE OBJECT FILE TO BE WRITTEN. THIS CAN BE ON THE SAME DISK AS CONTAINS THE SOURCE FILE. JUST MAKE SURE THE OBJECT FILE WILL NOT OVERWRITE THE SOURCE FILE OR OTHER VALUABLE FILES ON THE SAME DISK. YOU MUST ENTER AN OBJECT FILE EVEN IF NO OBJECT FILE IS TO BE CREATED.

EXAMPLE:

PASS? 1 SOURCE? 1045<CR>
PASS? 2 OBJECT? 1100<CR>

IN THIS EXAMPLE THE SOURCE FILE IS ON DRIVE ONE BEGINNING AT SECTOR 45. WHEN PASS 1 WAS COMPLETED, THE OPERATOR SPECIFIED PASS 2 WITH OBJECT TO BE WRITTEN ON DRIVE ONE BEGINNING AT SECTOR 100. AN OBJECT FILE WILL BE CREATED DURING PASS 2 IF THE 'TAPE' OPTION IS ON. AN OBJECT FILE WILL ALWAYS BE CREATED DURING PASS 3.

LOADING THE OBJECT FILE

THE OBJECT FILE CREATED BY THE ASSEMBLER IS IN THE MOTOROLA ASCII HEX FORMAT WITH WHICH YOU ARE FAMILIAR. SINCE THIS CANNOT BE LOADED DIRECTLY INTO MEMORY FOR PROGRAM EXECUTION, THE FILE MUST BE TRANSLATED INTO BINARY MACHINE CODE AS IT IS LOADED INTO MEMORY. THE 'HEXLDR' ROUTINE DESCRIBED IN PERCOM TECHNICAL MEMO TM-LFD-400-04 PERFORMS THIS FUNCTION. THIS ROUTINE IS INCLUDED ON 'SOFTWARE DISKETTE #1' ALONG WITH THE EDITOR, ASSEMBLER, AND BASIC PATCHES. IN FACT THE 'HEXLDR' ROUTINE IS USED TO OVERLAY THE PATCHES ON THE EXISTING SOFTWARE.

ONCE IN MEMORY, THE DATA MAY BE DUMPED BACK ONTO THE DISK IN A MORE COMPACT BINARY FORMAT USING THE 'SAVE' ROUTINE IN MINIDOS(TM).

FINAL COMMENTS

A. IF YOU MAKE AN ERROR WHEN ENTERING THE SOURCE OR OBJECT FILE, JUST KEEP TYPING. THE ASSEMBLER USES ONLY THE LAST 4 DIGITS ENTERED.

B. YOU CAN 'ESCAPE' TO YOUR MONITOR WITH THE KEYBOARD 'ESCAPE' KEY.

C. THE ASSEMBLER AS CONFIGURED RUNS IN 8K OF MEMORY. THIS IS ENOUGH FOR APPROXIMATELY 100-150 SYMBOLS. IF THIS IS NOT ENOUGH, CHANGE THE UPPER LIMIT OF THE SYMBOL TABLE IN LOCATION $0305-$0306. THE NUMBER YOU SUBSTITUTE MUST FALL ON AN EIGHT BYTE BOUNDARY.

TSC SOFTWARE IS AVAILABLE FROM MANY LOCAL COMPUTER STORES. THE ADDRESS IF YOU ARE UNABLE TO OBTAIN THE SOFTWARE LOCALLY IS:

