

MOTOROLA Semiconductor Products Inc.

M68FD3601 - M68FD3604

EXORdisk

MOTOROLA FLOPPY DISK SYSTEM

USER'S GUIDE

The information in this manual has been carefully reviewed and is believed to be entirely reliable. However, no responsibility is assumed for inaccuracies. Furthermore, such information does not convey to the purchaser of the semiconductor devices described any license under the patent rights of Motorola Inc. or others.

The material in this manual is subject to change, and Motorola Inc. reserves the right to change specifications without notice.

EXORdisk, EXORciser, and EXbug are trademarks of Motorola Inc.

TABLE OF CONTENTS

| | | |
|--------------------|---------------------------------------------------|------------|
| SECTION 1: | UNPACKING AND INSTALLATION INSTRUCTION | 1-1 |
| 1-1 | Unpacking Instruction | 1-1 |
| 1-2 | EXORDisk Installation Instruction | 1-1 |
| 1-3 | EDOS Installation Instruction | 1-1 |
| SECTION 2: | SYSTEM ORGANIZATION | 2-1 |
| 2-1 | EDOS Software | 2-1 |
| 2-1.1 | EDOS Resident Driver | 2-1 |
| 2-1.2 | EDOS Executive | 2-1 |
| 2-1.3 | EDOS Editor | 2-1 |
| 2-1.4 | EDOS Assembler | 2-1 |
| 2-2 | EDOS Software Logic | 2-2 |
| SECTION 3: | EDOS OPERATION | 3-1 |
| 3-1 | EDOS Start-up Procedures | 3-1 |
| 3-2 | File Organization | 3-1 |
| 3-3 | EDOS Directives | 3-2 |
| 3-3.1 | Assemble | 3-2 |
| 3-3.2 | Copy | 3-3 |
| 3-3.3 | Dump | 3-3 |
| 3-3.4 | Edit | 3-3 |
| 3-3.5 | Home | 3-4 |
| 3-3.6 | Initialize | 3-4 |
| 3-3.7 | List Directory | 3-5 |
| 3-3.8 | Monitor (EXbug) Return | 3-5 |
| 3-3.9 | Name File | 3-5 |
| 3-3.10 | Print | 3-6 |
| 3-3.11 | Run A Program | 3-6 |
| 3-3.12 | Store | 3-6 |
| 3-3.13 | Transfer (Append) | 3-6 |
| 3-3.14 | Update EDOS System Area | 3-7 |
| 3-4 | Generating EDOS System Area | 3-7 |
| 3-4.1 | Multi-Drive System With Existing System Diskette | 3-7 |
| 3-4.2 | Single-Drive System With Existing System Diskette | 3-8 |
| 3-4.3 | No Existing System Diskette | 3-8 |
| 3-5 | EDOS Error Messages | 3-9 |
| SECTION 4: | EDOS DRIVER | 4-1 |
| 4-1 | Disk Input/Output | 4-1 |
| 4-2 | EXORDisk Signal Specifications | 4-3 |
| APPENDIX A: | LISTING OF EDOS RESIDENT DRIVER | A-1 |
| APPENDIX B: | EXORDisk SCHEMATICS | B-1 |

Figure 1-1. M68R3602 EXORdisk

SECTION 1
UNPACKING AND INSTALLATION
INSTRUCTIONS

1-1 UNPACKING INSTRUCTIONS

Unpack the EXORdisk (Motorola's Floppy Disk) in accordance with the following procedures.

- (a) Remove unit from shipping box.
- (b) Remove door bracing materials from drive unit doors.
- (c) Remove chassis shroud by removing 2 screws in upper rear and 1 screw on lower rear of each side.
- (d) Remove all packing material from inside unit.
- (e) Visually inspect for physical damage.
- (f) Replace chassis shroud.

1-2 EXORdisk INSTALLATION INSTRUCTIONS

Insert the M68IFC EXORciser Interface Module into any free board slot within the EXORciser. Connect the ribbon cable connector from the EXORTape to the Interface Board. Plug the EXORTape power cord into any 3-wire, grounded, 117 VAC, 50/60 Hz. outlet.

1-3 EDOS INSTALLATION INSTRUCTIONS

There are no installation instructions for EDOS (EXORciser floppy Disk Operating System). All EDOS software is provided on one system diskette.

SECTION 2
SYSTEM ORGANIZATION

2-1 EDOS SOFTWARE PROGRAMS

EDOS consists of the following programs:

EDOS Resident Driver
EDOS Executive
EDOS Editor
EDOS Assembler

2-1.1 EDOS Resident Driver

The EDOS Resident Driver Program is that portion of EDOS contained in the PROM. In addition to performing the disk input/output and program loading functions for EDOS, this resident driver program is available for use by a user's program to perform disk read operations, disk write operations, and program overlay and chaining operations (see Section 4).

2-1.2 EDOS Executive

The EDOS Executive Program is brought into RAM memory when the EXbug's MAID command E800;G is typed. The EDOS Executive program performs all EDOS operational and file management functions. The EDOS Executive program is in RAM, and is awaiting an EDOS directive when it prints the character ! on the console device.

2-1.3 EDOS Editor

The EDOS Editor Program is functionally equivalent to the Motorola EXORciser Resident Editor except that editor input is derived from a file on disk, and Editor Program output is stored into a file on disk.

2-1.4 EDOS Assembler

The EDOS Assembler Program is functionally equivalent to the Motorola EXORciser Resident Assembler except that source input is derived from a file on disk, and hex object output is stored into a file on disk.

EDOS SOFTWARE LOCATION

All EDOS programs have been stored on the diskette enclosed with the EXORDisk. To EDOS, the disk storage space is divided into two distinct regions: system area and user file area.

The system area contains the EDOS Executive, EDOS Editor, and EDOS Assembler Programs. The user file area contains user programs in either source or hex form.

The delivered diskette contains a ready-to-use system area, and hex object files of the EDOS Executive, EDOS Editor, and EDOS Assembler Programs in the user file area as follows:

- File 1 contains EDOS Executive Hex
- File 2 contains EDOS Editor Hex
- File 3 contains EDOS Assembler Hex

NOTE

Due to the volatility of the diskette media, it is highly recommended that back-ups of the EDOS Executive, EDOS Editor, and EDOS Assembler Programs be made as soon as possible. This may be done by dumping files 1, 2, and 3 of the supplied diskette using the EDOS D (Dump) directive.

SECTION 3

EDOS OPERATION

3-1 EDOS START-UP PROCEDURES

To start EDOS, follow the EXbug start-up procedure for the EXORciser. Two sets of operator directives now exist, those for the EXbug and those for EDOS. Since the EXbug has not changed, it is awaiting directives. When an exclamation mark (!) is printed on the console device, EDOS is awaiting directives.

The operator can go from EDOS to the EXbug at will.

To enter EDOS from EXbug's MAID simply type E800;G.

To enter EXbug from EDOS simply type M.

NOTE

The operator should get into the habit of turning the EXOR-disk power on prior to starting EXbug. No diskette media should be in a drive while the EXORDisk power is being turned on or off.

- (1) Turn EXORciser, console, and EXORDisk power on.
- (2) Start EXbug.
- (3) Insert a system diskette, one with system area initialized, into drive unit 0 (referred to as system drive) and close door.
- (4) EDOS and EXbug are now ready for use.

3-2 FILE ORGANIZATION

The user file area on each diskette media is divided into seven fixed length areas (files) numbered 1 through 7. Any file may contain either program source or hex object data. When referencing a particular file, the operator enters the diskette file number 1 through 7 preceded by the drive unit number in which that diskette is presently loaded, 0 through 3.

File numbers 1-7, or 01 through 07, refer to files 1 through 7 on the diskette loaded in drive unit 0.

File numbers 11-17 refer to files 1 through 7 on the diskette loaded in drive unit 1.

File numbers 21-27 refer to files 1 through 7 on the diskette loaded in drive unit 2.

File numbers 31-37 refer to files 1 through 7 on the diskette loaded in drive unit 3.

The files are contiguous and begin on the following tracks:

| file | track |
|------|-------|
| 1 | 14 |
| 2 | 23 |
| 3 | 32 |
| 4 | 41 |
| 5 | 50 |
| 6 | 59 |
| 7 | 68 |

9 tracks

230

234 sectors

3-3 EDOS DIRECTIVES

When an exclamation mark (!) is printed on the console device, EDOS is awaiting any one of the following directives. The directives are also identified in Table 3-1 at the end of this Chapter.

3-3.1 ASSEMBLE

PURPOSE: To assemble a user's source program and to produce an assembly listing or a hex object or both.

FORMAT: !An, m, p

where n is the input source file number, m is the output hex file number, and p is the desired assembly alternatives.

if p = 3 only a listing is generated to the list device.

if p = 4 only a hex object output is generated to disk file m.

if p = 2 both a listing and an hex object are generated.

NOTE

Assembler OPT directives must allow list or object for either to be executed.

COMMENTS: All 3 parameters, n, m, and p must be specified. n cannot equal m. If p = 3, existing file m is unaffected. The assembler prints, on the console device, the pass it is performing, 0 through 4, where 0 imply assembly complete.

3-3.2 COPY

PURPOSE: To copy the contents of one diskette onto another.

FORMAT: !C

The contents of the diskette in drive unit 0 are copied onto the diskette in drive unit 1.
(7 minutes)

COMMENTS: The data contained on the diskette in drive unit 0 may be of any format, EDOS or not.

3-3.3 DUMP

PURPOSE: To dump the contents of a user file to the punch output device.

FORMAT: !DCn) or !DTn)

where n is the user file number. DC dumps to cassette & DT dumps to a TTY terminal.

COMMENTS: Leader and trailer (blank) tape is punched if DT is used: the user file is unaffected.

3-3.4 EDIT

PURPOSE: To perform edit operations on a user's source program and to produce an updated source file.

FORMAT: !En, m)

where n is the input source file number and m is the output (updated) source file number. If n is 0, a zero-content input file is assumed (used when entering a new source program from the keyboard). n cannot equal m.

COMMENTS: When the EDOS Editor starts, it prints

EDOS EDITOR
@

All EDOS Editor operations are functionally identical to those of the Motorola EXORciser Resident Editor. Completion of a program edit must be terminated by the editor command

@E\$\$

The E command completes the editing operations, updates and closes the source output file, and returns control to the EDOS Executive Program.