TECHNICAL SYSTEMS CONSULTANTS
P. O. BOX 2574
WEST LAFAYETTE, IN 47906
(317) 742-7509

```
                        NAM   TSC       ASSEMBLER PATCH VER 1.4
                      * COPYRIGHT 1978 PERCOM DATA CO.  INC.
                      * ALL RIGHTS RESERVED
                      * WRITTEN BY H.A.  MAUCH
                      * IMPROVEMENTS BY OTHERS
                      * MODIFIED FOR MINIDOS VER 1.4 9/30/78

                      * DISK HEADER
(0000)                DRV     EQU   0
(0001)                TRKSEC  EQU   $01
(0003)                BAKLNK  EQU   $03
(0005)                FWDLNK  EQU   $05
(0007)                BYTCNT  EQU   $07
(0008)                TRGTAD  EQU   $08
(000A)                FILTYP  EQU   $0A
(0014)                TA      EQU   $14
(0016)                TW      EQU   $16
                      * ASSEMBLER TEMPORARY LOCATIONS
(0040)                LBLBEG  EQU   $40
(0042)                LBLEND  EQU   $42
(0044)                SOURCE  EQU   $44
(0046)                OBJECT  EQU   $46
(0048)                DOBCNT  EQU   $48
(0049)                DOBPTR  EQU   $49
(004B)                PC      EQU   $4B
(004D)                SRCPTR  EQU   $4D
(004F)                LABEL   EQU   $4F
(0055)                PRFLG   EQU   $55
(0058)                ENDFLG  EQU   $58
(005D)                P3FLG   EQU   $5D
(0062)                OBJINT  EQU   $62
(0067)                SPSAVE  EQU   $67
(006D)                XTEMP2  EQU   $6D
(0079)                QTEMP2  EQU   $79
(007E)                OPCODE  EQU   $7E
(0085)                ERRPTR  EQU   $85
(0089)                OBJPTR  EQU   $89
(008B)                SRCDRV  EQU   $8B
(008C)                OBJDRV  EQU   $8C
(008D)                LINPTR  EQU   $8D
(008F)                PASS    EQU   $8F
(0090)                OPCNT   EQU   $90
(009A)                DSRPTR  EQU   $9A
(009C)                DSRCNT  EQU   $9C
(009D)                FLDCNT  EQU   $9D
(00A2)                SRCFIL  EQU   $A2
(00A7)                BUFCNT  EQU   $A7
(00A8)                LINCNT  EQU   $A8
(00AB)                MODFY   EQU   $AB
(00AC)                PAGENO  EQU   $AC
(00AE)                LIST    EQU   $AE
(00AF)                SYMBOL  EQU   $AF
(00B1)                PAGER   EQU   $B1
(00B2)                TAPE    EQU   $B2
(00B4)                OBJBUF  EQU   $B4
(0100)                ERRSTK  EQU   $100
(0280)                SRCBEG  EQU   $280      MAKE ROOM FOR LINE BUFFER
```

```
                    * ASSEMBLER LINKS
   (031B)           MON     EQU   $031B
   (031E)           OUTS    EQU   $031E
   (0320)           OUTCH   EQU   $0320
   (0326)           P1INIT  EQU   $0326
   (0422)           SHORT   EQU   $0422
   (04B2)           CONTRL  EQU   $04B2
   (04C4)           TAPEOF  EQU   $04C4
   (055E)           SYMGEN  EQU   $055E
   (07AB)           PDATA   EQU   $07AB
   (07BA)           PCRLF   EQU   $07BA
   (08A2)           PUTLBL  EQU   $08A2
   (0905)           FNDLBL  EQU   $0905
   (091F)           FNDOPT  EQU   $091F
   (0C44)           FND222  EQU   $0C44
   (11D1)           EJSTR   EQU   $11D1
   (1489)           OBJCOD  EQU   $1489
                    * MINIDOS(TM) LINKS
   (C00C)           RDSEC   EQU   $C00C
   (C00F)           WRTSEC  EQU   $C00F
   (C021)           FWDCAL  EQU   $C021
   (C027)           INITRK  EQU   $C027
   (C31F)           LSTSEC  EQU   $C31F
   (C353)           TYPERR  EQU   $C353
   (C36B)           CVTDTS  EQU   $C36B
   (CC03)           DVSTAT  EQU   $CC03
   (0004)           EOT     EQU   $04        END OF FILE MARK

   (E1AC)           INPCON  EQU   $E1AC
   (E1D1)           OUTCON  EQU   $E1D1
   (0000)           INITPR  EQU   *
   (1970)           SYMBEG  EQU   $1970
   (1FFF)           SYMEND  EQU   $1FFF

   (0300)                   ORG   $0300
   (0300)           MAIN    EQU   *
0300 CE 1970                LDX   #SYMBEG    SETUP SYMBOL TABLE
0303 DF 40                  STX   LBLBEG
0305 CE 1FFF                LDX   #SYMEND
0308 DF 42                  STX   LBLEND
030A 7E 1628                JMP   ASM
030D 7E E1AC                JMP   INPCON     INPUT FROM CONSOLE
0310 7E E1D1                JMP   OUTCON     OUTPUT TO CONSOLE
0313 7E 0000                JMP   INITPR     INITIALIZE PRINTER PORT
   (0323)                   ORG   $0323
0323 7E 1577    DSKOUT      JMP   PUTCHR     REDIRECT TAPE TO DISK

                    * CHANGES IN PASS 2 INITIALIZATION
   (036F)                   ORG   $036F
036F CE 0100    P2INIT      LDX   #ERRSTK    INITIALIZE ERROR PTR
0372 DF 85                  STX   ERRPTR
0374 CE 0000                LDX   #0
0377 DF 4B                  STX   PC         INITIALIZE PC
0379 08                     INX
037A DF AC                  STX   PAGENO     INITIALIZE PAGE NUMBER
037C 86 06                  LDA A #6         INIT LINE COUNT
037E 97 A8                  STA A LINCNT
0380 4F                     CLR A
```

```
0381 97 A7            STA A BUFCNT
0383 97 58            STA A ENDFLG
0385 43               COM A
0386 97 62            STA A OBJINT     SET OBJECT TOGGLE
0388 CE 00B4          LDX   #OBJBUF
038B DF 89            STX   OBJPTR
038D 39               RTS

                    * ASSEMBLY PASS ONE *
038E 9F 67    PASONE  STS   SPSAVE
0390 7F 008F          CLR   PASS       SET PASS 1
0393 8D 21            BSR   INTDBF     INIT DISK BUFFERS
0395 BD 15C1  PASS1   JSR   GETLIN     GET A LINE OF SOURCE
0398 25 1B            BCS   PASS13     BRANCH IF END OF FILE
039A CE 0280          LDX   #SRCBEG    POINT TO BEG OF LINE
039D 09               DEX
039E DF 4D            STX   SRCPTR
03A0 BD 0B75          JSR   PARSE
03A3 96 4F            LDA A LABEL
03A5 27 03            BEQ   PASS11
03A7 BD 08A2          JSR   PUTLBL
03AA 96 55    PASS11  LDA A PRFLG
03AC 26 03            BNE   PASS12
03AE BD 0C44          JSR   FND222
03B1 96 58    PASS12  LDA A ENDFLG
03B3 27 E0            BEQ   PASS1      BRANCH IF NOT DONE
03B5 39       PASS13  RTS
                    * INITIALIZE DISK BUFFERS
03B6 4F       INTDBF  CLR A
03B7 97 48            STA A DOBCNT     CLEAR OBJ BUF CNTR
03B9 CE 1867          LDX   #DOBBUF    INIT OBJECT PTR
03BC DF 49            STX   DOBPTR
03BE 97 9C            STA A DSRCNT     INIT SOURCE BUFFER
03C0 39               RTS

                    * ASSEMBLY PASS 2 *
03C1 86 01    PASTWO  LDA A #01        SET PASS 2
03C3 97 8F            STA A PASS
03C5 8D EF            BSR   INTDBF     INIT DISK BUFFERS
03C7 DE 4B    PASS2   LDX   PC
03C9 DF 6D            STX   XTEMP2
03CB BD 15C1          JSR   GETLIN     GET LINE OF SOURCE
03CE 24 03            BCC   PASS2Y
03D0 7E 04D6          JMP   FIN        JUMP IF END OF FILE
03D3 CE 0280  PASS2Y  LDX   #SRCBEG
03D6 09               DEX
03D7 DF 4D            STX   SRCPTR
03D9 BD 0B75  PASS2A  JSR   PARSE
03DC 96 55            LDA A PRFLG
03DE 26 03            BNE   PASS2X
03E0 BD 091F          JSR   FNDOPT
03E3 96 90    PASS2X  LDA A OPCNT
03E5 27 0B            BEQ   PASS2C
03E7 96 5D            LDA A P3FLG
03E9 27 04            BEQ   OBJGEN
03EB 96 B2            LDA A TAPE
03ED 27 03            BEQ   PASS2C
03EF BD 1489  OBJGEN  JSR   OBJCOD
```

```
03F2 96 5D    PASS2C LDA A P3FLG
03F4 26 2C           BNE   SHORT
03F6 7E 04A4         JMP   NOERR4

   (043F)            ORG   $043F      FIX ERROR PRINTOUTS
043F 9C 4B           CPX   PC

   (04A4)            ORG   $04A4
04A4 96 58    NOERR4 LDA A ENDFLG
04A6 26 2E           BNE   FIN
04A8 7E 03C7         JMP   PASS2
                   * PATCH TO INJECT TAPE(DISK) CONTROL CODE
   (04C3)            ORG   $04C3
04C3 12              FCB   $12,'S,'9,EOT,$14
04C4 53 39
04C6 04 14

                   * CHANGES TO END OF ASSEMBLY CLEANUP
   (04D6)            ORG   $04D6
   (04D6)      FIN   EQU   *

   (04EF)            ORG   $04EF
04EF 27 0E           BEQ   FIN2
   (04F4)            ORG   $04F4
04F4 CE 04C4         LDX   #TAPEOF    TAPE(DISK) OFF STRING
04F7 8D B9           BSR   CONTRL
04F9 BD 07BA         JSR   PCRLF
04FC BD C31F         JSR   LSTSEC     REPORT LAST SECTOR USED
04FF 96 5D    FIN2   LDA A P3FLG
0501 27 1D           BEQ   FIN6
0503 96 AF    FIN5   LDA A SYMBOL
0505 26 57           BNE   SYMGEN
0507 96 AE           LDA A LIST
0509 27 15           BEQ   FIN6
050B BD 07BA  FIN3   JSR   PCRLF
050E 96 B1           LDA A PAGER
0510 27 06           BEQ   FIN4
0512 CE 11D1         LDX   #EJSTR
0515 7E 07AB         JMP   PDATA
0518 C6 04    FIN4   LDA B #4
051A BD 07BA  GAPX   JSR   PCRLF
051D 5A              DEC B
051E 26 FA           BNE   GAPX
0520 39       FIN6   RTS

   (0560)            ORG   $560
0560 27 A9           BEQ   FIN3
   (05B8)            ORG   $5B8
05B8 7E 050B         JMP   FIN3

   (0642)            ORG   $0642      COLUMNER PATCH
0642 7E 16DA         JMP   PRTX

   (07C5)            ORG   $07C5      PAGE LENGTH CONTROL
07C5 40              FCB   $40
                   *
   (07FE)            ORG   $07FE      FIX ERROR PRINT FOR PC
07FE 96 4B           LDA A PC         GET HIGH
```

```
0800 D6 4C          LDA B PC+1      GET LOW
                    *
   (0B75)           ORG   $0B75
0B75 86 03  PARSE   LDA A #3        FIX LINBYT

   (0F21)           ORG   $0F21
0F21 02 80          FDB   SRCBEG MAKE ROOM FOR LINE BUFFER

   (1143)           ORG   $1143     TOP MARGIN CONTROL
1143 07             FCB   $07

   (1577)           ORG   $1577
                    * ROUTINE TO SAVE OBJECT FILE ON DISK *

1577 FF 15BF PUTCHR STX   PUT4      SAVE INDEX
157A 37             PSH B           SAVE B
157B 81 14          CMP A #$14      CHECK FOR END OF FILE (DC4)
157D 27 0C          BEQ   PUT1
157F DE 49          LDX   DOBPTR    GET DISK OBJECT POINTER
1581 A7 00          STA A 0,X       PUT CHAR IN DISK OBJ BUF
1583 08             INX
1584 DF 49          STX   DOBPTR
1586 7C 0048        INC   DOBCNT    BUMP OBJECT BUFFER COUNTER
1589 26 2C          BNE   PUT3      BRANCH IF NOT FULL
                    * WRITE OBJECT BUFFER TO DISK
158B 96 8C  PUT1    LDA A OBJDRV
158D 97 00          STA A DRV       SELECT OBJECT DRIVE
158F DE 46          LDX   OBJECT    GET OBJECT FILE PTR
1591 DF 01          STX   TRKSEC    SET UP SECTOR HEADER
1593 DF 05          STX   FWDLNK
1595 BD C021        JSR   FWDCAL    CALCULATE FWD LINK
1598 CE 0000        LDX   #0
159B DF 03          STX   BAKLNK    CLEAR BACK LINK
159D DF 08          STX   TRGTAD    CLEAR TARGET ADDRESS
159F 86 0C  PUT2    LDA A #$0C      IDENTIFY AS HEX FILE
15A1 97 0A          STA A FILTYP
15A3 96 48          LDA A DOBCNT    SET BYTE COUNT
15A5 97 07          STA A BYTCNT
15A7 CE 1867        LDX   #DOBBUF   POINT TO DISK OBJECT BUF
15AA DF 49          STX   DOBPTR    INIT DOB POINTER
15AC DF 14          STX   TA        SET UP DATA SOURCE
15AE BD C00F        JSR   WRTSEC    WRITE THE SECTOR
15B1 25 09          BCS   PUTERR    BRANCH IF ERROR
15B3 DE 05          LDX   FWDLNK    CLEAN UP
15B5 DF 46          STX   OBJECT
15B7 33     PUT3    PUL B           RESTORE B
15B8 FE 15BF        LDX   PUT4      RESTORE INDEX
15BB 39             RTS
15BC 7E 1648 PUTERR JMP   ASMERR
15BF         PUT4   RMB   2         INDEX STORAGE

                    * ROUTINES TO GET SOURCE CODE FROM DISK *

                    * GET A LINE OF TEXT FROM DISK SOURCE BUFFER
15C1 CE 0280 GETLIN LDX   #SRCBEG   POINT TO LINE BUFFER
15C4 8D 12   GETL1  BSR   GETCHR    GET A CHAR FROM SOURCE BUF
15C6 25 0F          BCS   GETL3     BRANCH IF END OF FILE
15C8 A7 00          STA A 0,X       STORE IN LINE BUFFER
```

```
15CA 81 0D          CMP A #$0D      CHECK FOR CR
15CC 27 08          BEQ   GETL2     BRANCH IF CR
15CE 8C 02FF        CPX   #SRCBEG+127 CHECK FOR END OF LINE BUF
15D1 27 F1          BEQ   GETL1
15D3 08             INX
15D4 20 EE          BRA   GETL1
15D6 0C       GETL2 CLC             CLEAR EOF FLAG
15D7 39       GETL3 RTS
                    * GET A CHAR FROM DISK VIA DISK BUFFER
15D8 FF 15FF  GETCHR STX  GETC3     SAVE INDEX
15DB 37             PSH B           SAVE B
15DC 7D 009C        TST   DSRCNT    CHECK SOURCE BUFFER COUNTER
15DF 26 0E          BNE   GETC1     BRANCH IF BUFFER NOT EMPTY
15E1 8D 1E          BSR   GTSCTR    GET NEXT SECTOR
15E3 25 15          BCS   GETC2     BRANCH IF EOF
15E5 CE 1767        LDX   #DSRBUF   POINT TO SOURCE BUFFER
15E8 DF 9A          STX   DSRPTR
15EA 96 07          LDA A BYTCNT    GET BYTE COUNT
15EC 40             NEG A
15ED 97 9C          STA A DSRCNT
15EF DE 9A    GETC1 LDX   DSRPTR    GET SOURCE BUF PTR
15F1 A6 00          LDA A 0,X       GET CHARACTER
15F3 08             INX
15F4 DF 9A          STX   DSRPTR
15F6 7C 009C        INC   DSRCNT
15F9 0C             CLC             CLEAR END OF FILE FLAG
15FA FE 15FF  GETC2 LDX   GETC3
15FD 33             PUL B           RESTORE B
15FE 39             RTS
15FF          GETC3 RMB   2         INDEX STORAGE
                    * GET A SECTOR OF DATA FROM DISC
1601 FF 1626  GTSCTR STX  GTS2      SAVE INDEX
1604 96 8B          LDA A SRCDRV
1606 97 00          STA A DRV       SELECT SOURCE DRIVE
1608 DE 44          LDX   SOURCE    GET SOURCE FILE PTR
160A 0D             SEC             SET END OF FILE FLAG
160B 27 15          BEQ   GTS1      BRANCH IF EOF
160D DF 01          STX   TRKSEC    SET UP DISK HEADER
160F CE 1767        LDX   #DSRBUF   SET UP TARGET ADD
1612 DF 16          STX   TW
1614 BD C00C        JSR   RDSEC     READ THE SECTOR
1617 25 2F          BCS   ASMERR    BRANCH IF READ ERROR
1619 DE 05          LDX   FWDLNK    PICK UP FORWARD LINK
161B DF 44          STX   SOURCE    PUT IN SOURCE PTR
161D 96 07          LDA A BYTCNT    PICK UP BYTE COUNT
161F 97 9C          STA A DSRCNT    PUT IN SOURCE BUF CNTR
1621 0C             CLC             CLEAR END OF FILE FLAG
1622 FE 1626  GTS1  LDX   GTS2      RESTORE INDEX
1625 39             RTS
1626          GTS2  RMB   2         INDEX STORAGE

                    * MAIN ASSEMBY CONTROL LOOP *
1628 8E A07F  ASM   LDS   #$A07F    SET STACK
162B BD 07BA        JSR   PCRLF
162E BD C027        JSR   INITRK
1631 CE 16BF        LDX   #PRTPAS
1634 BD 07AB        JSR   PDATA     PRINT "PASS?"
1637 BD 1692        JSR   INCHAR    GET PASS
```

```
163A 81 33            CMP A #$33
163C 27 32            BEQ   PAS3
163E 81 32            CMP A #$32
1640 27 28            BEQ   PAS2
1642 81 31            CMP A #$31
1644 27 07            BEQ   PAS1
1646 86 0A            LDA A #$A       INVALID PASS
1648 BD C353  ASMERR  JSR   TYPERR
164B 20 DB            BRA   ASM
164D BD 0326  PAS1    JSR   P1INIT
1650 7F 0058          CLR   ENDFLG
1653 CE 16C6          LDX   #PRTSRC
1656 BD 07AB          JSR   PDATA     PRINT "SOURCE?"
1659 BD 169D          JSR   IN4DEC    GET SOURCE FILE
165C 97 8B            STA A SRCDRV
165E DF 44            STX   SOURCE
1660 DF A2            STX   SRCFIL
1662 BD 07BA          JSR   PCRLF
1665 BD 038E          JSR   PASONE
1668 20 BE            BRA   ASM
166A 86 FF    PAS2    LDA A #$FF
166C 97 5D            STA A P3FLG
166E 20 03            BRA   P2X
1670 7F 005D  PAS3    CLR   P3FLG     SET PASS 3 FLAG
1673 CE 16D0  P2X     LDX   #PRTOBJ
1676 BD 07AB          JSR   PDATA     PRINT "OBJECT?"
1679 BD 169D          JSR   IN4DEC    GET OBJECT FILE
167C 97 8C            STA A OBJDRV
167E DF 46            STX   OBJECT
1680 BD 07BA          JSR   PCRLF
1683 DE A2            LDX   SRCFIL    REESTABLISH SOURCE FILE
1685 DF 44            STX   SOURCE
1687 BD 07BA          JSR   PCRLF
168A BD 036F          JSR   P2INIT
168D BD 03C1          JSR   PASTWO
1690 20 96            BRA   ASM

1692 BD E1AC  INCHAR  JSR   INPCON
1695 81 1B            CMP A #$1B      CHECK FOR ESCAPE
1697 27 01            BEQ   IN1
1699 39               RTS
169A 7E 031B  IN1     JMP   MON

169D BD 138A  IN4DEC  JSR   $138A     ZERO 16 BIT ACCUM
16A0 BD 1692  IN4A    JSR   INCHAR    GET CHAR
16A3 16               TAB
16A4 BD 134A          JSR   $134A     CONVERT TO BCD
16A7 24 0C            BCC   IN4C
16A9 BD 1391          JSR   $1391     SHIFT LEFT TWO PLACES
16AC BD 1391          JSR   $1391     SHIFT LEFT TWO PLACES
16AF DB 7A            ADD B QTEMP2+1
16B1 D7 7A            STA B QTEMP2+1
16B3 20 EB            BRA   IN4A
16B5 DE 79    IN4C    LDX   QTEMP2
16B7 BD C36B          JSR   CVTDTS
16BA 96 00            LDA A DRV
16BC DE 01            LDX   TRKSEC
16BE 39               RTS
```

```
16BF 50        PRTPAS FCC   'PASS? '
16C0 41 53
16C2 53 3F
16C4 20
16C5 04               FCB   4
16C6 20        PRTSRC FCC   ' SOURCE? '
16C7 53 4F
16C9 55 52
16CB 43 45
16CD 3F 20
16CF 04               FCB   4
16D0 20        PRTOBJ FCC   ' OBJECT? '
16D1 4F 42
16D3 4A 45
16D5 43 54
16D7 3F 20
16D9 04               FCB   4

               * COLUMNNER ALIGNMENT PATCH
16DA DE 8D     PRTX   LDX   LINPTR
16DC 7F 009D          CLR   FLDCNT
16DF A6 00            LDA A 0,X
16E1 81 2A            CMP A #$2A
16E3 27 4E            BEQ   P2
16E5 8D 60            BSR   PUNSP
16E7 C6 07            LDA B #7
16E9 8D 77            BSR   TAB
16EB C6 02            LDA B #2
16ED A6 00            LDA A 0,X
16EF 81 46            CMP A #'F
16F1 26 08            BNE   P3
16F3 8D 52            BSR   PUNSP
16F5 C6 0D            LDA B #13
16F7 8D 69            BSR   TAB
16F9 20 38            BRA   P2
16FB A6 00     P3     LDA A 0,X        PRINT OUT OPERATOR
16FD 08               INX
16FE 7C 009D          INC   FLDCNT
1701 BD 0320          JSR   OUTCH
1704 5A               DEC B
1705 2A F4            BPL   P3
1707 D6 AB            LDA B MODFY      IS MODIFIER FLAG SET?
1709 5A               DEC B
170A 2B 0B            BMI   P4
170C 8D 47            BSR   PUN3+1
170E BD 031E          JSR   OUTS
1711 7C 009D          INC   FLDCNT     SPACE AND PRINT MODIFIER
1714 8D 31            BSR   PUNSP
1716 09               DEX
1717 D6 7E     P4     LDA B OPCODE     GET OP CODE
1719 27 0C            BEQ   P5         IF ASM DIRECTIVE - NORMAL
171B C4 F0            AND B #$F0       MASK OFF LOW NYBBLE
171D 27 0E            BEQ   P6
171F C1 60            CMP B #$60       DETERMINE IF OP CODE HAS OPERAND
1721 24 04            BHS   P5
1723 C1 20            CMP B #$20
1725 26 06            BNE   P6
```

```
1727 8D 1E     P5       BSR   PUNSP      FIND OPERAND FIELD
1729 C6 0D              LDA B #13        TAB TO COLUMN 13
172B 8D 35              BSR   TAB
172D 8D 18     P6       BSR   PUNSP      FIND COMMENT FIELD
172F C6 17              LDA B #23        TAB TO COLUMN 23
1731 8D 29              BSR   TB1
1733 A6 00     P2       LDA A 0,X        PRINT UNTIL CR
1735 08                 INX
1736 81 0D              CMP A #$D
1738 27 05              BEQ   PEND
173A BD 0320            JSR   OUTCH
173D 20 F4              BRA   P2
173F 39        PEND     RTS
1740 08        PUN2     INX
1741 7C 009D            INC   FLDCNT
1744 BD 0320            JSR   OUTCH
1747 A6 00     PUNSP    LDA A 0,X        PRINT UNTIL SPACE FOUND,
1749 81 0D              CMP A #$D        FIND NEXT FIELD,
174B 26 03              BNE   PUN1       AND RETURN.
174D 31                 INS
174E 31                 INS
174F 39                 RTS              EARLY RETURN ON CR
1750 81 20     PUN1     CMP A #'
1752 26 EC              BNE   PUN2
1754 08        PUN3     INX
1755 A6 00              LDA A 0,X
1757 81 20              CMP A #'
1759 27 F9              BEQ   PUN3
175B 39                 RTS
175C 7C 009D   TB1      INC   FLDCNT
175F BD 031E            JSR   OUTS
1762 D1 9D     TAB      CMP B FLDCNT
1764 2E F6              BGT   TB1
1766 39                 RTS
1767           DSRBUF   RMB   256
1867           DOBBUF   RMB   256
   (1967)      FREE     EQU   *
                       END
      00  ERROR(S) DETECTED
```

SYMBOL TABLE:

ASM	1628	ASMERR	1648	BAKLNK	0003	BUFCNT	00A7
BYTCNT	0007	CONTRL	04B2	CVTDTS	C36B	DOBBUF	1867
DOBCNT	0048	DOBPTR	0049	DRV	0000	DSKOUT	0323
DSRBUF	1767	DSRCNT	009C	DSRPTR	009A	DVSTAT	CC03
EJSTR	11D1	ENDFLG	0058	EOT	0004	ERRPTR	0085
ERRSTK	0100	FILTYP	000A	FIN	04D6	FIN2	04FF
FIN3	050B	FIN4	0518	FIN5	0503	FIN6	0520
FLDCNT	009D	FND222	0C44	FNDLBL	0905	FNDOPT	091F
FREE	1967	FWDCAL	C021	FWDLNK	0005	GAPX	051A
GETC1	15EF	GETC2	15FA	GETC3	15FF	GETCHR	15D8
GETL1	15C4	GETL2	15D6	GETL3	15D7	GETLIN	15C1
GTS1	1622	GTS2	1626	GTSCTR	1601	IN1	169A
IN4A	16A0	IN4C	16B5	IN4DEC	169D	INCHAR	1692
INITPR	0000	INITRK	C027	INPCON	E1AC	INTDBF	03B6
LABEL	004F	LBLBEG	0040	LBLEND	0042	LINCNT	00A8

LINPTR	008D	LIST	00AE	LSTSEC	C31F	MAIN	0300
MODFY	00AB	MON	031B	NOERR4	04A4	OBJBUF	00B4
OBJCOD	1489	OBJDRV	008C	OBJECT	0046	OBJGEN	03EF
OBJINT	0062	OBJPTR	0089	OPCNT	0090	OPCODE	007E
OUTCH	0320	OUTCON	E1D1	OUTS	031E	P1INIT	0326
P2	1733	P2INIT	036F	P2X	1673	P3	16FB
P3FLG	005D	P4	1717	P5	1727	P6	172D
PAGENO	00AC	PAGER	00B1	PARSE	0B75	PAS1	164D
PAS2	166A	PAS3	1670	PASONE	038E	PASS	008F
PASS1	0395	PASS11	03AA	PASS12	03B1	PASS13	03B5
PASS2	03C7	PASS2A	03D9	PASS2C	03F2	PASS2X	03E3
PASS2Y	03D3	PASTWO	03C1	PC	004B	PCRLF	07BA
PDATA	07AB	PEND	173F	PRFLG	0055	PRTOBJ	16D0
PRTPAS	16BF	PRTSRC	16C6	PRTX	16DA	PUN1	1750
PUN2	1740	PUN3	1754	PUNSP	1747	PUT1	158B
PUT2	159F	PUT3	15B7	PUT4	15BF	PUTCHR	1577
PUTERR	15BC	PUTLBL	08A2	QTEMP2	0079	RDSEC	C00C
SHORT	0422	SOURCE	0044	SPSAVE	0067	SRCBEG	0280
SRCDRV	008B	SRCFIL	00A2	SRCPTR	004D	SYMBEG	1970
SYMBOL	00AF	SYMEND	1FFF	SYMGEN	055E	TA	0014
TAB	1762	TAPE	00B2	TAPEOF	04C4	TB1	175C
TRGTAD	0008	TRKSEC	0001	TW	0016	TYPERR	C353
WRTSEC	C00F	XTEMP2	006D				

PERCOM DATA CO.

TECHNICAL MEMO

TM-LFD-400-04
LFD-400 FLOPPY DISK SYSTEM
MARCH 31, 1978

SUBJECT: A LOADER FOR LOADING MOTOROLA ASCII-HEX OBJECT FILES FROM DISK INTO MEMORY.

THE OBJECT CODE OUTPUT FROM MOST 6800 ASSEMBLERS IN IS THE MOTOROLA ASCII-HEX FORMAT. SINCE THIS CANNOT BE LOADED DIRECTLY INTO MEMORY FOR PROGRAM EXECUTION, FILE MUST BE TRANSLATED INTO BINARY MACHINE CODE AS IT IS LOADED. THE ATTACHED ROUTINE PERFORMS THIS FUNCTION ON ASCII-HEX OBJECT FILES STORED ON THE PERCOM LFD-400 DISK.

ALTHOUGH THE LISTING OF THE HEX LOADER WAS ASSEMBLED TO ADDRESS $2000, THE ROUTINE IS RELOCATABLE AND MAY BE PLACED ANYWHERE IN MEMORY. THE ROUTINE AND ITS BUFFER REQUIRE 510 BYTES OF CONTIGUOUS MEMORY. THE ROUTINE ALSO USES 16 BYTES OF THE MONITOR RAM ($A04A-$A059) FOR TEMPORARY POINTER STORAGE.

THIS LOADER ROUTINE IS ALSO USED TO `OVERLAY` THE EDITOR, ASSEMBLER, AND BASIC PATCHES DESCRIBED IN EARLIER TECHNICAL MEMOS AND CONTAINED ON PERCOM `SOFTWARE DISKETTE #1`. IF YOU DO NOT HAVE `SOFTWARE DISKETTE #1` ENTER THE LOADER CODE BY HAND USING THE ATTACHED LISTING. SAVE A COPY ON DISKETTE.

USING THE LOADER

LOAD THE DISK-HEX-LOADER PROGRAM INTO THE COMPUTER USING MINIDOS(TM). SINCE THE ROUTINE IS RELOCATABLE, IT MAY BE LOCATED ANYWHERE IN MEMORY AWAY FROM POTENTIAL INTERFERENCE.

WHEN YOU JUMP TO THE STARTING ADDRESS, THE LOADER WILL REQUEST A FILE. ENTER THE FOUR DIGIT DRIVE AND SECTOR NUMBER OF THE OBJECT FILE YOU WISH TO LOAD. THE DRIVE WILL START AND WHEN THE LOAD IS COMPLETED THE LOADER WILL AGAIN REQUEST A FILE. IF YOU DO NOT WISH TO LOAD ANOTHER FILE STRIKE THE `ESCAPE` KEY ON YOUR KEYBOARD. THIS RETURNS CONTROL TO THE MONITOR.