3-3.5 HOME

PURPOSE: To return the head on the selected drive unit back to the "home" or track 0 position.

FORMAT: !Hu)

where u is the drive unit number (0 through 3), in which the diskette to be repositioned, is loaded.

3-3.6 INITIALIZE

PURPOSE: To clear the user file area and to delete the user file names on the specified diskette.

FORMAT: !Iu)

where u is the drive unit number (0 through 3), in which the diskette to be initialized, is loaded. If u is omitted, drive unit 0 is assumed.

COMMENTS: The initialize command does not affect the system area of the diskette media.

3-3.7 LIST DIRECTORY

PURPOSE: To produce a listing on the list device, of all user filenames, filenumbers, and file sizes (in sectors) on the specified diskette.

FORMAT: !Lu)

where u is the drive unit number (0 through 3), in which the diskette whose files are to be listed, is loaded. If u is omitted, drive unit 0 is assumed.

COMMENTS: Each line of the list output contains the file number, file name, and file size in sectors.

3-3.8 MONITOR (EXbug) RETURN

PURPOSE: To return control to EXbug, from the EDOS Executive Program.

FORMAT: !M

3-3.9 NAME FILE

PURPOSE: To assign a .1 to 10 character alphanumeric name to a user file.

FORMAT: !Nn, ~~xxxxxxxxxxx~~)

where n is the file number to be named, and ~~xxxxxxxxxxx~~ is the 1 to 10 character alphanumeric name to be assigned to file n.

COMMENTS: Naming a file replaces any previous name assigned to that file. The Name function has no effect on the files content.

3-3.10 PRINT

PURPOSE: To print the contents of a user file to the list device.

FORMAT: !Pn

where n is the user file number.

COMMENTS: The user file is unaffected.

3-3.11 RUN A PROGRAM

PURPOSE: To execute a user program from a user file.

FORMAT: !Rn

where n is the user file number of the hex file to be loaded. This command is functionally identical to the EXbug LOAD C command.

COMMENTS: Following the loading of the user program control will return to EXbug.

3-3.12 STORE

PURPOSE: To load a user file from the tape input device.

FORMAT: !SCn !SPn or !STn

where n is the user file number to be loaded. SC loads from TI cassette, SP loads from the EXORTape, and ST loads from a TTY terminal.

COMMENTS: The previous contents of the specified user file are replaced with the new data.

3-3.13 TRANSFER (Append)

PURPOSE: To append one user's file to another user's file.

FORMAT: !Tn, m

where n is the file number of the input file whose contents get appended to file number m. The contents file number n are unchanged.

3-3.14 UPDATE EDOS SYSTEM AREA

PURPOSE: To update the EDOS System Area (EDOS Executive, EDOS Editor, or EDOS Assembler Programs) from the tape input device.

FORMAT: !XCn !XPn or !XTn

where n specifies the EDOS Program to be updated. If n = 0, the EDOS Executive is replaced; if n = 1, the EDOS Editor is replaced; if n = 2, the EDOS Assembler is replaced. (See STORE command for input device definition.)

COMMENTS: This command is used primarily to update EDOS System Programs as new versions become available, and to generate additional diskette media with EDOS modules in the System Area (see Paragraph 3-4).

3-4 GENERATING EDOS SYSTEM AREA

The EXORdisk is delivered with one diskette media which contains the EDOS programs in the system area of the diskette. When operating the EXORdisk, it is imperative that the diskette media loaded in drive unit 0 be such a system disk, since EDOS looks to drive unit 0 for the EDOS Executive, EDOS Editor, and EDOS Assembler Programs.

If additional system diskettes are desired, or if a new system diskette must be generated, follow one of the following procedures.

3-4.1 MULTI-DRIVE SYSTEM WITH EXISTING SYSTEM DISKETTE

This procedure assumes an existing system diskette and a system which contains a drive unit 0 and a drive unit 1.

(1) Load existing system diskette into drive unit 0 and a new diskette into drive unit 1.

(2) Type:

!C

(3) When EDOS returns with !, type:

!I1

3-4.2 SINGLE DRIVE SYSTEM WITH EXISTING SYSTEM DISKETTE

This procedure assumes an existing system diskette and a system which contains only one drive unit.

- (1) Follow EDOS START-UP procedure in Paragraph 3-1.
- (2) Type MAID command:

E800;G

- (3) Insert a new diskette into drive unit 0.
- (4) Place the EDOS Executive Program (see Paragraph 2-2) into the tape reader device and type:

!XC0 !XP0 or !XT0

- (5) Place the EDOS Editor Hex into the paper tape reader device and type:

!XC1 !XP1 or !XT1

- (6) Place the EDOS Assembler Hex into the paper tape reader device and type:

!XC2 !XP2 or !XT2

- (7) Type:

!I

3-4.3 NO EXISTING SYSTEM DISKETTE

This procedure assumes that no system diskette exists.

- (1) Start EXbug.
- (2) Place the EDOS Executive Program (see Paragraph 2-2) into the tape reader device and load it.
- (3) Type the MAID command:

20;G

- (4) Proceed to step 3 of Paragraph 3-4. 2.

EDOS ERROR MESSAGES

The following error messages may be printed by EDOS during its operation.

- E1 Disk read error (CRC error after 5 read trys). Copy diskette to recover all but bad data.
- E2 Output file overflow. Output data exceeded 9 track maximum file size.
- E3 Requested drive not ready or diskette not loaded.

Table 3-1. Summary of EDOS Directives

| DIRECTIVE | DESCRIPTION |
|-------------------|-----------------------------------------------------------------------------|
| An, m p) | Assemble user source file n, direct hex to file m, and perform pass 2 of p. |
| C | Copy contents of diskette in drive unit 0 to diskette in drive unit 1. |
| DCn) or DTn) | Dump contents of user file n to punch device. |
| En, m) | Edit user source file n, direct edited source to file m. |
| Hu) | Return head on drive unit u to track 0. |
| Iu) | Initialize (clear) the user file area of the diskette in drive unit u. |
| Lu) | List the user file directory of the diskette in drive unit u. |
| M | Return to EXbug. |
| Nn, xxxxxxxxxxxx) | Assign the 1 to 10 character alphanumeric name to user file n. |
| Pn) | Print contents of user file n to list device. |
| Rn) | Run (load) user hex file n. |
| SCn) SPn) or STn) | Store tape loaded in reader device into file m. |
| Tn, m | Transfer (Append) the contents of file n to the contents of file m. |
| XCn) XPn) or XTn) | Update system module n (See Paragraph 3-3.12). |

30 chas of commt.
52 chs/hour
MAX

© 24 ch/hour

29 952 chas/record
1200 commuted lines.

SECTION 4

EDOS DRIVER

4-1 DISK INPUT/OUTPUT

Provisions have been made in the EDOS Resident Driver (see Appendix A) to enable the programmer to develop user oriented programs which utilize the EXORDISK as a peripheral mass storage device outside of the EDOS environment. Contained in the Driver is a disk read routine (RI) and a disk write routine (WRT) which provide byte oriented input and output capabilities, respectively, to the user.

In order to use the disk input/output routines, RI and WRT, it is the programmer's responsibility to first set up pointers to the area on disk which is to be accessed. This is known as "opening" a disk file. Once a file area on disk has been opened, RI and WRT may be called any number of times. Once the disk file has been opened, the Driver handles all maintenance of the file pointers from then on. It should be noted that only one input file and one output file may be opened at any given time.

The following RAM memory locations are used by the Driver:

| LOC | DESCRIPTION |
|--------|---------------------------------------------|
| 06 | Input file size (sectors) |
| 07 | Input file's beginning track address |
| 08 | Input file's beginning unit/sector address |
| 09 | Controller's read buffer counter |
| 0A | Output file size (sectors) |
| 0B | Output file's beginning track address |
| 0C | Output file's beginning unit/sector address |
| 0D | Controller's write buffer counter |
| 0E, 0F | Temporary locations |

To open an input file, the user simply stores the appropriate input file information into locations 06-09. Then each call to RI will return the next byte of data, from the disk, in the A-register. If no more data exists (i. e. the Input file size = 0) the carry bit is returned as a "1", else the carry bit is returned as a "0". The input file size should be set to the number of sectors + 1 that are to be read before the Driver is to return an end-of-file indication (carry bit set). If the programmer is going to perform his own end-of-file monitoring, the file size may be set to some

arbitrarily large number (i. e. OFFH). The input file's beginning track address should be set to the track number (00-4CH) from which input data is to begin being retrieved. The input file's beginning unit/sector address should be set to contain the drive unit number (00-A1) in bits 6 & 7, and the sector -1 (i. e. 00-19H as opposed to 01-1AH) from which input data is to begin being retrieved. Location 09 should be set to 00. Each call to RI will bring in the next sequential byte of data from the disk. As a sector (128 bytes) of data is read, RI increments the disk address (locations 07 and 08) and decrements the input file size (location 06). Any sector containing a DD mark is ignored.

To open an output file, the user simply stores the appropriate output file information into locations 0A-0D. Then each call to WRT will output to disk the byte contained in the A-register. The output file size should be set to the number of sectors that are allowed to be written before the Driver terminates by printing E3 onto the TTY console and returning to EXbug. If the programmer is going to perform his own maximum output file size monitoring, the output file size must always be kept between 01 and OFFH. The output file's beginning track address should be set to the track number (00-4CH) to which output data is to begin being written. The output file's beginning unit/sector address should be set to contain the drive unit number (00-11) in bits 6 & 7, and the sector (01-1AH) to which output data is to begin being written. Location 0D should be set to 00. Each call to WRT will take the byte contained in the A-register and output it to the EXORdisk. When 128 bytes have been sent to the EXORdisk, WRT writes that data onto the disk and increments the disk address (locations 0B & 0C) and decrements the output file size (location 0A). WRT verifies every sector it has written and if, after 5 attempts, it is unable to write a sector, it writes a DD mark to that sector and advances to the next contiguous disk address and attempts the disk write again.