EXAMPLE:

```
FILE? 1100
FILE? <ESC>
$
```

IN THIS EXAMPLE, THE OBJECT CODE FILE ON DRIVE ONE BEGINNING AT SECTOR 100 WAS LOADED. SINCE THE OPERATOR DID NOT WISH TO LOAD ANYMORE OBJECT CODE FILES, THE `ESCAPE` KEY RETURNED CONTROL TO THE MONITOR.

```
                        NAM   HEXLDR    VER 2.0
                      * COPYRIGHT (C) 1978 PERCOM DATA CO. INC.
                      * ALL RIGHTS RESERVED
                      * WRITTEN BY H.A. MAUCH
                      * MODIFIED FOR MINIDOS 1.4 AUG 20, 1978
                      ***********************************************
                      * THIS PROGRAM READS AND CONVERTS THE ASCII-HEX
                      * OBJECT FILE CREATED BY THE ASSEMBLER TO BINARY.
                      * THE BINARY CODE IS THEN STORED IN MEMORY FOR
                      * PROGRAM EXECUTION. REQUIRES MINIDOS VERSION 1.4.
                      *
                      * ALTHOUGH THIS LISTING WAS ASSEMBLED TO START AT
                      * ADDRESS $2000, `HEXLDR` IS RELOCATABLE AND MAY
                      * BE PLACED ANYWHERE IN MEMORY.  `HEXLDR` AND ITS
                      * BUFFER REQUIRE 470 BYTES OF CONTIGUOUS MEMORY.
                      * `HEXLDR` ALSO USES 14 BYTES OF THE MONITOR RAM
                      * ($A04A-$A057) FOR TEMPORARY POINTER STORAGE.
                      ***********************************************

   (0001)      TRKSEC EQU   $01
   (0005)      FWDLNK EQU   $05
   (0007)      BYTCNT EQU   $07
   (0016)      TW     EQU   $16
   (00FF)      EOF    EQU   $FF        END OF FILE FLAG
   (C00C)      RDSEC  EQU   $C00C
   (C01E)      TYPERR EQU   $C01E
   (C363)      PCRLF  EQU   $C363
   (C003)      CVTDTS EQU   $C003
   (E07E)      PDATA  EQU   $E07E
   (E047)      BADDR  EQU   $E047
   (A04A)             ORG   $A04A
A04A           DSKFIL RMB   2
A04C           DSKPTR RMB   2
A04E           BYTECT RMB   1
A04F           BUFCNT RMB   1
A050           BUFADD RMB   2
A052           BUFPTR RMB   2
A054           ADDRES RMB   2
A056           XTEMP1 RMB   2

   (2000)             ORG   $2000

2000 8E A07F   LOAD   LDS   #$A07F     SET STACK
2003 8D 00            BSR   HERE       THIS SEQUENCE PERMITS THIS
2005 30        HERE   TSX              TO BE LOCATED ANYWHERE
2006 01               NOP
2007 A6 01            LDA A 1,X
2009 8B CB            ADD A #PROMPT-HERE
200B B7 A051          STA A BUFADD+1
200E A6 00            LDA A 0,X
2010 89 00            ADC A #0
2012 B7 A050          STA A BUFADD
2015 BD C363          JSR   PCRLF
2018 FE A050          LDX   BUFADD
201B BD E07E          JSR   PDATA
```

```
201E 08              INX
201F FF A050         STX   BUFADD
2022 BD E047         JSR   BADDR      GET FILE
2025 BD C003         JSR   CVTDTS
2028 25 38           BCS   ERR
202A DE 01           LDX   TRKSEC
202C FF A04A         STX   DSKFIL
202F FF A04C         STX   DSKPTR
2032 86 01           LDA A #1
2034 B7 A04F         STA A BUFCNT

2037 8D 54    LOAD3  BSR   GETCHR
2039 27 C5           BEQ   LOAD
203B 81 53           CMP A #'S
203D 26 F8           BNE   LOAD3
203F 8D 4C           BSR   GETCHR
2041 27 BD           BEQ   LOAD
2043 81 39           CMP A #'9
2045 27 B9           BEQ   LOAD
2047 81 31           CMP A #'1
2049 26 EC           BNE   LOAD3
204B 8D 28           BSR   ONEBYT     GET BYTE COUNT
204D 80 02           SUB A #2
204F B7 A04E         STA A BYTECT
2052 8D 13           BSR   TWOBYT
2054 8D 1F    LOAD11 BSR   ONEBYT
2056 7A A04E         DEC   BYTECT
2059 27 DC           BEQ   LOAD3
205B A7 00           STA A 0,X
205D 08              INX
205E 20 F4           BRA   LOAD11

2060 86 0E    ERRE   LDA A #$E        WHAT?
2062 BD C01E  ERR    JSR   TYPERR     ERROR TRAP
2065 20 99           BRA   LOAD       UNSTRUCTURED RETURN

2067 8D 0C    TWOBYT BSR   ONEBYT
2069 B7 A054         STA A ADDRES
206C 8D 07           BSR   ONEBYT
206E B7 A055         STA A ADDRES+1
2071 FE A054         LDX   ADDRES
2074 39              RTS
2075 8D 09    ONEBYT BSR   GETHEX
2077 48              ASL A
2078 48              ASL A
2079 48              ASL A
207A 48              ASL A
207B 16              TAB
207C 8D 02           BSR   GETHEX
207E 1B              ABA
207F 39              RTS
2080 8D 0B    GETHEX BSR   GETCHR
2082 27 DC           BEQ   ERRE
2084 80 30           SUB A #$30       REMOVE ASCII OFFSET
2086 81 09           CMP A #9
2088 2F 02           BLE   GH1
```

```
208A 80 07           SUB A #7
208C 39       GH1    RTS

208D FF A056  GETCHR STX   XTEMP1     SAVE INDEX
2090 37              PSH B           SAVE B
2091 7A A04F         DEC   BUFCNT     BUMP DISK BUFFER COUNTER
2094 26 16           BNE   GETC1      BRANCH IF NOT EMPTY
2096 86 FF           LDA A #EOF      END OF FILE DEFAULT
2098 FE A04C         LDX   DSKPTR
209B 27 18           BEQ   GETC2      BRANCH IF END OF FILE
209D 8D 1E           BSR   GTSCTR     GET NEXT SECTOR
209F 25 C1           BCS   ERR        BRANCH IF ERROR
20A1 FE A050         LDX   BUFADD     POINT TO DISK BUFFER
20A4 FF A052         STX   BUFPTR
20A7 96 07           LDA A BYTCNT     GET BYTE COUNT
20A9 B7 A04F         STA A BUFCNT
20AC FE A052  GETC1  LDX   BUFPTR     GET DISK BUF PTR
20AF A6 00           LDA A 0,X       GET CHARACTER
20B1 08              INX
20B2 FF A052         STX   BUFPTR
20B5 FE A056  GETC2  LDX   XTEMP1
20B8 33              PUL B           RESTORE B
20B9 81 FF           CMP A #EOF
20BB 0C              CLC
20BC 39              RTS

                     * GET A SECTOR OF DATA FROM DISC
                     * ON ENTRY, X CONTAINS TRK-SEC
20BD DF 01    GTSCTR STX   TRKSEC     SET UP DISK HEADER
20BF FE A050         LDX   BUFADD     SET UP TARGET ADDRESS
20C2 DF 16           STX   TW
20C4 BD C00C         JSR   RDSEC      READ THE SECTOR
20C7 25 06           BCS   GTS1       BRANCH IF ERROR
20C9 DE 05           LDX   FWDLNK
20CB FF A04C         STX   DSKPTR     PUT IN DISK PTR
20CE 0C              CLC
20CF 39       GTS1   RTS

20D0 46       PROMPT FCC   'FILE? '
20D1 49 4C
20D3 45 3F
20D5 20
20D6 04              FCB   $04
20D7          BUFFER RMB   256
                     END
      00  ERROR(S) DETECTED
```

```
                        NAM   DISKMAP   VERSION 2.0
                      * COPYRIGHT (C) 1978 PERCOM DATA CO. INC.
                      * ALL RIGHTS RESERVED
                      * WRITTEN BY R.R. WIER
                      * MODIFICATIONS BY H.A. MAUCH
                      * MODIFIED FOR MINIDOS 1.4  AUG 21, 1978
                      ********************************************
                      * THIS PROGRAM PROVIDES A MEANS TO STUDY THE
                      * CONTENT OF DISK SECTORS.  IT PRINTS OUT THE
                      * CONTENTS OF THE SECTOR HEADER AS WELL AS THE
                      * DATA CONTENT OF THE SECTOR IN BOTH HEX AND
                      * ASCII FORMATS.  UPON ENTRY, THE PROGRAM WILL
                      * ASK:
                      *      DISK MAP-HEADER ONLY? (Y OR N)
                      *
                      * A 'Y' RESPONSE WILL SUPPRESS THE DATA PRINT
                      * OUT AND WILL ONLY PRINT THE HEADER INFORMATION.
                      ********************************************

   (A07F)             STACK  EQU   $A07F
   (E0E3)             MON    EQU   $E0E3
   (E07E)             PDATA  EQU   $E07E
   (E0C8)             OUT4HS EQU   $E0C8
   (E0CA)             OUT2HS EQU   $E0CA
   (E0CC)             OUTS   EQU   $E0CC
   (E1AC)             INEEE  EQU   $E1AC
   (E1D1)             OUTCH  EQU   $E1D1

   (C00C)             RDSEC  EQU   $C00C
   (C01E)             TYPERR EQU   $C01E
   (C021)             FWDCAL EQU   $C021
   (C027)             INITRK EQU   $C027
   (C324)             PRTSEC EQU   $C324
   (C363)             PCRLF  EQU   $C363
   (C369)             INDTS  EQU   $C369

   (0000)             DRV    EQU   $0
   (0001)             TRKSEC EQU   $01
   (0003)             BAKLNK EQU   $03
   (0005)             FWDLNK EQU   $05
   (0016)             TW     EQU   $16
   (0010)             COUNT  EQU   16

   (0100)                    ORG   $0100
0100 8E A07F                 LDS   #STACK
0103 BD C027                 JSR   INITRK     INITIALIZE DRIVES
0106 BD C363                 JSR   PCRLF
0109 CE 0258                 LDX   #MAPMSG    PRINT "DISK MAP-HEADER ONLY? (Y OR N)"
010C BD E07E                 JSR   PDATA
010F BD E1AC                 JSR   INEEE      INPUT RESPONSE
0112 81 59                   CMP A #'Y
0114 26 01                   BNE   DM1        BRANCH IF "NO"
0116 4F                      CLR A            DATFLG = 0 IF HEADER ONLY
0117 B7 02DF   DM1           STA A DATFLG
011A BD C363                 JSR   PCRLF
```

```
011D CE 0278          LDX   #STMSSG   PRINT "FIRST SECTOR?"
0120 BD E07E          JSR   PDATA
0123 BD C369          JSR   INDTS     GET STARTING SECTOR
0126 DE 01            LDX   TRKSEC
0128 FF 02E1          STX   CURADD    SAVE STARTING SECTOR AS CURRENT ADDRESS
012B 96 00            LDA A DRV
012D B7 02E0          STA A DRVTMP
0130 CE 0287          LDX   #LSTMSG   PRINT "LAST SECTOR?"
0133 BD E07E          JSR   PDATA
0136 BD C369          JSR   INDTS     GET ENDING SECTOR
0139 DE 01            LDX   TRKSEC
013B FF 02E3          STX   ENDADD    SAVE IN ENDING ADDRESS
013E BD C363          JSR   PCRLF
0141 CE 0296          LDX   #HEDMSG   PRINT HEADING
0144 BD E07E          JSR   PDATA

0147 7F 02E7  LOOP    CLR   ERR       CLEAR PREVIOUS ERROR CONDITION
014A CE 0000          LDX   #0        ZERO SECTOR HEADER LOCATIONS
014D 86 0E            LDA A #$0E
014F 5F       LOP1    CLR B
0150 E7 00            STA B 0,X
0152 08               INX
0153 4A               DEC A
0154 26 F9            BNE   LOP1
0156 FE 02E1  LOP3    LDX   CURADD
0159 DF 01            STX   TRKSEC
015B B6 02E0          LDA A DRVTMP
015E 97 00            STA A DRV
0160 CE 02EB          LDX   #BUFFER
0163 DF 16            STX   TW        SETUP BUFFER AS TARGET ADDRESS
0165 BD C00C          JSR   RDSEC     READ THE SECTOR
0168 24 11            BCC   LOP5      BRANCH IF NO ERRORS
016A 81 05            CMP A #5        READ ERROR?
016C 27 0A            BEQ   LOP4      BR IF READ ERROR
016E 81 03            CMP A #3        EMPTY SECTOR?
0170 27 06            BEQ   LOP4      BR IF EMPTY SECTOR
0172 BD C01E          JSR   TYPERR    FATAL ERROR, ANNOUNCE AND TERMINATE
0175 7E E0E3          JMP   MON

0178 B7 02E7  LOP4    STA A ERR       SAVE ERRORS (3 AND 5) FOR LATER
017B BD C363  LOP5    JSR   PCRLF
017E BD 0201          JSR   PRSEC     PRINT OUT CURRENT SECTOR NUMBER
0181 CE 0001          LDX   #TRKSEC   PRINT OUT CURRENT SECTOR IN DTS FORMAT
0184 BD E0C8          JSR   OUT4HS
0187 FF 02E5          STX   XTEMP     SAVE HEADER ADDRESS
018A DE 03            LDX   BAKLNK    LOAD BACKLINK POINTER IN X
018C DF 01            STX   TRKSEC    PRINT OUT BACKLINK POINTER
018E BD 0201          JSR   PRSEC
0191 DE 05            LDX   FWDLNK    LOAD FORWARD LINK POINTER IN X
0193 DF 01            STX   TRKSEC    PRINT OUT FORWARD LINK POINTER
0195 BD 0201          JSR   PRSEC
0198 FE 02E5          LDX   XTEMP     RESTORE HEADER ADDRESS
019B 08               INX             INCREMENT X PAST BP AND FP
019C 08               INX
019D 08               INX
019E 08               INX
```

```
019F BD E0CA          JSR   OUT2HS    PRINT BYTE COUNT
01A2 BD E0C8          JSR   OUT4HS    PRINT MEMORY ADDRESS
01A5 BD E0CA          JSR   OUT2HS    PRINT FILE TYPE
01A8 08               INX             SKIP CRC
01A9 08               INX
01AA BD E0C8          JSR   OUT4HS    PRINT POSTAMBLE

01AD B6 02E7          LDA A ERR       GET ERROR CONDITION (IF ANY)
01B0 27 11            BEQ   LOP10     BRANCH IF NO ERROR
01B2 81 05            CMP A #5        CHECK FOR READ ERROR
01B4 26 05            BNE   LOP6      BR IF NOT
01B6 CE 02BB          LDX   #BADMSG   PRINT "BAD DATA"
01B9 20 19            BRA   LOP8
01BB CE 02C5  LOP6    LDX   #EMTMSG   PRINT "EMPTY SECTOR"
01BE BD E07E  LOP7    JSR   PDATA
01C1 20 1B            BRA   LOP13
01C3 DE 03    LOP10   LDX   BAKLNK    IF BACK POINTER =0, ANNOUNCE START
01C5 26 06            BNE   LOP11
01C7 CE 02D3          LDX   #START    PRINT "START"
01CA BD E07E          JSR   PDATA
01CD DE 05    LOP11   LDX   FWDLNK    IF FORWARD POINTER = 0, PRINT "END"
01CF 26 06            BNE   LOP12
01D1 CE 02DA          LDX   #END      PRINT "END"
01D4 BD E07E  LOP8    JSR   PDATA
01D7 7D 02DF  LOP12   TST   DATFLG
01DA 27 02            BEQ   LOP13     BR IF HEADER ONLY
01DC 8D 29            BSR   PRTDAT    PRINT DATA MAP
01DE FE 02E1  LOP13   LDX   CURADD
01E1 BC 02E3          CPX   ENDADD
01E4 26 03            BNE   LOP14
01E6 7E E0E3          JMP   MON
01E9 DF 05    LOP14   STX   FWDLNK
01EB BD C021          JSR   FWDCAL
01EE DE 05            LDX   FWDLNK
01F0 FF 02E1          STX   CURADD
01F3 7D 02DF          TST   DATFLG
01F6 27 06            BEQ   LOP15
01F8 CE 0296          LDX   #HEDMSG
01FB BD E07E          JSR   PDATA
01FE 7E 0147  LOP15   JMP   LOOP

0201 BD C324  PRSEC   JSR   PRTSEC
0204 7E E0CC          JMP   OUTS

0207 CE 02EB  PRTDAT  LDX   #BUFFER
020A FF 02E9          STX   BUFPTR
020D BD C363          JSR   PCRLF
0210 BD C363          JSR   PCRLF
0213 7F 02E8          CLR   LINHDR
0216 CE 02E8  PRT1    LDX   #LINHDR
0219 BD E0CA          JSR   OUT2HS
021C BD E0CC          JSR   OUTS
021F FE 02E9          LDX   BUFPTR
0222 C6 10            LDA B #COUNT
0224 BD E0CA  PRT2    JSR   OUT2HS
0227 5A               DEC B
```

```
0228 26 FA            BNE   PRT2
022A BD E0CC          JSR   OUTS
022D FE 02E9          LDX   BUFPTR
0230 C6 10            LDA B #COUNT
0232 A6 00    PRT3    LDA A 0,X
0234 84 7F            AND A #$7F      MASK PARITY BIT
0236 81 7F            CMP A #$7F
0238 27 04            BEQ   PRT4
023A 81 20            CMP A #$20
023C 24 02            BHS   PRT5
023E 86 2E    PRT4    LDA A #'.
0240 BD E1D1  PRT5    JSR   OUTCH
0243 08               INX
0244 5A               DEC B
0245 26 EB            BNE   PRT3
0247 FF 02E9          STX   BUFPTR
024A BD C363          JSR   PCRLF
024D C6 10            LDA B #COUNT
024F FB 02E8          ADD B LINHDR
0252 F7 02E8          STA B LINHDR
0255 24 BF            BCC   PRT1
0257 39               RTS

                      * MESSAGES
0258 44       MAPMSG  FCC   /DISK MAP-HEADER ONLY? (Y OR N) /
0259 49 53
025B 4B 20
025D 4D 41
025F 50 2D
0261 48 45
0263 41 44
0265 45 52
0267 20 4F
0269 4E 4C
026B 59 3F
026D 20 28
026F 59 20
0271 4F 52
0273 20 4E
0275 29 20
0277 04               FCB   4
0278 46       STMSSG  FCC   /FIRST SECTOR? /
0279 49 52
027B 53 54
027D 20 53
027F 45 43
0281 54 4F
0283 52 3F
0285 20
0286 04               FCB   4
0287 20       LSTMSG  FCC   / LAST SECTOR? /
0288 4C 41
028A 53 54
028C 20 53
028E 45 43
0290 54 4F
```

```
0292 52 3F
0294 20
0295 04                 FCB   4
0296 0D           HEDMSG FCB   $0D,$0A,0,0
0297 0A 00
0299 00
029A 53                 FCC   /SEC DTS  BP  FP  BC ADDR TY POST/
029B 45 43
029D 20 44
029F 54 53
02A1 20 20
02A3 42 50
02A5 20 20
02A7 46 50
02A9 20 20
02AB 42 43
02AD 20 41
02AF 44 44
02B1 52 20
02B3 54 59
02B5 20 50
02B7 4F 53
02B9 54
02BA 04                 FCB   4
02BB 42           BADMSG FCC   /BAD DATA /
02BC 41 44
02BE 20 44
02C0 41 54
02C2 41 20
02C4 04                 FCB   4
02C5 45           EMTMSG FCC   /EMPTY SECTOR /
02C6 4D 50
02C8 54 59
02CA 20 53
02CC 45 43
02CE 54 4F
02D0 52 20
02D2 04                 FCB   4
02D3 53           START  FCC   /START /
02D4 54 41
02D6 52 54
02D8 20
02D9 04                 FCB   4
02DA 45           END    FCC   /END /
02DB 4E 44
02DD 20
02DE 04                 FCB   4

02DF              DATFLG RMB   1
02E0              DRVTMP RMB   1
02E1              CURADD RMB   2
02E3              ENDADD RMB   2
02E5              XTEMP  RMB   2
02E7              ERR    RMB   1
02E8              LINHDR RMB   1
02E9              BUFPTR RMB   2
```

```
02EB            BUFFER RMB   256
      00  ERROR(S) DETECTED
```

PERCOM DATA CO.

TECHNICAL MEMO

TM-LFD-400-13
LFD-400 FLOPPY DISK SYSTEM
August 14, 1978

SUBJECT: A memory test to determine if a disk read error (error 5) is caused by bad memory.

Unlike other disk operating systems, MINIDOS (TM) performs an error check following a disk read operation over what is actually stored in memory rather than the data coming from the disk. For this reason, read errors (error #5) may occur if there is a fault in the mainframe RAM. The memory tests MEMCON and ROBIT are worthless in finding anything other than dead data bits or lower address lines. CDAT and SUMTEST are good but CDAT has two drawbacks. One is the pattern used is fixed - it would be desirable to at least test the compliment pattern as well. The other is the long time to test a 4K block of RAM with just one pattern. The Galloping Memory Test described below addresses these problems and introduces additional memory activity for a more complete test.

The Galloping Memory Test checks for the influencing of the contents of cells in a RAM chip when storing in another cell. The program itself is loaded in the first 256 bytes of RAM (0020-00FF) and can be run with either MIKBUG (TM) or SWTBUG (TM). The test asks for a START address and an END address. The END address must be greater than the START address. 512 byte blocks are tested and each block begins 256 bytes from the last, providing overlapping testing. This is not as thorough as CDAT in checking 1K or 4K chips but does check memory in the manner the disk is using it. Each block is tested with a background of 00 and FF.

The program executes an SWI when a error is detected. Accumulator B contains the background pattern, the X register contains the address at fault and Accumulator A contains the pattern read from that address if the address is a background address. If it was the base address, Accumulator A will contain the compliment of the contents of that address. In either case, A should have been equal to B and the exclusive or of the two will point to the bit(s) in error.