When the user has written all his data to the disk, using the driver, it is possible that a partial sector of data still remains in the EXORdisk write buffer. To insure that all data has been written to disk, the user should continue outputting a pad character (i. e. 00) until the write buffer reaches 128 bytes and WRT writes it to disk. An example of such a "fill" routine is as follows:

| | | |
|-------|-----|-------|
| FILL | TST | \$0D |
| | BNE | FILL1 |
| | RTS | |
| FILL1 | CLR | A |
| | JSR | WRT |
| | BRA | FILL |

It should be noted that the driver utilizes a logical/physical technique of disk addressing. Sectors on a diskette are physically adjacent and contiguous from 1-26 (01-1AH). It is obvious that after accessing sector 1, an entire revolution of the disk must occur if sector 2 cannot be accessed immediately. To overcome the rotational delays, the driver translates the requested sector address (logical sector) into some other sector address (physical sector) which is then used by the driver. Table TBL is the conversion table for this translation. If sector 2 is requested, physical sector 10 (0AH) is the area on disk accessed; if sector 20 (14H) is requested, physical sector 16 (10H) is the area on disk requested. This entire technique is normally transparent to the user if he remains under the EDOS Driver. Of course, if desired, the contents of TBL may be altered, even to the point of providing a 1:1 translation of logical:physical sectoring.

4-2 EXORDisk SIGNAL SPECIFICATIONS

Electrical - All signals are compatible with MC6820 PIA chips.

Input Data and Status: 8 bits, negative true, PIA address EC00, bits 0-7.

Logic 1: 0 to 0.4 volts
Logic 0: 2.4 volts min.

| | | | | | | | |
|--------------------|---|------------------------|--------------------------|--------------|------------------|------------------|---|
| Read DD Mark | - | Drive Fail Error | Drive Write Prot'd | CRC Error | Unit # MSB | Unit # LSB | - |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |

NOTE
IRQA1 bit 7 is device BUSY

These 8 lines contain status or data, depending upon the state of the "Read Data Byte" signal.

DATA: Bits 0-7 where bit 0 is LSB.

STATUS:

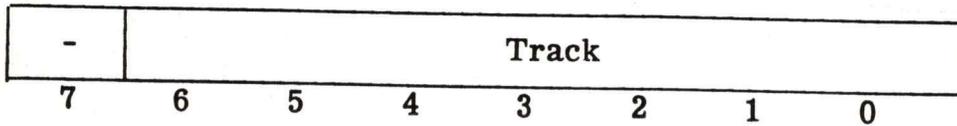
Bits 1 & 2 - Defines last selected unit (00-11).

Bit 3 - If 1, a CRC error was encountered on the last read operation. This must be reset with a "clear error flags" command.

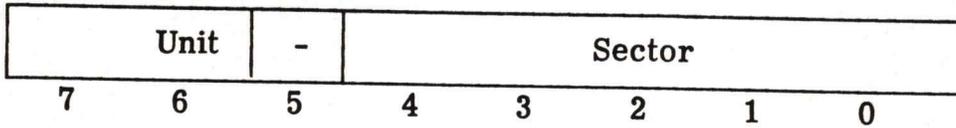
- Bit 4 - If 1, selected drive unit is write protected.
- Bit 5 - If 1, selected drive unit is not up to speed, door is opened, or no diskette is inserted.
- Bit 7 - If 1, a DD mark was encountered on the last read operation. The sector's data was still read. This must be reset with a "clear error flags" command.

Output Data: 8 bits, negative true, PIA address EC06, bits 0-7.

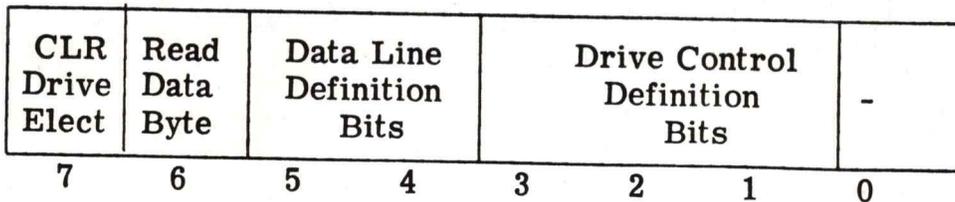
If data is track address:



If data is unit and sector address:



Output Commands: 8 bits, negative true, PIA address EC02, bits 0-7.



NOTE

CB2 provides command acceptance strobe.

Drive control definition bits (3 lines encoded)

A control operation causes "unit busy" on the leading edge of CB2. When the operation is complete, IRQ1 goes Low.

- 001 read a 128-character sector into the controller's read buffer.
- 010 write 128-character sector from controller's write buffer. Data is recycled into write buffer during write operation.
- 011 read a 128-character sector for CRC verification. Controller's read buffer is unaffected.
- 100 seek to specified unit and track.
- 101 clear the controller's error flags and abort the present operation.
- 110 return the selected unit to track 0.
- 111 write the deleted data address mark onto the specified unit/track/sector, when the next "write sector" command is issued.

Data line definition bits (2 lines encoded)

Indicates that the 8 data lines are valid, and describes the type of data contained on the 8 data lines.

- 01 data lines specify track address.
- 10 data lines specify unit and sector address.
- 11 data lines contain data which is to be written into the controller's write buffer. The data is transferred at the leading edge of the "accept-control-strobe" signal.

READ DATA BYTE

As long as this signal is false (0), the drive status signals are gated onto the output lines. When this signal is true (1), the output data is gated onto the output lines. If CB2 signal goes true (1) while this "read-data-byte" signal is true (1), the next data byte is shifted from the controller's read buffer onto the output lines.

Clear drive electronics, data buffers, and data buffer counters. This is a general controller and drive reset command.

MEDIA

IBM DISKETTE OR EQUIVALENT

- . Tracks per inch 48
- . Number of tracks 77

FORMAT

- . Tracks Per Diskette 77 (00-4CH)
- . Sectors Per Track 26 (01-1AH)
- . Bytes Per Sector 128
- . Bytes Per Diskette 256, 256
- . Bits Per Diskette 2, 050, 048

SECTION 5
MAINTENANCE

5-1 DIAGNOSTIC TEST

The EXORdisk diagnostic program listing is presented in Figure 5-1.

5-2 DRAWINGS

The EXORdisk schematic diagrams are presented in Appendix B.

APPENDIX A

A-1

LISTING OF EDOS RESIDENT DRIVER

00001
 00002
 00003

NAM
 OPT
 OPT

FDOS
 SYMBOLS
 O=RESOBJ

RESIDENT MODULE

00005 * 4/9/75

00007 * VERSION 2.0

| | | | | |
|-------|------|-------|-----|--------|
| 00009 | EC00 | DKDID | EQU | \$EC00 |
| 00010 | EC01 | DKDIC | EQU | \$EC01 |
| 00011 | EC02 | DKCOD | EQU | \$EC02 |
| 00012 | EC03 | DKCOC | EQU | \$EC03 |
| 00013 | EC06 | DKDOD | EQU | \$EC06 |
| 00014 | EC07 | DKDOC | EQU | \$EC07 |

| | | | | |
|-------|------|--------|-----|--------|
| 00016 | F564 | XBUG | EQU | \$F564 |
| 00017 | F018 | CO | EQU | \$F018 |
| 00018 | FF90 | TEMP | EQU | \$FF90 |
| 00019 | FF8A | XSTACK | EQU | \$FF8A |

| | | | | |
|-------|------|--------|-----|------|
| 00021 | 0020 | EXEC | EQU | \$20 |
| 00022 | 0023 | UPDATE | EQU | \$23 |

| | | | | |
|-------|------|------|-----|-------|
| 00024 | 0020 | EDIT | EQU | \$20 |
| 00025 | 0400 | ASMB | EQU | \$400 |

| | | | | | |
|-------|------|-------|-----|----|-----------------------|
| 00027 | 0000 | PASS | EQU | 0 | ASSEMBLY PASS INFO |
| 00028 | 0001 | OFIL | EQU | 1 | OFIL NUMBER |
| 00029 | 0002 | OUNIT | EQU | 2 | OFIL UNIT |
| 00030 | 0003 | IUNIT | EQU | 3 | IFIL UNIT - SCTR/UNIT |
| 00031 | 0004 | ISIZE | EQU | 4 | IFIL SIZE |
| 00032 | 0006 | ITRK | EQU | 6 | IFIL TRACK |
| 00033 | 0007 | ISCTR | EQU | 7 | IFIL SECTOR |
| 00034 | 0008 | ICNTR | EQU | 8 | IFIL BUFFER COUNTER |
| 00035 | 0009 | OSIZE | EQU | 9 | OFIL SIZE |
| 00036 | 000B | OTRK | EQU | 11 | OFIL TRACK |
| 00037 | 000C | OSCTR | EQU | 12 | OFIL SECTOR |
| 00038 | 000D | OCNTR | EQU | 13 | OFIL BUFFER COUNTER |
| 00039 | 000E | TITRK | EQU | 14 | TEMP LOC 1 |
| 00040 | 000F | TISZ | EQU | 15 | TEMP LOC 2 |

| | | | | | | |
|-------|------|---------|-------|--------|-------------|----------------|
| 00042 | E800 | | ORG | \$E800 | | |
| 00044 | E800 | FDOS | EQU | * | | |
| 00045 | E800 | BD E841 | JSR | FDOS1 | LOAD | FDOS |
| 00046 | E803 | 7E 0020 | JMP | EXEC | START | EXEC |
| 00048 | E806 | INTIO | EQU | * | | |
| 00049 | E806 | 7E E859 | JMP | RESET | INITIALIZE | I/O |
| 00051 | E809 | XRI | EQU | * | | |
| 00052 | E809 | 7E E91B | JMP | RI | DISK | READ VECTOR |
| 00054 | E80C | XWRT | EQU | * | | |
| 00055 | E80C | 7E E98E | JMP | WRT | DISK | WRITE VECTOR |
| 00057 | E80F | UPDT | EQU | * | | |
| 00058 | E80F | 7E EA79 | JMP | PATCH | | |
| 00059 | E812 | 7E 0023 | JMP | UPDATE | START | UPDATE |
| 00061 | E815 | PROG | EQU | * | | |
| 00062 | E815 | BD E887 | JSR | REDX | LOAD | PROGRAM |
| 00063 | E818 | 7E F564 | JMP | XBUG | GO TO | EXBUG |
| 00065 | E81B | ASSEM | EQU | * | | |
| 00066 | E81B | BD E887 | JSR | REDX | LOAD | ASSEMBLER |
| 00067 | E81E | BD E82D | JSR | RESTR | RESTORE | IFILE POINTERS |
| 00068 | E821 | 7E 0400 | JMP | ASMB | START | ASSEMBLER |
| 00070 | E824 | EDITR | EQU | * | | |
| 00071 | E824 | BD E887 | JSR | REDX | LOAD | EDITOR |
| 00072 | E827 | BD E82D | JSR | RESTR | RESTORE | IFILE PNTRS |
| 00073 | E82A | 7E 0020 | JMP | EDIT | START | EDITOR |
| 00075 | E82D | RESTR | EQU | * | | |
| 00076 | E82D | DE 0F | LDX | TISZ | RESTORE | IFILE PNTRS |
| 00077 | E82F | DF 04 | STX | ISIZ | IFILE | SIZE |
| 00078 | E831 | 96 0E | LDA A | TITRK | | |
| 00079 | E833 | 97 06 | STA A | ITRK | TRACK | |
| 00080 | E835 | 96 03 | LDA A | IUNIT | UNIT/SECTOR | |
| 00081 | E837 | 0C | CLC | | | |
| 00082 | E838 | 46 | ROR A | | | |
| 00083 | E839 | 46 | ROR A | | | |
| 00084 | E83A | 46 | ROR A | | | |
| 00085 | E83B | 97 07 | STA A | ISCTR | | |
| 00086 | E83D | 7F 0008 | CLR | ICNTR | BUFFER | COUNTER |
| 00087 | E840 | 39 | RTS | | | |

```

00089      E841      FDOS1  EQU      *
00090 E841 BD E859      JSR      RESET   RESET ELECTRONICS
00091 E844 CE 004E      LDX      #78     SET ISIZE=78
00092 E847 DF 04       STX      ISIZE
00093 E849 7F 0006      CLR      ITRK    TRACK=1
00094 E84C 7C 0006      INC      ITRK
00095 E84F 7F 0007      CLR      ISCTR   SETCTOR=0
00096 E852 7F 0008      CLR      ICNTR   RD BFR EMPTY
00097 E855 BD E887      JSR      REDX    LOAD FDOS
00098 E858 39          RTS

```

```

00100      * SUBROUTINE TO SET UP PIA'S AND RESET
00101      * DRIVE ELECTRONICS.

```

```

00103      E859      RESET  EQU      *
00104 E859 7F EC01      CLR      DKDIC   SET DIRECTIONS
00105 E85C 7F EC03      CLR      DKCOC
00106 E85F 7F EC07      CLR      DKDOC
00107 E862 7F EC00      CLR      DKDID
00108 E865 86 FF        LDA A    #$FF
00109 E867 B7 EC02      STA A    DKCOD
00110 E86A B7 EC06      STA A    DKDOD
00111 E86D 86 04        LDA A    #$4     SET DATA IN CNTRL
00112 E86F B7 EC01      STA A    DKDIC
00113 E872 86 04        LDA A    #$4     SET DATA OUT CNTRL
00114 E874 B7 EC07      STA A    DKDOC
00115 E877 86 2C        LDA A    #$2C    SET CMD OUT CNTRL
00116 E879 B7 EC03      STA A    DKCOC
00117 E87C 86 80        LDA A    #$80    ISSUE CLEAR ELECTRONICS
00118 E87E B7 EC02      STA A    DKCOD
00119 E881 86 0C        LDA A    #$0C
00120 E883 BD EA3E      JSR      LOOP
00121 E886 39          RTS

```

```

00123      * SUBROUTINE TO READ AN OBJECT FILE
00124      * INTO MEMORY

```

```

00126      E887      REDX   EQU      *
00127 E887 BD E912      JSR      RIX     GET A CHAR
00128 E88A 25 23        BCS     REDX3    EOF
00129 E88C 81 53        CMP A    #$53    S?
00130 E88E 26 F7        BNE     REDX     NO
00131 E890 7F FF92      CLR     TEMP+2   RESET CHKSM
00132 E893 8D 7D        BSR     RIX     GET A CHAR
00133 E895 25 10        BCS     REDX2    LOAD ERROR
00134 E897 81 30        CMP A    #$30
00135 E899 77 FC        BEQ     REDX     HOP RIX-CKTD

```

```

00136 E89B 81 31      CMP A  ##31
00137 E89D 27 11      BEQ   REDX4   DATA BLK
00138 E89F 81 39      CMP A  ##39
00139 E8A1 27 E4      BEQ   REDX   EOF BLK-SKIP
00140 E8A3 81 1B      CMP A  ##1B   ESC?
00141 E8A5 27 08      BEQ   REDX3   YES-END OF OBJECT FILE
00142 E8A7 86 3F      REDX2 LDA A  ##3F   LOAD ERROR
00143 E8A9 BD F018     JSR   CO
00144 E8AC 7E F564     JMP   XBUG

```

```

00146 E8AF 39      REDX3 RTS

00148 E8B0 8D 29      REDX4 BSR   RDBYT   GET BYTE COUNT
00149 E8B2 4A      DEC A  DECR   COUNT
00150 E8B3 B7 FF93     STA A  TEMP+3  SAVE CNT
00151 E8B6 8D 23      BSR   RDBYT   READ ADDR(H)
00152 E8B8 B7 FF90     STA A  TEMP
00153 E8BB 8D 1E      BSR   RDBYT   READ ADDR(L)
00154 E8BD B7 FF91     STA A  TEMP+1
00155 E8C0 7D FF93     REDX5 TST   TEMP+3  COUNT=0?
00156 E8C3 27 0D      BEQ   REDX6   YES
00157 E8C5 8D 14      BSR   RDBYT   NO-READ DATA
00158 E8C7 FE FF90     LDX   TEMP
00159 E8CA A7 00      STA A  X      SAVE IT
00160 E8CC 08      INX   INCR ADDRESS
00161 E8CD FF FF90     STX   TEMP
00162 E8D0 20 EE      BRA   REDX5   CONTINUE
00163 E8D2 8D 07      REDX6 BSR   RDBYT   READ CHKSM
00164 E8D4 7C FF92     INC   TEMP+2
00165 E8D7 26 CE      BNE   REDX2   CHKSM ERROR
00166 E8D9 20 AC      BRA   REDX   CHKSM OK

```

```

00168      E8DB      RDBYT EQU   *
00169 E8DB 8D 35      BSR   RIX   GET A CHAR
00170 E8DD 25 C8      BCS   REDX2  EOF
00171 E8DF 8D 1D      BSR   CHEX  CONVERT TO HEX
00172 E8E1 25 C4      BCS   REDX2
00173 E8E3 48      ASL A
00174 E8E4 48      ASL A
00175 E8E5 48      ASL A
00176 E8E6 48      ASL A
00177 E8E7 36      PSH A
00178 E8E8 8D 28      BSR   RIX
00179 E8EA 25 BB      BCS   REDX2
00180 E8EC 8D 10      BSR   CHEX
00181 E8EE 25 B7      BCS   REDX2
00182 E8F0 33      PUL B
00183 E8F1 1B      ABA
00184 E8F2 16      TAB
00185 E8F3 BB FF92     ADD A  TEMP+2  ADD TO CHKSM
00186 E8F6 B7 FF92     STA A  TEMP+2
00187 E8F9 7A FF93     DEC   TEMP+3  DECR BYTE CNT
00188 E8FC 17      TBA
00189 E8FD 39      RTS

```

```

00191 E8FE 80 30 CHEX SUB A ##30
00192 E900 25 0F BCS CHEX2
00193 E902 8B E9 ADD A ##E9
00194 E904 25 0B BCS CHEX2
00195 E906 8B 06 ADD A #6
00196 E908 2A 04 BPL CHEX1
00197 E90A 8B 07 ADD A #7
00198 E90C 25 03 BCS CHEX2
00199 E90E 8B 0A CHEX1 ADD A #10
00200 E910 0C CLC
00201 E911 39 CHEX2 RTS
    
```

```

00203 * SUBROUTINE TO READ AN ASCII BYTE FROM DISK
00204 * & RETURN IT IN A-REGISTER. IF EOF, CARRY IS SET
    
```

```

00206 E912 RIX EQU *
00207 E912 BD E91B JSR RI GET BYTE
00208 E915 25 03 BCS RIX1 EOF
00209 E917 84 7F AND A ##7F
00210 E919 0C CLC
00211 E91A 39 RIX1 RTS
    
```

```

00213 * SUBROUTINE TO READ AN 8-BIT BYTE FROM DISK &
00214 * RETURN IT IN A-REGISTER. IF EOF, CARRY IS SET.
    
```

```

00216 E91B RI EQU *
00217 E91B 7D 0008 TST ICNTR COUNT=0
00218 E91E 26 4D BNE RI10 NO
00219 E920 CE 0006 RI5 LDX #ITRK
00220 E923 BD E9EF JSR INCDA
00221 E926 DE 04 LDX ISIZE DECR & CHK IF FILE SIZE
00222 E928 09 DEX
00223 E929 26 05 BNE RI3 OK
00224 E92B 7F 0008 CLR ICNTR
00225 E92E 0D SEC SET EOF
00226 E92F 39 RTS
    
```

```

00228 E930 DF 04 RI3 STX ISIZE
00229 E932 96 07 LDA A ISCTR XMIT U/S
00230 E934 BD EA03 JSR XUS
00231 E937 BD EA4D JSR CHK MAKE SURE A DISK
00232 E93A 86 80 LDA A #128 SET CNTR =128
00233 E93C 97 08 STA A ICNTR
00234 E93E 86 05 LDA A #5 SET TRY COUNT=5
00235 E940 B7 FF94 STA A TEMP+4
00236 E943 96 06 LDA A ITRK SEEK TRACK
    
```