After each 512 byte block is tested, a "+" is printed. When the end of the test is reached, a "/" is printed and the program will prompt the user to enter new START and END addresses. If location location labeled "BUMP" is changed from a CMPB to an INCB, all 256 possible backgrounds will be tried, however the execution times will increase by a factor of 128.

EXECUTION TIMES	SIZE	TIME
	256	4 sec.
(two background	1K	46 sec.
patterns)	4K	3 min. 36 sec.

```
                        NAM   GALLOPING MEMTST VER 1.1
                    * SWTBUG ROUTINES USED
    (E0E3)          MON   EQU   $E0E3     MONITOR RETURN ADDRESS
    (E047)          BADDR EQU   $E047     ROUTINE TO GET 4 HEX DIGITS
    (E07E)          PDATA1 EQU  $E07E     ROUTINE TO PRINT MSG
    (E1D1)          OUTCH EQU   $E1D1     CHARACTER OUTPUT ROUTINE
    (C363)          PCRLF EQU   $C363     PRINT CR-LF
                    * REGISTERS
    (0020)          I     EQU   $0020     ADDRESS OF FOREGROUND
    (0022)          J     EQU   $0022     ADDRESS OF BACKGROUND
    (0024)          BEGADD EQU  $0024     FIRST ADDRESS IN TEST BLOCK
    (0026)          ENDADD EQU  $0026     LAST ADDRESS IN TEST BLOCK
    (0028)          BEG   EQU   $0028     FIRST ADDRESS TO BE TESTED
    (002A)          END   EQU   $002A     LAST ADDRESS TO BE TESTED
    (0030)                ORG   $0030
0030 CE 00C4    START   LDX   #SM        ASK FOR START ADDRESS
0033 BD E07E            JSR   PDATA1
0036 BD E047            JSR   BADDR     GET START ADDRESS
0039 DF 28              STX   BEG       MOVE TO START ADDRESS
003B DF 24              STX   BEGADD    ALSO INIT BEGIN ADDRESS
003D CE 00D1            LDX   #EM
0040 BD E07E            JSR   PDATA1    ASK FOR END ADDRESS
0043 BD E047            JSR   BADDR
0046 DF 2A              STX   END
0048 BD C363            JSR   PCRLF     PRINT CR-LF
004B 96 25      NEWBLK  LDA A BEGADD+1
004D 8B FF              ADD A #$FF      ENDADD = BEGADD + 511
004F 97 27              STA A ENDADD+1
0051 96 24              LDA A BEGADD
0053 89 01              ADC A #$01
0055 97 26              STA A ENDADD
0057 91 2A              CMP A END       IS ENDADD > END?
0059 25 10              BLO   NXTBLK
005B 26 06              BNE   SHORT
005D 96 27              LDA A ENDADD+1
005F 91 2B              CMP A END+1
0061 25 08              BLO   NXTBLK
0063 96 2A      SHORT   LDA A END       IF SO, SET TO END
0065 97 26              STA A ENDADD
0067 96 2B              LDA A END+1
0069 97 27              STA A ENDADD+1
006B 8D 1D      NXTBLK  BSR   GALLOP    DO NEXT 512 BLOCK
006D 86 2B              LDA A #$2B      "+"
006F BD E1D1            JSR   OUTCH     OUTPUT A + EVERY BLOCK
0072 96 24              LDA A BEGADD
0074 4C                 INC A           ADD 256 TO BEGADD
0075 97 24              STA A BEGADD
0077 91 2A              CMP A END       IS BEGINNING > END?
0079 25 D0              BLO   NEWBLK
007B 26 06              BNE   DONE
007D 96 25              LDA A BEGADD+1
007F 91 2B              CMP A END+1
0081 25 C8              BLO   NEWBLK
0083 86 2F      DONE    LDA A #$2F      PRINT / FOR PASS ALL
0085 BD E1D1            JSR   OUTCH
```

```
0088 20 A6            BRA   START     LOOP ON ASKING FOR NEW ADDRESS LIMITS
008A 5F       GALLOP  CLR B           INITIALIZE BACKGROUND
008B DE 24    NEWBG   LDX   BEGADD    INIT MEMORY TO BACKGROUND PATTERN
008D E7 00    INIT    STA B 0,X
008F 9C 26            CPX   ENDADD
0091 27 03            BEQ   INITX
0093 08               INX
0094 20 F7            BRA   INIT
0096 DE 24    INITX   LDX   BEGADD    INITIALIZE I
0098 DF 20    NEWI    STX   I         SAVE NEW I ADDRESS
009A 17               TBA             GET BACKGROUND
009B 43               COM A
009C A7 00            STA A 0,X       SET [I] TO COMPL. OF BACKGROUND
009E 08       NEWJ    INX
009F DF 22            STX   J         SAVE NEW J ADDRESS
00A1 DE 20            LDX   I
00A3 A6 00    SKIP    LDA A 0,X       FETCH [I]
00A5 43               COM A
00A6 11               CBA             MAKE SURE I STILL COMPL. OF BAKGND
00A7 26 1A            BNE   ERROR
00A9 DE 22            LDX   J
00AB A6 00            LDA A 0,X
00AD 11               CBA             CHECK FOR [J] = BACKGROUND
00AE 26 13            BNE   ERROR
00B0 9C 26            CPX   ENDADD    IS J = END ADDRESS?
00B2 26 EA            BNE   NEWJ
00B4 DE 20            LDX   I
00B6 08               INX
00B7 9C 26            CPX   ENDADD    IS I = END ADDRESS?
00B9 26 DD            BNE   NEWI
00BB C1 FF            CMP B #$FF      IS BACKGROUND = FF?
00BD 27 03            BEQ   PASS
00BF 53       BUMP    COM B           LONGER TEST IF THIS IS INCB
00C0 20 C9            BRA   NEWBG
00C2 39       PASS    RTS
00C3 3F       ERROR   SWI
00C4 0D 0A    SM      FDB   $0D0A
00C6 00 00            FDB   $0000
00C8 53               FCC   /START = /
00C9 54 41
00CB 52 54
00CD 20 3D
00CF 20
00D0 04               FCB   4
00D1 20       EM      FCC   /  END = /
00D2 20 45
00D4 4E 44
00D6 20 3D
00D8 20
00D9 04               FCB   4
                      END
      00  ERROR(S) DETECTED
```

```
                        NAM   PRINTOUT  VER 1.1
                      * COPYRIGHT (C) 1978 PERCOM DATA CO. INC.
                      * ALL RIGHTS RESERVED
                      * WRITTEN BY H.A. MAUCH
                      * REVISED AUG 22, 1978
                      ****************************************
                      * THIS PROGRAM IS USED TO PRINT OUT THE  CONTENTS
                      * OF AN ASCII FILE FROM THE PERCOM LFD-400 DISK
                      * TO THE SYSTEM PRINTER. THE PROGRAM IS DESIGNED
                      * TO `PAGENATE` THE PRINTOUT AND SUPPRESS
                      * `GARBAGE` BYTES WHICH MAY BE INSERTED BY THE
                      * EDITOR AT THE BEGINNING OF EACH LINE.
                      *
                      * SUFFICIENT SPACE HAS BEEN PROVIDED FOR PRINTER
                      * INITIALIZATION (INTPTR) AND PRINTER OUTPUT
                      * (PRINT) ROUTINES TO ADAPT THE PROGRAM TO A
                      * VARIETY OF PRINTER INTERFACES.  PROGRAM ENTRY
                      * ADDRESS IS AT `LOAD`.  THIS PROGRAM IS RELOCAT-
                      * ABLE AND MAY BE LOCATED ANYWHERE IN MEMORY.
                      * REQUIRES MINIDOS VERSION 1.2 1.4
                      ****************************************
   (0001)             DTKSEC EQU   $01
   (0005)             FWDLNK EQU   $05
   (0007)             BYTCNT EQU   $07
   (0016)             TW     EQU   $16
   (00FF)             EOF    EQU   $FF
   (000A)             LF     EQU   $0A
   (000D)             CR     EQU   $0D
   (C006)             GTKO   EQU   $C006
   (C00C)             RDSEC  EQU   $C00C
   (C012)             DRIVE  EQU   $C012
   (C015)             MOTOR  EQU   $C015
   (C353)             TYPERR EQU   $C353
   (C363)             PCRLF  EQU   $C363
   (C36B)             CVTDTS EQU   $C36B
   (E07E)             PDATA  EQU   $E07E
   (E047)             BADDR  EQU   $E047
   (A04A)                    ORG   $A04A
A04A                  DSKFIL RMB   2
A04C                  DSKPTR RMB   2
A04E                  BYTECT RMB   1
A04F                  BUFCNT RMB   1
A050                  BUFADD RMB   2
A052                  BUFPTR RMB   2
A054                  ADDRES RMB   2
A056                  XTEMP1 RMB   2
A058                  LINCNT RMB   1

   (2000)                    ORG   $2000

2000 20 21                BRA   LOAD
2002 39               INTPTR RTS                ROOM TO INITIALIZE PRINTER
2003                      RMB   32
2023 8E A07F          LOAD   LDS   #$A07F      SET STACK
2026 8D DA                BSR   INTPTR
```

```
2028 8D 00          BSR   HERE      THIS SEQUENCE PERMITS THIS
202A 30       HERE  TSX             TO BE LOCATED ANYWHERE
202B 01             NOP
202C A6 01          LDA A 1,X
202E 8B EC          ADD A #PROMPT-HERE
2030 B7 A051        STA A BUFADD+1
2033 A6 00          LDA A 0,X
2035 89 00          ADC A #0
2037 B7 A050        STA A BUFADD
203A BD C363        JSR   PCRLF
203D FE A050        LDX   BUFADD
2040 BD E07E        JSR   PDATA
2043 08             INX
2044 FF A050        STX   BUFADD
2047 BD E047        JSR   BADDR     GET FILE
204A BD C36B        JSR   CVTDTS
204D 25 42          BCS   ERR
204F DE 01          LDX   DTKSEC
2051 FF A04A        STX   DSKFIL
2054 FF A04C        STX   DSKPTR
2057 DF 01    INTDSK STX  DTKSEC
2059 BD C012        JSR   DRIVE
205C BD C015        JSR   MOTOR
205F 25 30          BCS   ERR
2061 BD C006        JSR   GTKO
2064 86 01          LDA A #1
2066 B7 A04F        STA A BUFCNT
2069 8D 2B          BSR   CRLF

206B 7F A058  PRINT1 CLR  LINCNT
206E C6 03    PRINT2 LDA B #3       GARBAGE CHARACTER COUNT
2070 8D 5C    PRINT3 BSR  GETCHR    BYPASS GARBAGE
2072 27 AF           BEQ  LOAD      END OF FILE
2074 5A              DEC B
2075 26 F9           BNE  PRINT3    NOT END OF GARBAGE YET
2077 8D 55    PRINT4 BSR  GETCHR
2079 27 A8           BEQ  LOAD      END OF FILE
207B 81 0D           CMP A #CR
207D 27 04           BEQ  ENDLIN    END OF LINE
207F 8D 2A           BSR  PRINT     PRINT CHARACTER
2081 20 F4           BRA  PRINT4    GET NEXT CHARACTER
2083 8D 11    ENDLIN BSR  CRLF
2085 81 36           CMP A #54
2087 26 E5           BNE  PRINT2    NOT END OF PAGE YET
2089 8D 0B    PRINT5 BSR  CRLF
208B 81 42           CMP A #66
208D 26 FA           BNE  PRINT5    NOT END OF MARGIN YET
208F 20 DA           BRA  PRINT1    START NEXT PAGE

2091 BD C353  ERR    JSR  TYPERR    ERROR TRAP
2094 20 8D           BRA  LOAD      UNSTRUCTURED RETURN

              *PRINT CARRIAGE RETURN - LINE FEED
2096 86 0D    CRLF   LDA A #CR
2098 8D 11           BSR  PRINT
209A 86 0A           LDA A #LF
```

```
209C 8D 0D            BSR   PRINT
209E 4F               CLR A
209F 8D 0A            BSR   PRINT
20A1 4F               CLR A
20A2 8D 07            BSR   PRINT
20A4 7C A058          INC   LINCNT
20A7 B6 A058          LDA A LINCNT
20AA 39               RTS
20AB 7E E1D1 PRINT    JMP   $E1D1     ROOM FOR PRINTER DRIVER
20AE                  RMB   32

20CE FF A056 GETCHR   STX   XTEMP1    SAVE INDEX
20D1 37               PSH B           SAVE B
20D2 7A A04F          DEC   BUFCNT    BUMP DISK BUFFER COUNTER
20D5 26 16            BNE   GETC1     BRANCH IF NOT EMPTY
20D7 86 FF            LDA A #EOF      END OF FILE DEFAULT
20D9 FE A04C          LDX   DSKPTR
20DC 27 18            BEQ   GETC2
20DE 8D 1E            BSR   GTSCTR    GET NEXT SECTOR
20E0 25 AF            BCS   ERR       BRANCH IF ERROR
20E2 FE A050          LDX   BUFADD    POINT TO DISK BUFFER
20E5 FF A052          STX   BUFPTR
20E8 96 07            LDA A BYTCNT    GET BYTE COUNT
20EA B7 A04F          STA A BUFCNT
20ED FE A052 GETC1    LDX   BUFPTR    GET DISK BUF PTR
20F0 A6 00            LDA A 0,X       GET CHARACTER
20F2 08               INX
20F3 FF A052          STX   BUFPTR
20F6 FE A056 GETC2    LDX   XTEMP1
20F9 33               PUL B           RESTORE B
20FA 81 FF            CMP A #EOF
20FC 0C               CLC
20FD 39               RTS

                      * GET A SECTOR OF DATA FROM DISC
20FE DF 01    GTSCTR  STX   DTKSEC    SET UP DISK HEADER
2100 FE A050          LDX   BUFADD    SET UP TARGET ADD
2103 DF 16            STX   TW
2105 BD C00C          JSR   RDSEC     READ THE SECTOR
2108 25 0B            BCS   GTS1
210A DE 05            LDX   FWDLNK
210C FF A04C          STX   DSKPTR    PUT IN DISK PTR
210F 96 07            LDA A BYTCNT    PICK UP BYTE COUNT
2111 B7 A04F          STA A BUFCNT    PUT IN DISK BUF CNTR
2114 0C               CLC
2115 39       GTS1    RTS

2116 46       PROMPT  FCC   'FILE? '
2117 49 4C
2119 45 3F
211B 20
211C 04               FCB   $04
211D          BUFFER  RMB   256
                      END
      00  ERROR(S) DETECTED
```

PERCOM DATA CO.