| | | | | | | | |
|-------|------|----|------|------|-------|--------|-------------------|
| 00237 | E945 | BD | EA2B | | JSR | SEEK | |
| 00238 | E948 | 86 | 02 | RI6 | LDA A | #2 | READ DATA |
| 00239 | E94A | BD | EA3E | | JSR | LOOP | |
| 00240 | E94D | B6 | EC00 | | LDA A | DKDID | DD MARK? |
| 00241 | E950 | 84 | 80 | | AND A | #\$80 | |
| 00242 | E952 | 27 | 05 | | BEQ | RI4 | NO |
| 00243 | E954 | BD | EA38 | | JSR | RFLAG | YES-RESET FLAG |
| 00244 | E957 | 20 | C7 | | BRA | RI5 | GO TO NEXT SECTOR |
| 00246 | E959 | B6 | EC00 | RI4 | LDA A | DKDID | CRC ERROR |
| 00247 | E95C | 84 | 08 | | AND A | #\$8 | |
| 00248 | E95E | 27 | 0D | | BEQ | RI10 | NO |
| 00249 | E960 | BD | EA38 | | JSR | RFLAG | YES-RESET FLAG |
| 00250 | E963 | 7A | FF94 | | DEC | TEMP+4 | DECR TRIES |
| 00251 | E966 | 26 | E0 | | BNE | RI6 | TRY AGAIN |
| 00252 | E968 | 86 | 01 | | LDA A | #1 | CAN'T READ MEDIA |
| 00253 | E96A | 7E | EA57 | | JMP | CHK1 | |
| 00255 | E96D | 86 | 3C | RI10 | LDA A | #\$3C | SET CMD CNTRL |
| 00256 | E96F | B7 | EC03 | | STA A | DKCOC | |
| 00257 | E972 | 86 | 40 | | LDA A | #\$40 | SET FOR READ DATA |
| 00258 | E974 | B7 | EC02 | | STA A | DKCOD | |
| 00259 | E977 | B6 | EC00 | | LDA A | DKDID | READ DATA |
| 00260 | E97A | 36 | | | PSH A | | |
| 00261 | E97B | 86 | 2C | | LDA A | #\$2C | RESET CMD CNTRL |
| 00262 | E97D | B7 | EC03 | | STA A | DKCOC | |
| 00263 | E980 | 86 | 40 | | LDA A | #\$40 | STROBE BFR |
| 00264 | E982 | B7 | EC02 | | STA A | DKCOD | |
| 00265 | E985 | 7F | EC02 | | CLR | DKCOD | |
| 00266 | E988 | 7A | 0008 | | DEC | ICNTR | DECR READ CNTR |
| 00267 | E98B | 32 | | | PUL A | | |
| 00268 | E98C | 0C | | | CLC | | |
| 00269 | E98D | 39 | | | RTS | | |

00271 * SUBROUTINE TO WRITE A BYTE TO DISK.
 00272 * EXPECTS BYTE IN A-REGISTER

| | | | | | | | |
|-------|------|----|------|------|-------|-------|------------------|
| 00274 | E98E | | | WRT | EQU | * | |
| 00275 | E98E | B7 | EC06 | | STA A | DKDOD | OUTPUT DATA |
| 00276 | E991 | 86 | 30 | | LDA A | #\$30 | |
| 00277 | E993 | B7 | EC02 | | STA A | DKCOD | |
| 00278 | E996 | 7C | 000D | | INC | OCNTR | INCR BFR COUNT |
| 00279 | E999 | 96 | 0D | | LDA A | OCNTR | =128? |
| 00280 | E99B | 81 | 80 | | CMP A | #128 | |
| 00281 | E99D | 27 | 01 | | BEQ | WRT4 | YES |
| 00282 | E99F | 39 | | | RTS | | NO-EXIT |
| 00284 | E9A0 | 7F | 000D | WRT4 | CLR | OCNTR | CLR COUNT |
| 00285 | E9A3 | 96 | 0C | WRT1 | LDA A | OSCTR | XMIT U/S |
| 00286 | E9A5 | BD | EA03 | | JSR | XUS | |
| 00287 | E9A8 | BD | EA4D | | JSR | CHK | MAKE SURE A DISK |
| 00288 | E9AB | 86 | 05 | | LDA A | #5 | SET TRY COUNT=5 |

```

00289 E9AD B7 FF94      STA A  TEMP+4
00290 E9B0 96 0B      LDA A  OTRK      SEEK TRACK
00291 E9B2 BD EA2B      JSR    SEEK
00292 E9B5 86 04      WRT2  LDA A  #4      WRITE DATA
00293 E9B7 BD EA3E      JSR    LOOP
00294 E9BA 86 06      LDA A  #6      READ FOR CRC
00295 E9BC BD EA3E      JSR    LOOP
00296 E9BF B6 EC00      LDA A  DKDID    CRC ERROR?
00297 E9C2 84 08      AND A  #8
00298 E9C4 27 12      BEQ    WRT3     NO
00299 E9C6 BD EA38      JSR    RFLAG    YES-RESET FLAG
00300 E9C9 7A FF94      DEC    TEMP+4   DECR TRY COUNT
00301 E9CC 26 E7      BNE    WRT2     TRY AGAIN
00302 E9CE 86 0E      LDA A  ##E     WRTIE AS DD
00303 E9D0 BD EA3E      JSR    LOOP
00304 E9D3 BD E9DC      JSR    WRTN     INCR DA & CHK SIZE
00305 E9D6 20 CB      BRA    WRT1
00306 E9D8 BD E9DC      WRT3  JSR    WRTN
00307 E9DB 39          RTS
    
```

00309 * SUBROUTINE TO INCR DA & CHK OFILE SIZE

```

00311      E9DC      WRTN  EQU    *
00312 E9DC CE 000B      LDX    #OTRK
00313 E9DF BD E9EF      JSR    INCDA
00314 E9E2 DE 09      LDX    OSIZE
00315 E9E4 09          DEX
00316 E9E5 DF 09      STX    OSIZE
00317 E9E7 2B 01      BMI    WRTN1
00318 E9E9 39          RTS
00319 E9EA 86 02      WRTN1 LDA A  #2
00320 E9EC 7E EA57      JMP    CHK1
    
```

00322 * SUBROUTINE TO INCR DA
 00323 * TRACK IN 0, X, SECTOR IN 1, X

```

00325      E9EF      INCDA  EQU    *
00326 E9EF 6C 01      INC    1, X
00327 E9F1 A6 01      LDA A  1, X      SECTOR=27?
00328 E9F3 84 1F      AND A  ##$1F
00329 E9F5 81 1B      CMP A  #27
00330 E9F7 27 01      BEQ    INCDB     YES
00331 E9F9 39          RTS             NO
    
```

00333 E9FA A6 01 INCDB LDA A 1, X
 00334 E9FC 84 C1 AND A ##C1 SET CTR=1

```

00335 E9FE A7 01      STA A  1,X
00336 EA00 6C 00      INC   X          INCR TRACK
00337 EA02 39          RTS

```

```

00339          * SUBROUTINE TO XMIT UNIT/SECTOR (LOGICAL) BYTE

```

```

00341          EA03      XUS      EQU      *
00342 EA03 36          PSH A  EXTRACT  LOG SCTR
00343 EA04 84 1F      AND A  #$1F      MASK OFF DRIVE NO
00344 EA06 CE EA5E    LDX   #TBL      GET TABLE PNTR
00345 EA09 FF FF94    STX   TEMP+4    AND SAVE FOR ARITHMETIC
00346 EA0C 5F          CLR B  MAKE      → INTO SCTR PNTR
00347 EA0D BB FF95    ADD A  TEMP+5    ADD SECTOR NO to
00348 EA10 F9 FF94    ADC B  TEMP+4    TABLE ADDRESS to get
00349 EA13 B7 FF95    STA A  TEMP+5    PHYSICAL SECTOR NO
00350 EA16 F7 FF94    STA B  TEMP+4
00351 EA19 FE FF94    LDX   TEMP+4
00352 EA1C 33          PUL B  MERGE    → UNIT & PHYS SCTR.
00353 EA1D C4 C0      AND B  #$C0
00354 EA1F A6 00      LDA A  X
00355 EA21 1B          ABA
00356 EA22 B7 EC06    STA A  DKDOD     ISSUE IT
00357 EA25 86 20      LDA A  #$20
00358 EA27 B7 EC02    STA A  DKCOD
00359 EA2A 39          RTS

```

```

00361          * SUBROUTINE TO SEEK TRACK IN A

```

```

00363          EA2B      SEEK     EQU      *
00364 EA2B B7 EC06    STA A  DKDOD
00365 EA2E 86 10      LDA A  #$10
00366 EA30 B7 EC02    STA A  DKCOD
00367 EA33 86 08      LDA A  #$8
00368 EA35 7E EA3E    JMP   LOOP

```

```

00370          * SUBROUTINE TO RESET ERROR FLAG

```

```

00372          EA38      RFLAG    EQU      *
00373 EA38 86 0A      LDA A  #$A
00374 EA3A B7 EC02    STA A  DKCOD

```

00375 EA3D 39

RTS

00377

* SUBROUTINE TO ISSUE (A) CMD & LOOP ON BUSY

```

00379      EA3E      LOOP      EQU      *
00380 EA3E F6 EC00      LDA B      DKDID      CLEAR BUSY
00381 EA41 B7 EC02      STA A      DKCOD      ISSUE CMD
00382 EA44 B6 EC01 LOOP1  LDA A      DKDIC      DONE?
00383 EA47 2A FB        BPL      LOOP1      NO
00384 EA49 F6 EC00      LDA B      DKDID      YES-CLR BUSY
00385 EA4C 39          RTS          EXIT
    
```

00387

* SUBROUTINE TO CHECK IF A DISK, ELSE ERROR 3

```

00389      EA4D      CHK      EQU      *
00390 EA4D B6 EC00      LDA A      DKDID
00391 EA50 84 20        AND A      #$20
00392 EA52 26 01        BNE      CHK2
00393 EA54 39          RTS          OK
00394 EA55 86 03      CHK2  LDA A      #3          ERROR
    
```

00396

* ROUTINE TO PUT ERROR (A)

```

00398 EA57 8A 30      CHK1  ORA A      #$30      CONVERT TO ASCII
00399 EA59 BD F018      JSR      CO
00400 EA5C 7E F564      JMP      XBUG
    
```