TECHNICAL MEMO

TM-LFD-400-17
LFD-400 FLOPPY DISK
OCTOBER 12, 1978

SUBJECT: LFD-400 CONTROLLER CARD THEORY OF OPERATION

The LFD-400 Controller Card is designed to operate as a memory mapped I/O device on the SS-50 Bus. It contains the hardware required by a CPU software intensive driver to interface to 10 sector hard-sectored Mini-Diskette drives.

ADDRESS DECODING AND SS-50 BUS INTERFACE

The LFD-400 Mini-Disk Controller occupies a 4K block of mainframe memory beginning at address $C000. The first 3K of this block is assigned to 3 1K ROM sockets. The last 1K is assigned to the disk controller I/O devices.

ICs B10 and B11 decode the address and valid address timing. One section of A9 is enabled by the output of B11-8 and provides the chip selects for the three 2708 ROMs and the enable for the I/O decoders. The output of B11-8 is also combined with the R/W line to enable the bus drivers when the controller is addressed during a read cycle.

ICs B8 and B9 further decode the address lines into 1 of 7 I/O read functions and 1 of 5 I/O write functions. The functions are mapped into memory locations $CC00 through $CCFF. Since only the four least significant address lines are used in the decode, the same set of functions repeat every 16 locations in the memory address space from $CC00 to $CCFF. The MINIDOS listing identifies the specific function address assignments.

ICs B16 and B17 are Data Bus transceivers which buffer the data lines.

SECTOR-INDEX LOGIC

The LFD-400 Controller Card provides index and sector timing information via the sector-index logic. A hard sectored diskette contains 10 evenly spaced sector holes with an additional index hole in the middle of sector 0. The disk drive generates a pulse each time a hole is detected. By using a one-shot set for approximately 70% of the time it takes the diskette to rotate from one sector to the next, the sector-index logic on the controller card separates the index pulse from the sector pulses.

ICs A1 and A2 separate the sector and index pulses. The D Flipflops in A2 are clocked by the combined sector-index pulse from the disk drive. When A2-11 is pulsed, A2-9 is clocked high. This transition triggers one-shot A1.

The output of the one-shot is feed back to the D input of the first flipflop to inhibit retriggering of the one-shot when an index pulse is received. Thus the one-shot triggers at the beginning of a sector and times out about 70% of the way through the sector.

When the one-shot times out (approximately 14 milliseconds) the low-to-high transition at A1-6 increments sector counter A10 and resets A2-9. If another pulse is received from the disk drive before the one-shot times out, A2-6 is clocked low, causing A10 to be reset to sector count 0. This signifies the occurrence of an INDEX pulse. Resetting the sector counter with the Index pulse guarantees the sector counter will always be synchronized with the physical sectors. The count state of sector counter (A10) can be read at memory address $CC02. The separated index and sector pulses are available as bits 5 and 4 in memory location $CC03.

DRIVE CONTROLS

The drive control section controls the drive motor, selects the desired drive, and steps the R/W head from track to track. Also status bits indicating the state of the drive may be read.

The drive motor is enabled by a one-shot (A8) which times out in two to five seconds. The CPU can trigger this one-shot by reading memory location $CC05. Since the one-shot is retriggerable, the CPU can keep the motor running by successive reads of memory location $CC05. The output of the one-shot is available in the drive status word as bit 2.

The CPU can write into a register at $CC03 which sets the drive select, the step direction, and the step line to the disk drive. The drive select is written into bits 7 and 6 as a binary code for drives 1, 2, or 3. This two bit code may be read back by reading memory location $CC03. Bits 7 and 6 will be a copy of what was written. A '1' written in the direction step bit (bit 4) will cause the drive to step in (towards track 34) when the step pulse is applied. A step pulse is applied by writing first a '1' then a '0' into bit 4 of memory location $CC03.

The CPU can read the drive status bits by reading memory location $CC03. Bits 7 and 6 are the drive select code as described above. Bit 5 is the index pulse and bit 4 is the sector pulse as described in the Sector-Index section. Bit 3 is the write gate flipflop. Bit 2 is the motor one-shot described earlier in this section. Bit 1 is the TRACK 00 sense switch on the selected drive. When this line is low, the drive is at track 0. Bit 0 is the Write Protect switch output from the selected drive. When this bit is a '0', the diskette is write protected.

A6 is the drive control register. A decoder in A9 decodes the 1 of 3 drive selects. A5 and A3 serve as line drivers for the signals sent to the disk drives. A12 gates the drive status bits to the data bus.

DISK DATA CONTROLLER (DDC)

The Data Data Controller (DDC) handles the protocol between the disk drive and the CPU. The Data Separator logic supplies the Disk Data Controller with separated serial clock and data. The Disk Data Controller in turn supplies the serial data to be written to the disk.

The Disk Data Controller has three input registers at address $CC00 thru $CC02 and two output registers at address $CC00 and $CC01. Address $CC04 is a single function control input to the Data Controller which 'restarts' the read data synchronization procedure. For more information about the Disk Data Controller, refer to the description attached to this memo.

DATA SEPARATOR

The Disk Data Separator logic handles all clocking of the Data Controller as well as read data and read clock separation. The data format used to record on the diskette is known as Bi-Phase or double frequency. A clock pulse is written on the diskette for every data bit. If the data bit is a '1', an additional pulse is written half-way between successive clock pulses.

Part of IC B1 forms an oscillator to provide a 10Mhz clock for the controller. ICs B3 and B5 form a divider chain to derive the data sample window. IC B2 synchronizes the incoming data pulses to the 10 Mhz clock. If B5-12 is high when B2-5 pulses, B6-6 sets data latch B7-9. If B5-12 is low when B2-5 pulses, data latch B7-9 is reset via B6-3 and the timing cycle of B5 is resynchronized. This condition occurs if the incoming pulse is a clock pulse. The Disk Data Controller uses the output of data latch B7-9 as its incoming serial data and the output of B5-12 as its incoming clock.

B7-5 is a one bit memory of the previous data bit which forces B5 to 'post compensate' its timing cycle. 'Post Compensation' is an adaptive process which diminishes the detrimental effects of a characteristic of high density magnetic data recording called 'bit-shifting'.

B4-12 and the 'divide by 2' section B3 are used to synchronize B5 to the clock pulses. If B5 goes through one complete timing cycle without a clock pulse appearing at B2-5, the next pulse from the drive is assumed to be a clock pulse. If B5 is not in a proper count state for a clock pulse is will be reset via B5-13.

During a write cycle, the serial data from the B12-6 is shaped by combining it with the write clock using sections of B4 and B6. The result is a single pulse (the clock pulse) if the data bit is a '0', and two pulses, the clock pulse and the data pulse, if the data bit is a '1'.

POWER SUPPLY

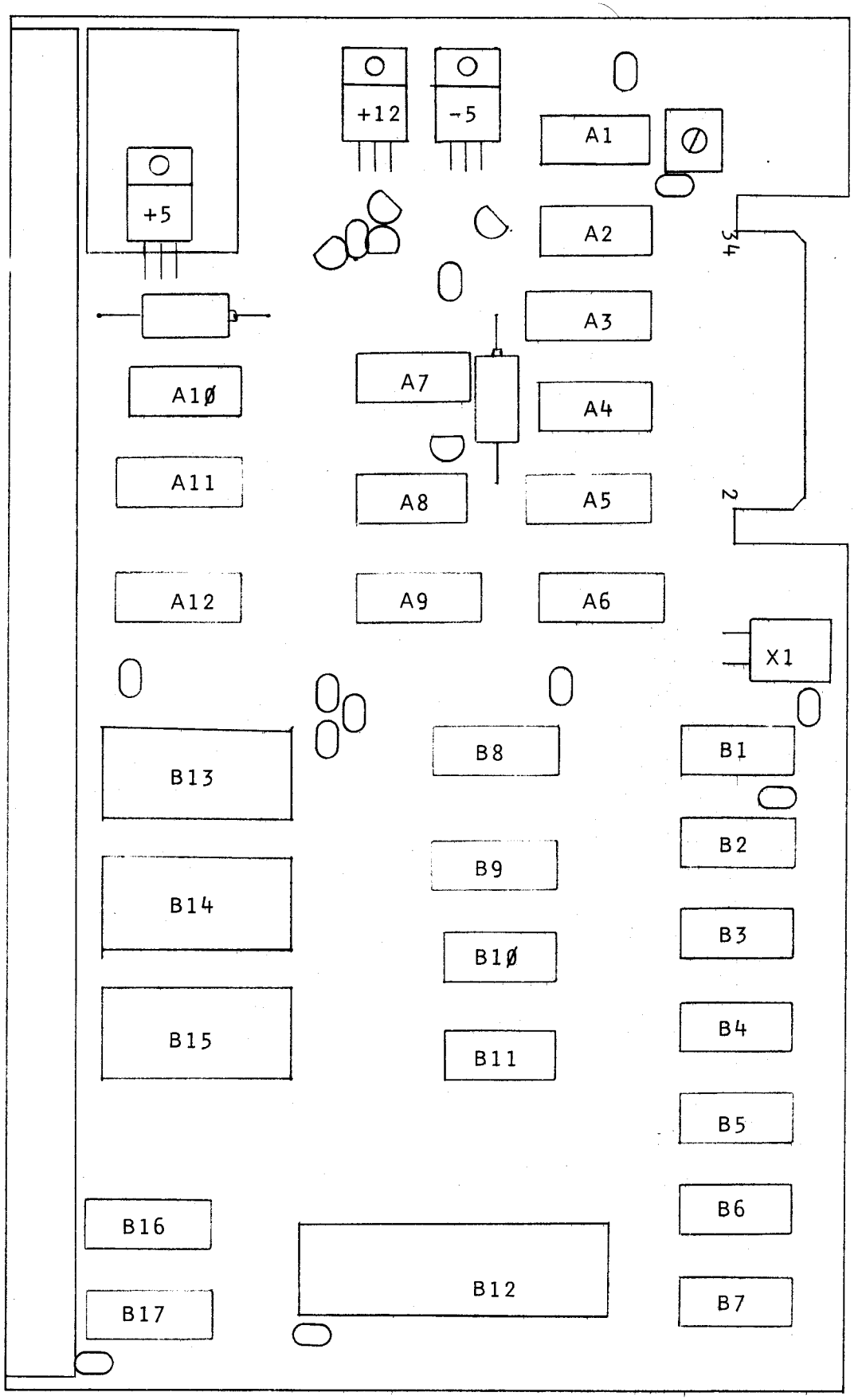
The +5 volt regulator is a conventional integrated circuit regulator. The -5 volt regulator is a simple 'zener follower' circuit with current limiting via transistor Qp and a 3.3 ohm resistor.

In many SWTP 6800 computers the +12 volt supply is not sufficient to properly drive 12 volt integrated circuit regulators. Consequently the +12 volt regulator on the LFD-400 controller is a specially designed circuit which will function with minimal regulator voltage drop. The 2 NPN transistors (Qn) form a differential amplifier which compares a resistor divided sample of the regulator output voltage to the regulated 5 volts on the base of the left transistor. If the output voltage is low, the transistor arrangement is such that current flow through the power transistor is increased. Transistor Qp and the 3.3 ohm resistor provide circuit current limit protection.

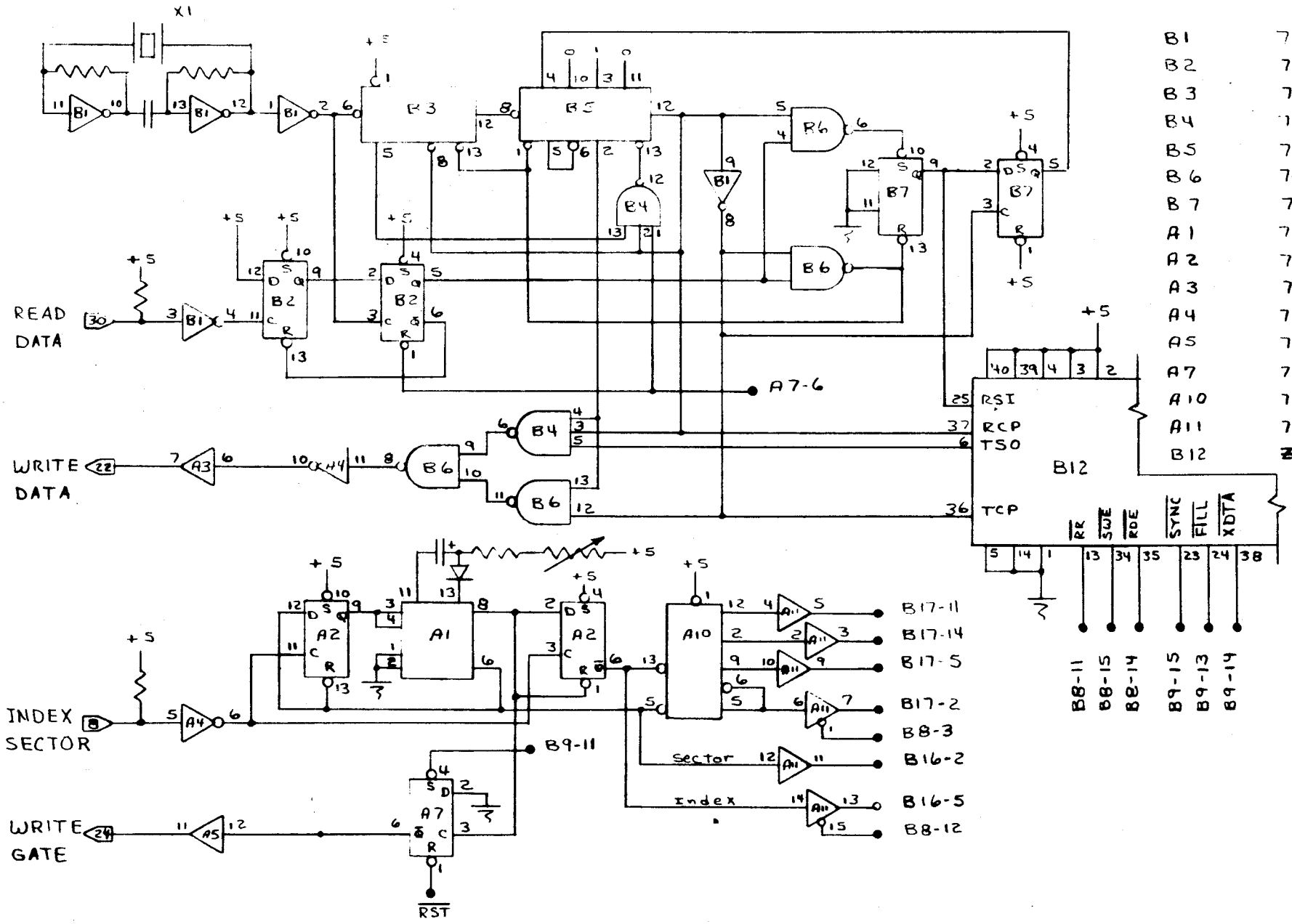
ADJUSTMENT

The LFD-400 has only one adjustment; the Sector-Index pulse separator one-shot.

1. Insert a 10 sector diskette in Disk Drive #1.
2. Short the positive (+) end of the large capacitor immediately above IC A7 to Ground with a clip lead. This permits the drive motor to run continuously.
3. Store a $40 in memory address $CC03 to select the Disk Drive.
4. Examine memory location $CC05 to start the Disk Drive motor.
5. Connect an oscilloscope to IC A1-8. Trigger Positive, Internal.
6. Adjust the Trim Pot immediately above IC A1 for a 14 millisecond positive pulse. The pulse should repeat every 20 milliseconds.

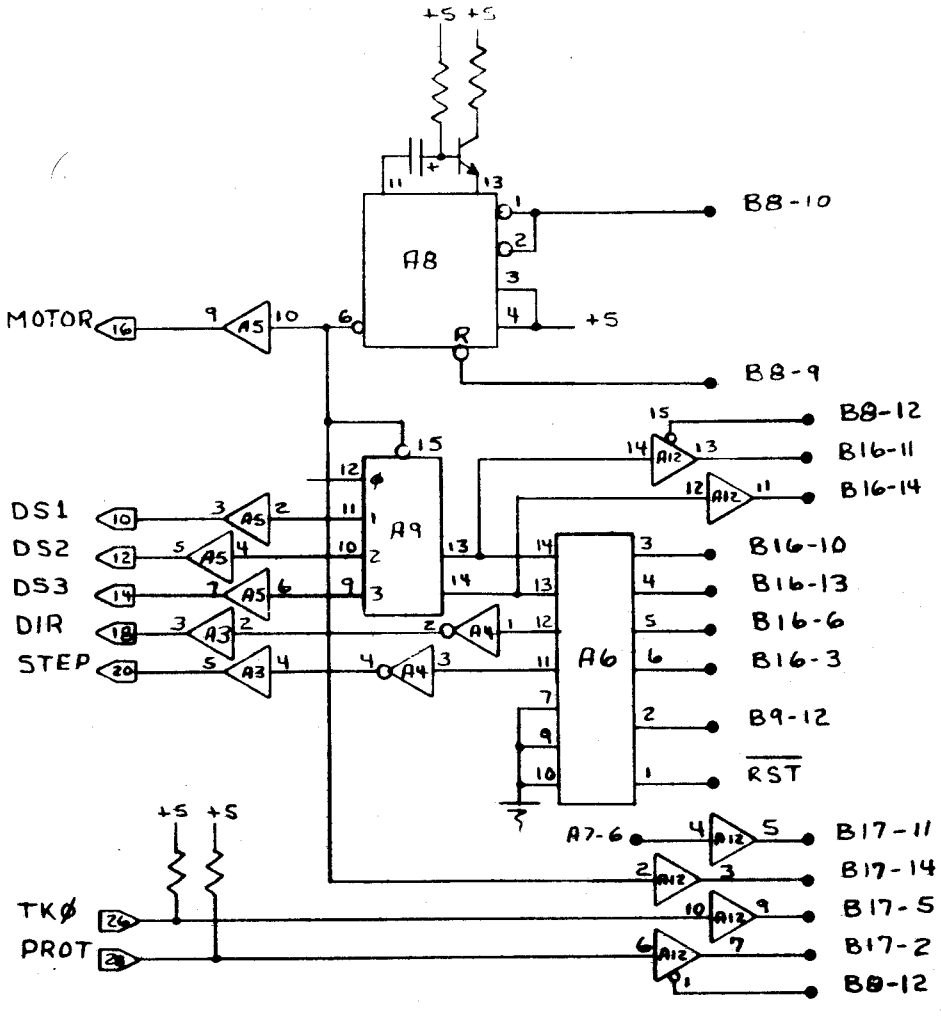

+12
-5
+5
A1
A2
A3
A4
A5
A6
A7
A8
A9
A10
A11
A12
X1
B1
B2
B3
B4
B5
B6
B7
B8
B9
B10
B11
B12
B13
B14
B15
B16
B17
34
2

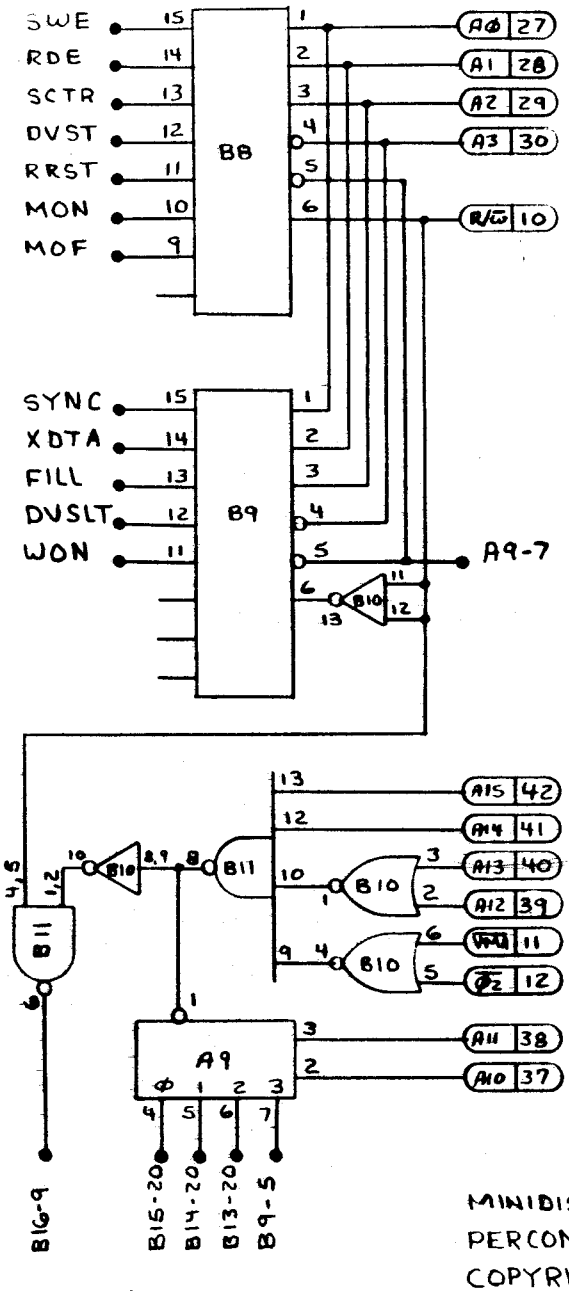
LFD-400B



B1	74LS04
B2	74LS74 7474
B3	74LS196
B4	74LS10 7410
B5	74LS197
B6	74LS00 MC8400
B7	74LS74 7474
A1	74LS122
A2	74LS74 7474
A3	74367
A4	74LS04
A5	74367
A7	74LS74 7474
A10	74LS196
A11	74367
B12	≠5023

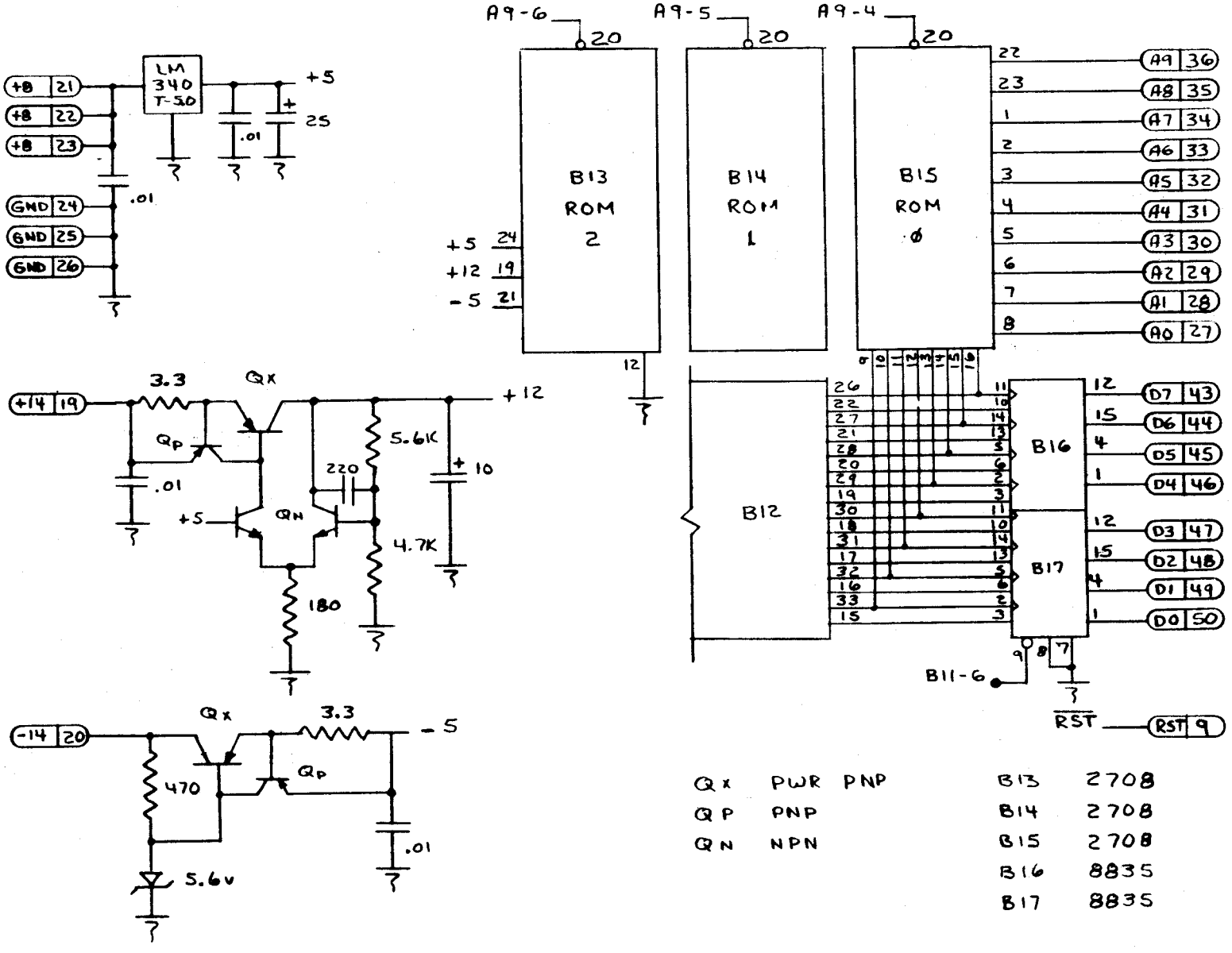
MINI DISC CONTROLLER
PERCOM DATA CO.
COPYRIGHT 1977
Sheet 1 of 3
REV C 10-10-77 HM





A4	74LS04
A5	74367
A6	74LS161
A8	74LS122
A9	74LS139
A12	74367
B8	74LS138
B9	74LS138
B10	74LS02
B11	74LS20

MINIDISC CONTROLLER
PERCOM DATA CO.
COPYRIGHT 1977
SHEET 2 of 3
REV C 10-10-77 HM



Qx	PWR PNP	B13	2708
QP	PNP	B14	2708
QN	NPN	B15	2708
		B16	8835
		B17	8835

MINIDISC CONTROLLER
PERCOM DATA CO.

SHEET 3 of 3
REV D 10-10-77 HM
5-24-78 HM

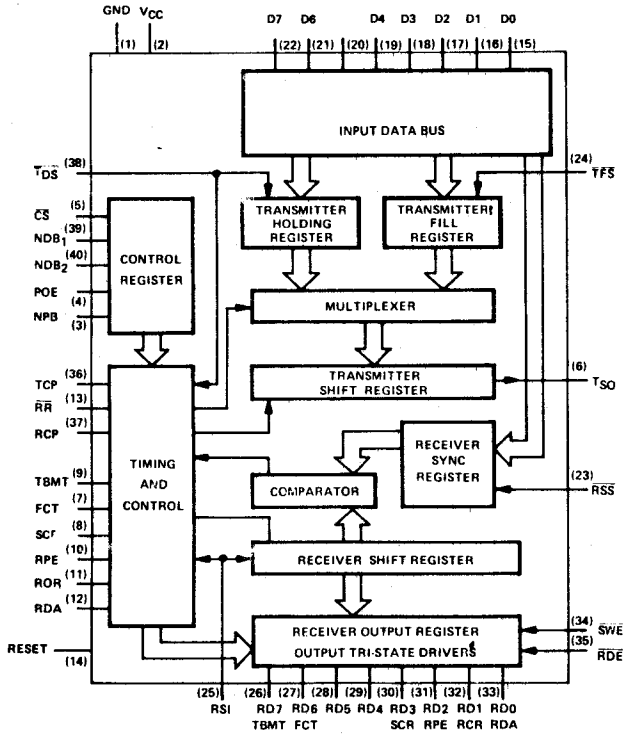
PERCOM DISK DATA CONTROLLER (DDC)
Z-5023

The Percom Disk Data Controller (DDC) is a single chip MOS/LSI device which performs the serial to parallel and parallel to serial conversion logic required to interface a byte-parallel processor to a bit-serial, synchronous Disk Data System.