00402

* PHYSICAL SECTOR TABLE IS IN ORDER OF
 * LOGICAL SECTOR NUMBER.

```

00405      EA5E      TBL      EQU      *-1
00407 EA5F 01          FCB      $1
00408 EA60 0A          FCB      $A
00409 EA61 13          FCB      $13
00410 EA62 02          FCB      $2
00411 EA63 0B          FCB      $B
00412 EA64 14          FCB      $14
    
```

| | | | | |
|-------|------|---------|-----|---------|
| 00413 | EA65 | 03 | FCB | \$3 |
| 00414 | EA66 | 0C | FCB | \$C |
| 00415 | EA67 | 15 | FCB | \$15 |
| 00416 | EA68 | 04 | FCB | \$4 |
| 00417 | EA69 | 0D | FCB | \$D |
| 00418 | EA6A | 16 | FCB | \$16 |
| 00419 | EA6B | 05 | FCB | \$5 |
| 00420 | EA6C | 0E | FCB | \$E |
| 00421 | EA6D | 17 | FCB | \$17 |
| 00422 | EA6E | 06 | FCB | \$6 |
| 00423 | EA6F | 0F | FCB | \$F |
| 00424 | EA70 | 18 | FCB | \$18 |
| 00425 | EA71 | 07 | FCB | \$7 |
| 00426 | EA72 | 10 | FCB | \$10 |
| 00427 | EA73 | 19 | FCB | \$19 |
| 00428 | EA74 | 08 | FCB | \$8 |
| 00429 | EA75 | 11 | FCB | \$11 |
| 00430 | EA76 | 1A | FCB | \$1A |
| 00431 | EA77 | 09 | FCB | \$9 |
| 00432 | EA78 | 12 | FCB | \$12 |
| 00434 | EA79 | 8E FF8A | LDS | #XSTACK |
| 00435 | EA7C | BD E841 | JSR | FDOS1 |
| 00436 | EA7F | 7E 0023 | JMP | UPDATE |

| | | | | | |
|-----------|--------------|-------|-------|--------|------------------------|
| 00438 | EB00 | | ORG | \$EB00 | |
| 00440 | EC04 | PTDTA | EQU | \$EC04 | |
| 00441 | EC05 | PTCTL | EQU | \$EC05 | |
| 00443 | EB00 | PTLDR | EQU | * | |
| 00444 | EB00 20 0E | | BRA | LDR0 | |
| 00446 | EB02 | RDRIN | EQU | * | |
| 00447 | EB02 20 39 | | BRA | GETC | |
| 00449 | EB04 | INITR | EQU | * | |
| 00450 | EB04 7F EC05 | | CLR | PTCTL | |
| 00451 | EB07 7F EC04 | | CLR | PTDTA | |
| 00452 | EB0A 86 3C | | LDA A | ##3C | |
| 00453 | EB0C B7 EC05 | | STA A | PTCTL | |
| 00454 | EB0F 39 | | RTS | | |
| 00456 | EB10 | LDR0 | EQU | * | |
| 00457 | EB10 0F | | SEI | | |
| 00458 | EB11 BD EB04 | | JSR | INITR | |
| 00459 | EB14 8D 27 | LDR1 | BSR | GETC | GET A CHAR |
| 00460 | EB16 25 22 | | BCS | LDR5 | NO TAPE |
| 00461 | EB18 81 53 | | CMP A | ##53 | S? |
| 00462 | EB1A 26 F8 | | BNE | LDR1 | NO |
| 00463 | EB1C 7F FF92 | | CLR | TEMP+2 | |
| 00464 | EB1F 8D 1C | | BSR | GETC | GET A CHAR |
| 00465 | EB21 25 10 | | BCS | LDR3 | NO TAPE |
| ****ERROR | 205 | | | | |
| 00466 | EB23 81 30 | D | CMPA | ##30 | |
| 00467 | EB25 27 ED | | BEQ | LDR1 | HEADER BLOCK-SKIP |
| 00468 | EB27 81 31 | | CMP A | ##31 | |
| 00469 | EB29 27 5F | | BEQ | LDR6 | DATA BLOCK |
| 00470 | EB2B 81 39 | | CMP A | ##39 | |
| 00471 | EB2D 27 E5 | | BEQ | LDR1 | EOF BLOCK-SKIP |
| 00472 | EB2F 81 1B | | CMP A | ##1B | ESC? |
| 00473 | EB31 27 07 | | BEQ | LDR5 | YES-END OF OBJECT FILE |
| 00474 | EB33 86 3F | LDR3 | LDA A | ##3F | PRINT ? |
| 00475 | EB35 BD F018 | | JSR | CO | |
| 00476 | EB38 20 00 | | BRA | LDR5 | |
| 00478 | EB3A 7E F564 | LDR5 | JMP | XBUG | |

```

00480      EB3D      GETC      EQU      *
00481 EB3D B6 EC04      LDA A   PTDTA      CLR INTERRUPT
00482 EB40 86 34      LDA A   #$34      STROBE RDR
00483 EB42 B7 EC05      STA A   PTCTL
00484 EB45 86 3C      LDA A   #$3C
00485 EB47 B7 EC05      STA A   PTCTL
00486 EB4A FF FF94      STX    TEMP+4
00487 EB4D CE 0000      LDX    #0          SET TIME OUT
00488 EB50 B6 EC05 GETC1  LDA A   PTCTL      DONE?
00489 EB53 2B 08      BMI    GETC2      YES
00490 EB55 09          DEX
00491 EB56 26 F8      BNE    GETC1      NO-TIME OUT?
00492 EB58 FE FF94      LDX    TEMP+4      NO
00493 EB5B 0D          SEC
00494 EB5C 39          RTS
00495 EB5D B6 EC04 GETC2  LDA A   PTDTA      GET CHAR
00496 EB60 84 7F      AND A   #$7F      STRIP PARITY
00497 EB62 FE FF94      LDX    TEMP+4
00498 EB65 0C          CLC
00499 EB66 39          RTS      CLR CARRY

```

```

00501      EB67      RDPR      EQU      *          MAKE A BYTE FROM 2 CHARS
00502 EB67 8D D4          BSR    GETC        GET A CHAR
00503 EB69 25 C8          BCS    LDR3
00504 EB6B 8D 23          BSR    CVHEX      CONVERT TO HEX
00505 EB6D 25 C4          BCS    LDR3
00506 EB6F 48          ASL A
00507 EB70 48          ASL A
00508 EB71 48          ASL A
00509 EB72 48          ASL A
00510 EB73 36          PSH A
00511 EB74 8D C7          BSR    GETC
00512 EB76 25 BB          BCS    LDR3
00513 EB78 8D 16          BSR    CVHEX
00514 EB7A 25 B7          BCS    LDR3
00515 EB7C 33          PUL B
00516 EB7D 1B          ABA
00517 EB7E 16          TAB
00518 EB7F BB FF92      ADD A   TEMP+2    ADD TO CHKSM
00519 EB82 B7 FF92      STA A   TEMP+2
00520 EB85 7A FF93      DEC    TEMP+3
00521 EB88 17          TBA
00522 EB89 39          RTS

```

```

00524 EB8A 20 18      LDR6   BRA    DATA
00525 EB8C 20 A5      LDR7   BRA    LDR3
00526 EB8E 20 84      LDR8   BRA    LDR1

```

```

00528 EB90 80 30      CVHEX  SUB A   #$30
00529 EB92 25 0F      BCS    CVHEX2
00530 EB94 8B F6      ADD A   #$F0

```

```

00531 EB96 25 0B      BCS      CVHEX2
00532 EB98 8B 06      ADD A    #6
00533 EB9A 2A 04      BPL      CVHEX1
00534 EB9C 8B 07      ADD A    #7
00535 EB9E 25 03      BCS      CVHEX2
00536 EBA0 8B 0A      CVHEX1  ADD A    #10
00537 EBA2 0C          CLC
00538 EBA3 39          CVHEX2  RTS

```

```

00540          EBA4      DATA  EQU      *
00541 EBA4 8D C1      BSR      RDPR      GET COUNT BYTE
00542 EBA6 4A          DEC A     DECR      COUNT
00543 EBA7 B7 FF93    STA A    TEMP+3
00544 EBAA 8D BB      BSR      RDPR      READ ADDR(H)
00545 EBAC B7 FF90    STA A    TEMP
00546 EBAF 8D B6      BSR      RDPR      READ ADDR(L)
00547 EBB1 B7 FF91    STA A    TEMP+1
00548 EBB4 FE FF90    LDX      TEMP
00549 EBB7 7D FF93    DATA1  TST      TEMP+3
00550 EBBA 27 07      BEQ      DATA2    YES
00551 EBBC 8D A9      BSR      RDPR      NO-READ DATA
00552 EBBE A7 00      STA A    X         SAVE IT
00553 EBC0 08          INX
00554 EBC1 20 F4      BRA      DATA1    INCR ADDRESS
00555 EBC3 8D A2      DATA2  BSR      RDPR      READ CHKSM
00556 EBC5 7C FF92    INC      TEMP+2
00557 EBC8 26 C2      BNE      LDR7      CHKSM ERROR
00558 EBCA 20 C2      BRA      LDR8      CHKSM OK

```

```

00560          END
DKDID  EC00
DKDIC  EC01
DKCOD  EC02
DKCOC  EC03
DKDOD  EC06
DKDOC  EC07
XBUG   F564
CO     F018
TEMP   FF90
XSTACK FF8A
EXEC   0020
UPDATE 0023
EDIT   0020
ASMB   0400
PASS   0000
OFILE  0001
OUNIT  0002
IUNIT  0003
ISIZE  0004
ITRK   0006
ISCTR  0007
ICNTR  0008
OSIZE  0009
OTRK   000B

```

00001

NAM DIAG68

00003

*ICOM, INC. FD360-X-68 DIAGNOSTIC

00005

*PROCEDURE:

00006

* LOAD DIAGNOSTIC TAPE INTO RAM

00007

* START THE DIAGNOSTIC PROGRAM AT LOCATION 100 HEX

00008

* INSERT A SCRATCH DISKETTE INTO THE DRIVE UNIT
* TO BE TESTED.

00009

* TYPE THE DESIRED COMMAND

00010

* CONTINUOUSLY OPERATING TESTS MUST BE MANUALLY
* ABORTED (CARRIAGE RETURN STARTS THE TEST)

00011

00012

00014

*U = DRIVE UNIT NUMBER 0, 1, 2, OR 3.

00015

*T = TRACK 0 - 76 DECIMAL -

00016

*S = SECTOR 1 - 26 DECIMAL

00018

*COMMANDS:

00020

*A - CLEAR DRIVE ELECTRONICS

00021

*BU,T - SEEK TO TRACK

00022

*DU,S - READ (SECTOR) TO BUFFER FROM PRESENT TRACK

00023

*FU,S - WRITE (BUFFER) TO SECTOR ON PRESENT TRACK

00024

*GU,S - READ/WRITE TEST (CONTINUOUS) USING (BUFFER)

00025

*HU - TRACK 0 TO TRACK 76 LOOP (CONTINUOUS)

00026

*I - UNIT SELECT TEST

00027

*JU - SEEK TEST PERFORMED ONCE

00028

*KU - SEEK TEST PERFORMED CONTINUOUSLY

00029

*LU - SEEK TEST READ ONLY (CONTINUOUS)

00030

*MU - DD MARK TEST PERFORMED ONCE

00031

*N - RETURN TO EXBUG

00032

*OXX - FILL THE BUFFER WITH THE HEX VALUE XX

00033

*P - PRINT THE CONTENTS OF THE BUFFER

00035

*LIST OF ERROR MESSAGES:

00037

*XX - SELECTED DRIVE UNIT NOT READY

00038

*01 - CRC ERROR ON 5 READ ATTEMPTS -- 01(TRK)

00039

*02 - CRC ERROR ON 5 WRITE ATTEMPTS -- 02(TRK)

00040

*03 - READ/WRITE TEST DATA ERROR --

00041

* 03(REC'D)(EXP'D)(BYTE#)

00042

*04 - UNIT SELECT ERROR -- 04(REC'D)(EXP'D)

00043

*05 - SEEK ERROR -- 05(REC'D)(EXP'D)(TRK)(SCTR)

00044

*06 - DD MARK ERROR -- 06(SCTR)

00045

*07 - DD MARK ERROR ON READ/WRITE TEST

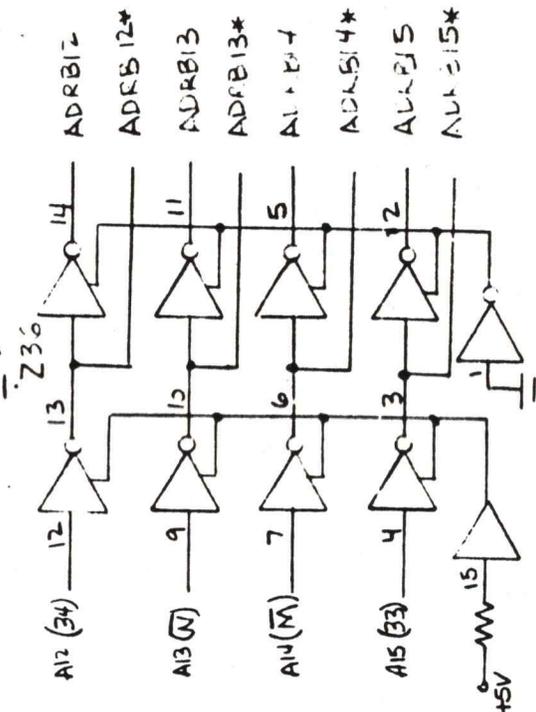
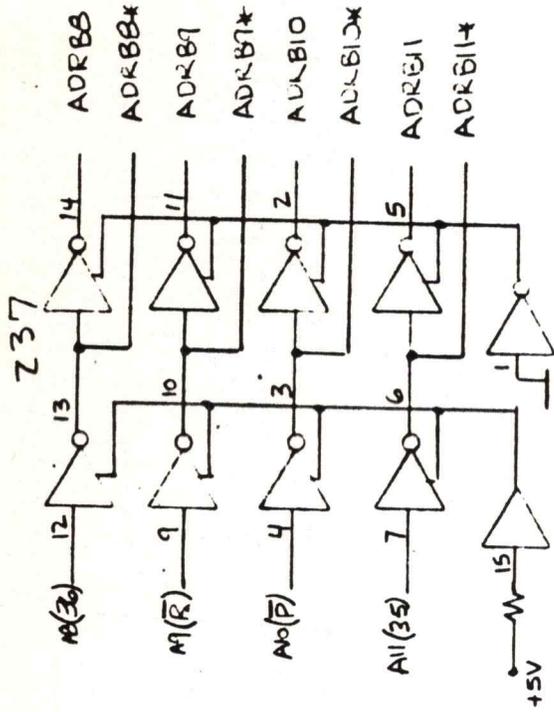
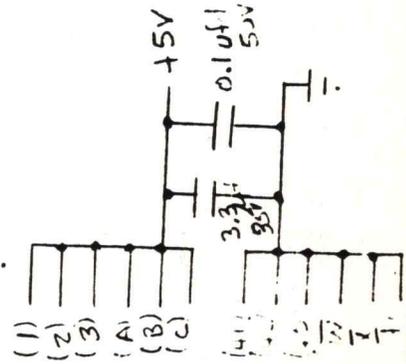
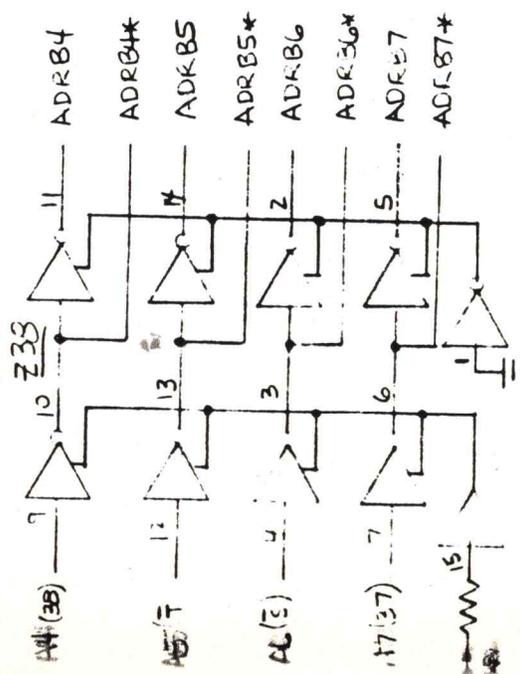
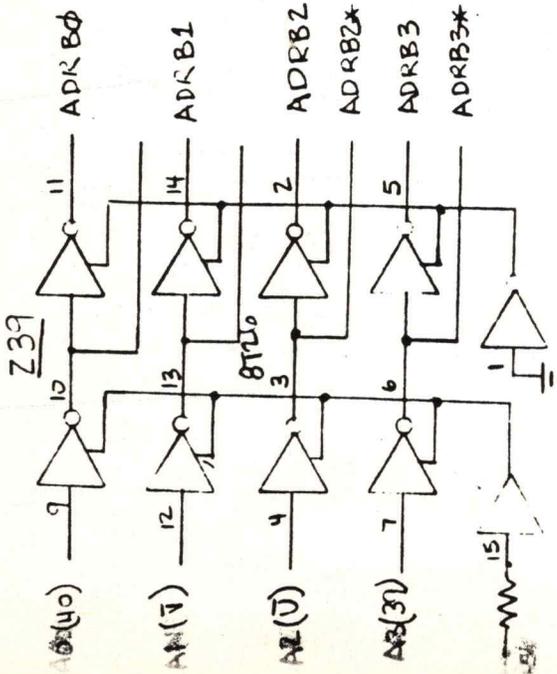
00047

*THE BUFFER IS LOCATIONS 1000-107F HEX

APPENDIX B

B-1 EXORDisk SCHEMATICS

The EXORDisk schematics are incorporated on the following pages.



ADDRESS BUFFERS

SCALE: NONE

DATE: 7-11-75

APPROVED BY:

DRAWN BY JDC

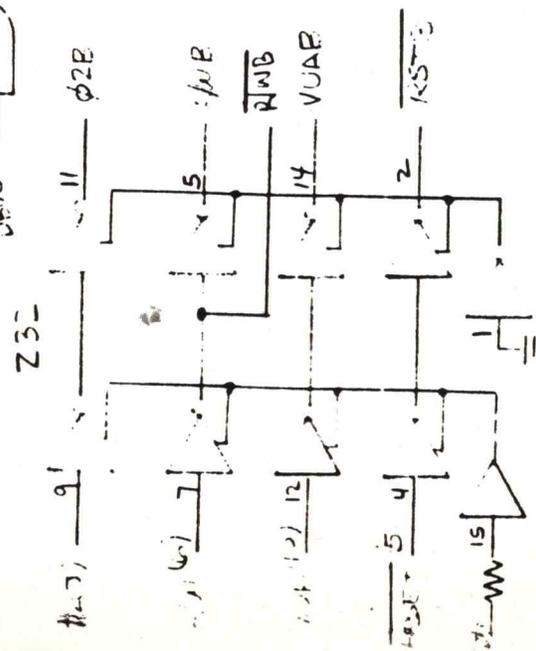
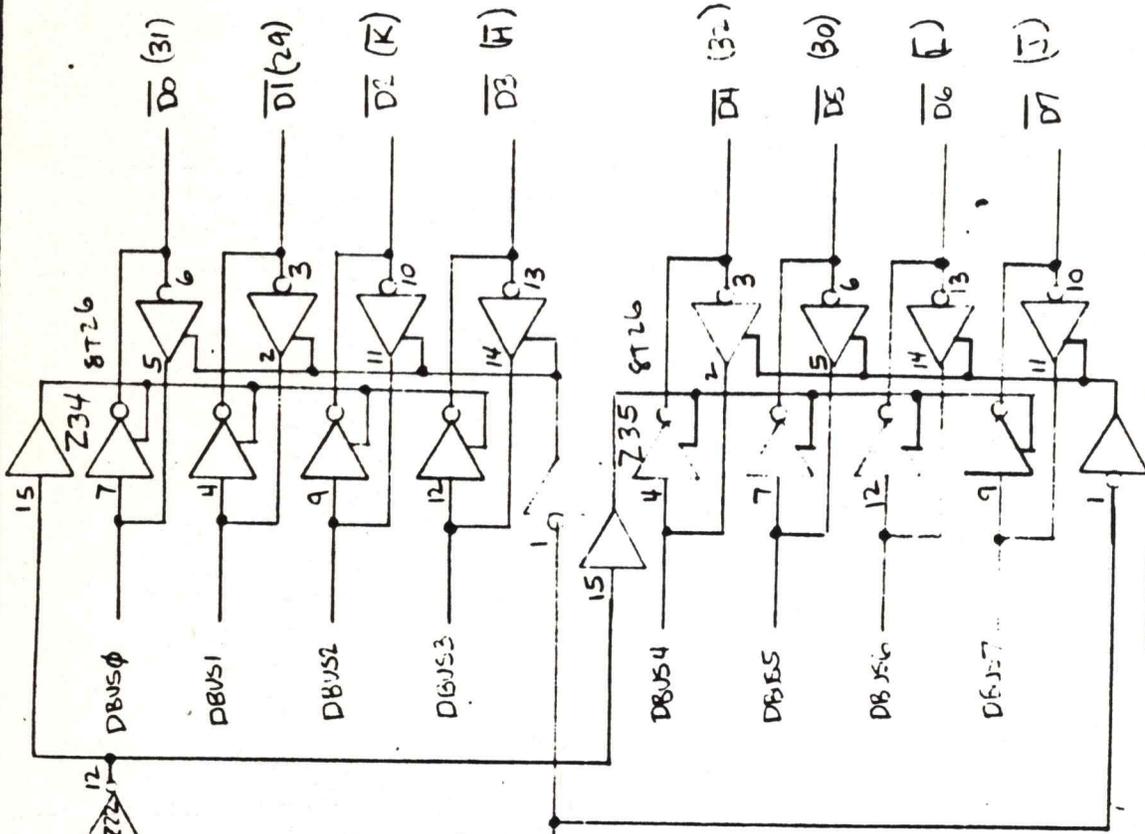
REVISED