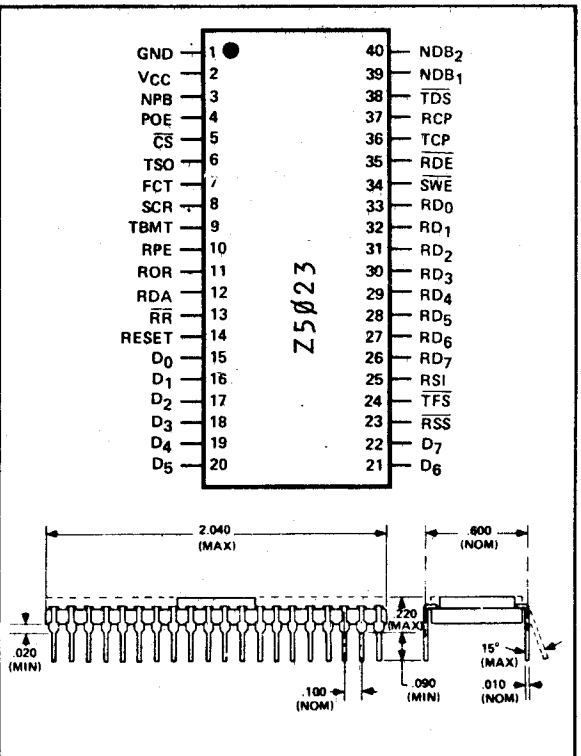
The DDC consists of separate receiver and transmitter sections with independent clocks, data lines and status. Common with the transmitter and receiver are word length and parity mode. Data is transmitted and received in a NRZ format at a rate equal to the respective input clock frequency.

Data messages are transmitted as a continuous character stream, bit synchronous with respect to a clock and character synchronous with respect to framing or "sync" characters initializing each message. The DDC receiver compares the contents of the internal Receiver SYNC Register with the incoming data stream in a bit transparent mode. When a compare is made, the receiver becomes character synchronous formatting a 5,6,7, or 8 bit character for output each character time. The receiver has an output buffer register allowing a full character time to transfer the data out. The receiver status outputs indicate received data available (RDA), receiver over-run (ROR), receive parity error (RPE), and sync character received (SCR). Status bits are available on individual output lines and can also be multiplexed onto the output data lines for bus organized systems. The data lines have tri-state outputs.

The DDC transmitter outputs 5,6,7, or 8 bit characters with correct parity at the transmitter serial output (TSO). The transmitter is buffered to allow a full character time to respond to a transmitter buffer empty (TBMT) request for data. Data is transmitted in a NRZ format changing on the positive transition of the transmitter clock (TCP). The character in the transmitter fill register is inserted into the data message if a data character is not loaded into the transmitter after a TBMT request.



BLOCK DIAGRAM



PIN/PACKAGE CONFIGURATION

Pin	Lable	Function
(1)	GND	Ground
(2)	V_{CC}	+5 VOLTS ±5%
(14)	RESET	MASTER RESET A V_{IH} initializes both the receiver and transmitter. The Transmitter Shift Register is set to output a character of all logic 1's. FCT is reset to V_{OL} and TBMT set to V_{OH} indicating the Transmitter Holding Register is empty. The receiver status is initialized to a V_{OL} on RPE, ROR, SCR, and RDA. The transmitter and receiver shift registers are reset to logic "0"s. The sync character detect logic is inhibited until a RR pulse is received.
(15)	D0	DATA INPUTS Data on the eight data lines is loaded into the Transmitter Holding Register by $\overline{TDS}$, the Transmitter Fill Register by $\overline{TFS}$, and the Receiver Sync Register by $\overline{RSS}$. Data is right jusitifed with the LSB at D0. For word lengths less than 8 bits, the unused inputs are ignored. Data is transmitted LSB first.
(16)	D1	
(17)	D2	
(18)	D3	
(19)	D4	
(20)	D5	
(21)	D6	
(22)	D7	
(38)	$\overline{TDS}$	TRANSMIT DATA STROBE A V_{IL} loads data on D0-D7 into the Transmitter Holding Register and resets TBMT to a V_{OL}.
(24)	$\overline{TFS}$	TRAMSMIT FILL STROBE A V_{IL} loads data on D0-D7 into the Transmitter Fill Register. The character in the Transmitter Fill Register is transmitted whenever a new character is not loaded in the allotted time after the TBMT is set to V_{OH}.
(23)	$\overline{RSS}$	RECEIVER SYNC STROBE A V_{IL} loads data on D0-D7 into the Receiver Sync Register. SCR is set to V_{OH} whenever data in the Receiver Shift Register compares with the character in the Receiver Sync Register.
(9)	TBMT	TRANSMIT BUFFER EMPTY A V_{OH} indicates the data in the Transmitter Holding Register has been transferred to the Transmitter Shift Register and new data may be loaded. TBMT is reset to V_{OL} by a V_{IL} on $\overline{TDS}$. A V_{IH} on RESET sets TBMT to a V_{OH}. TBMT is also multiplexed onto the RD7 output (26) when $\overline{SWE}$ is at V_{IL} and $\overline{RDE}$ is at V_{IH}.
(6)	TSO	TRANSMITTER SERIAL OUTPUT Data entered on D0-D7 are transmitted serially, least significant bit first, on TSO at a rate equal to the Transmit Clock frequency, TCP. Source of the data to the transmitter shift register is the Transmitter Holding Register or Transmitter Fill Register.
(36)	TCP	TRANSMIT CLOCK Data is transmitted on TSO at the frequency of the TCP input in a NRZ format. A new data bit is started on each negative to positive transition (V_{IL} to V_{IH}) of TCP.
(3)	NPB	NO PARITY BIT A V_{IH} eliminates generation of a parity bit in the transmitter and checking of parity in the receiver. With parity disabled, the RPE status bit is held at V_{OL}.
(4)	POE	PARITY ODD/EVEN A V_{IH} directs both the transmitter and receiver to operate with even parity. A V_{IL} forces odd parity operation. NPB must be VIL for parity to be enabled.
(5)	$\overline{CS}$	CONTROL STROBE A V_{IL} loads the control inputs NDB1, NDB2, POE, and NPB into the Control Register. For static operation, $\overline{CS}$ can be tied directly to ground.

Pin	Label	Function
(26) (27) (28) (29) (30) (31) (32) (33)	RD7 RD6 RD5 RD4 RD3 RD2 RD1 RD0	RECEIVED DATA OUTPUTS RDO-RD7 contain data from the Receiver Output Register or selective status conditions depending on the state of $\overline{SWE}$ and $\overline{RDE}$ per the following table:

(34)	(35)	(33)	(32)	(31)	(30)	(39)	(28)	(27)	(26)
$\overline{SWE}$	$\overline{RDE}$	RD0	RD1	RD2	RD3	RD4	RD5	RD6	RD7
V_{IL}	V_{IL}	X	X	X	X	X	X	X	X
V_{IL}	V_{IH}	RDA	ROR	RPE	SCR	V_{OL}	V_{OL}	FCT	TBMT
V_{IH}	V_{IL}	DB0	DB1	DB2	DB3	DB4	DB5	DB6	DB7
V_{IH}	V_{IH}	X	X	X	X	X	X	X	X

X Output is in the OFF or Tri-State condition
DB0 LSB of Receiver Output Register
DB7 MSB of Receiver Output Register
The two unused outputs are held at V_{OL} in the output status condition.

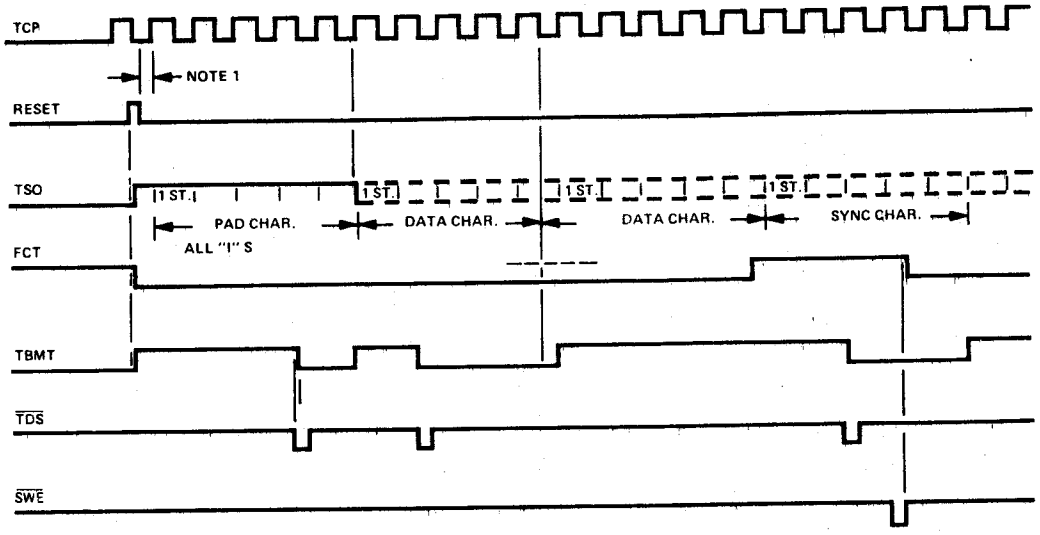
(35)	$\overline{RDE}$	RECEIVE DATA ENABLE A V_{IL} enables the data in the Receiver Output Register onto the output data lines RD0–RD7. The trailing edge (V_{IL} to V_{IH} transition) of $\overline{RDE}$ resets RDA to the V_{OL} condition.
(7)	FCT	FILL CHARACTER TRANSMITTED A V_{OH} on FCT indicates data from the Transmitter Fill Register has been transferred to the Transmitter Shift Register. FCT is reset to V_{OL} when data is transferred from the Transmitter Holding Register to the Transmitter Shift Register, or on the trailing edge (V_{IL} to V_{IH}) of the $\overline{SWE}$ pulse, or when RESET is V_{IH}. FCT is multiplexed onto the RD6 output (27) when $\overline{SWE}$ is at V_{IL} and $\overline{RDE}$ is at V_{IH}.
(25)	RSI	RECEIVER SERIAL INPUT Serial data is clocked into the Receiver Shift Register, least significant bit first, on RSI at a rate equal to the Receive Clock frequency RCP.
(37)	RCP	RECEIVE CLOCK Data is transferred from RSI input to the Receiver Shift Register at the frequency of the RCP input. Each data bit is entered on the positive to negative transition (V_{IH} to V_{IL}) of RCP.
(12)	RDA	RECEIVED DATA AVAILABLE A V_{OH} indicates a character has been transferred from the Receiver Shift Register to the Receiver Output Register. RDA is reset to V_{OL} on the trailing edge (V_{IL} to V_{IH} transition) of $\overline{RDE}$, by a V_{IL} on $\overline{RR}$ or a V_{IH} on RESET. RDA is multiplexed onto the RD0 output (33) when $\overline{SWE}$ is V_{IL} and $\overline{RDE}$ is V_{IH}.

Pin	Label	Function
(8)	SCR	SYNC CHARACTER RECEIVED A V_{OH} indicates the data in the Receiver Shift Register is identical to the data in the Receiver Sync Register. SCR is reset to a V_{OL} when the character in the Receiver Shift Register does not compare to the Receiver Sync Register, on the trailing edge (V_{IL} to V_{IH} transition) of $\overline{SWE}$, by a V_{IL} on $\overline{RR}$ or a V_{IH} on RESET. SCR is multiplexed onto the RD3 output (30) when $\overline{SWE}$ is a V_{IL} and $\overline{RDE}$ is V_{IH}.
(34)	$\overline{SWE}$	STATUS WORD ENABLE A V_{IL} enables the internal status conditions onto the output data lines RD0–RD7. The trailing edge of $\overline{SWE}$ pulse resets FCT, ROR, RPE, and SCR to V_{OL}.
(11)	ROR	RECEIVER OVERRUN A V_{OH} indicates data has been transferred from the Receiver Shift Register to the Receiver Output Register when RDA was still set to V_{OH}. The last data in the Output Register is lost. ROR is reset by a V_{IL} on $\overline{RDE}$ by the trailing edge (V_{IL} to V_{IH}) of $\overline{SWE}$, a V_{IL} on $\overline{RR}$ or a V_{IH} on RESET. ROR is multiplexed onto the RD1 output (32) when $\overline{SWE}$ is V_{IL} and $\overline{RDE}$ is V_{IH}.
(10)	RPE	RECEIVER PARITY ERROR A V_{OH} indicates the accumulated parity on the received character transferred to the Output Register does not agree with the parity selected by POE. RPE is reset with the next received character with correct parity, the trailing edge (V_{IL} to V_{IH}) of $\overline{SWE}$, a V_{IL} on $\overline{RR}$ or a V_{IH} on RESET. RPE is multiplexed onto the RD2 output (31) when $\overline{SWE}$ is V_{IL} and $\overline{RDE}$ is V_{IH}.
(13)	$\overline{RR}$	RECEIVER RESTART A V_{IL} resets the receiver section by clearing the status RDA, SCR, ROR, and RPE to V_{OL}. The trailing edge of $\overline{RR}$ (V_{IL} to V_{IH}) also puts the receiver in a bit transparent mode to search for a comparison, each bit time, between the contents of the Receiver Shift Register and the Receiver Sync Register. The number of data bits per character for the comparison is set by NDB1 and NDB2. After a compare is made SCR is set to V_{OH}, the sync character is transferred to the Receiver Output Register, and the receiver enters a word synchronous mode framing an input character each word time. NOTE: Parity is not checked on the first sync character but is enabled for every succeeding character.
(39)	NDB1	NUMBER DATA BITS The number of Data Bits per character are determined by NDB1 and NDB2. The number of data bits does not include the parity bit.

NDB2	NDB1	CHARACTER LENGTH
V_{IL}	V_{IL}	5 Bits
V_{IL}	V_{IH}	6 Bits
V_{IH}	V_{IL}	7 Bits
V_{IH}	V_{IH}	8 Bits

For character lengths less than 8 bits, unused inputs are ignored and unused outputs are held to V_{OL}. Data is always right justified with D0 and RD0 being the least significant bits.

TRANSMITTER TIMING DIAGRAM



NOTE 1 DATA TRANSMISSION WILL START ON THE FIRST LOW TO HIGH TRANSITION OF TCP AFTER RESET IS LOW. THE INITIAL RESET PULSE SHOULD NOT OCCUR UNTIL 100 MICROSECONDS AFTER POWER IS APPLIED.

RECEIVER TIMING DIAGRAM