-68 INTERFACE PCB

1 OF 7

DRAWING NUMBER

2000031-200



DATA BUS BUFFER

SCALE: NONE

APPROVED BY:

DRAWN BY JDC

DATE: 7-11-75

REVISED

-63 INTERFACE PCB

2067

DRAWING NUMBER

31-200

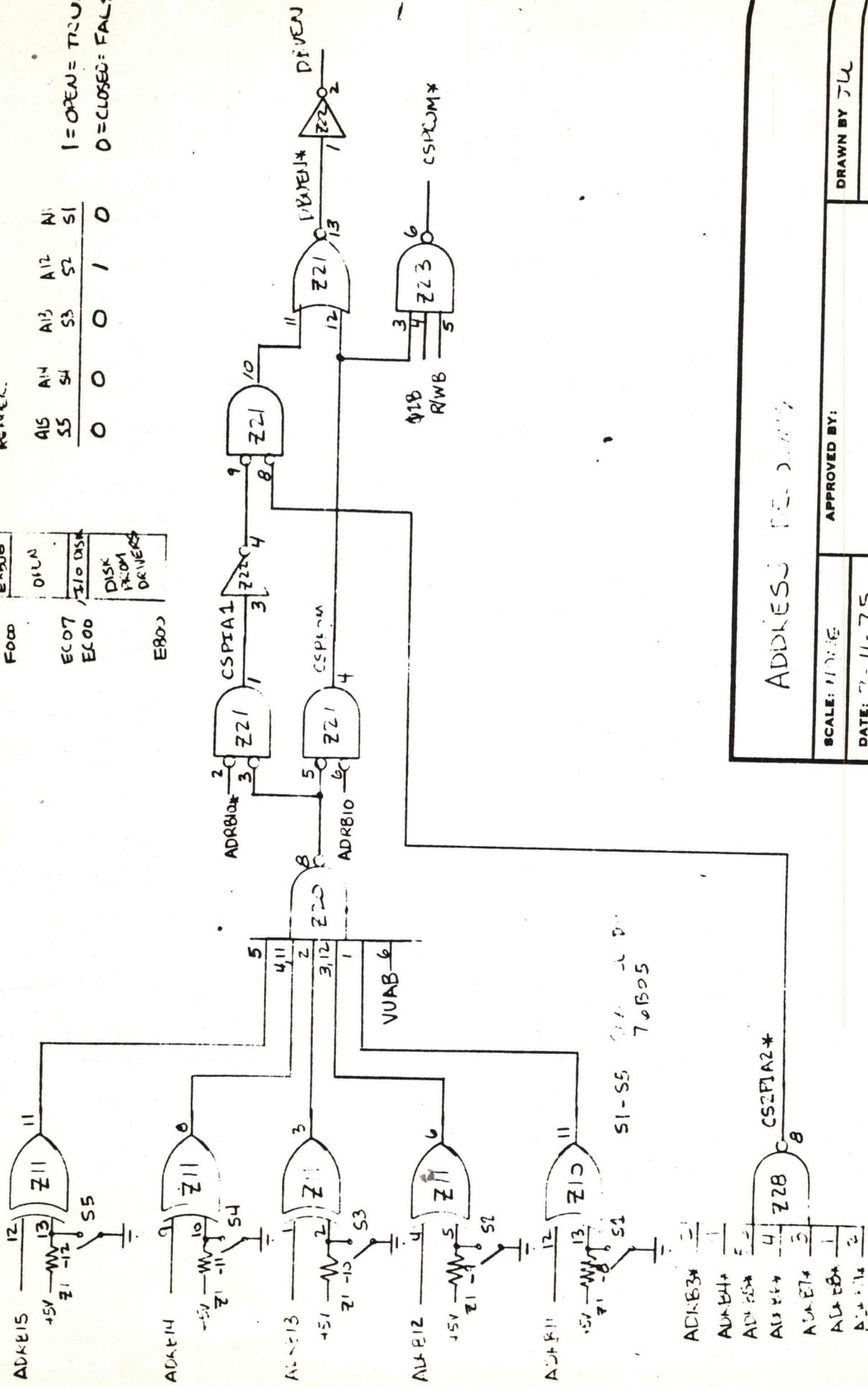
FDSO5 REQUIRES

| | |
|-------|-------------------|
| EX9J6 | DISK |
| EC07 | I/O DISK |
| EC00 | DISK FROM DRIVERS |
| EB00 | DRIVERS |

1KPKOM AT E800-EBFC
 DISK I/O AT EC00-EC07
 RESEC.

| | | | | |
|-----|-----|-----|-----|-----|
| A15 | A14 | A13 | A12 | A11 |
| S5 | S4 | S3 | S2 | S1 |
| 0 | 0 | 0 | 1 | 0 |

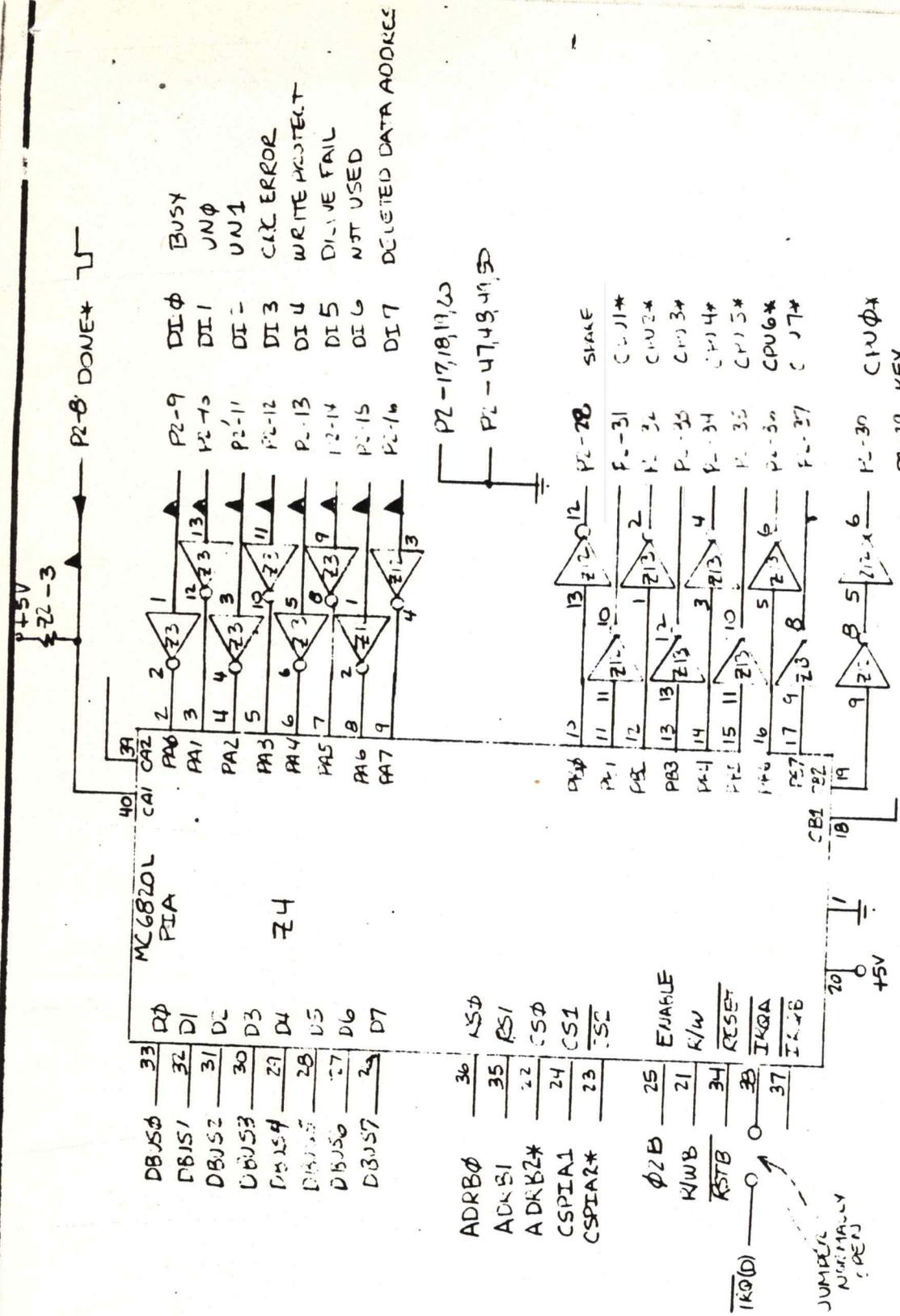
1 = OPEN = TRUE
 0 = CLOSE = FALSE



ADDRESS DECODING

| | |
|---------------|--------------|
| SCALE: NONE | APPROVED BY: |
| DATE: 7-11-75 | REVISION: |

.68 INTERFACE PCB

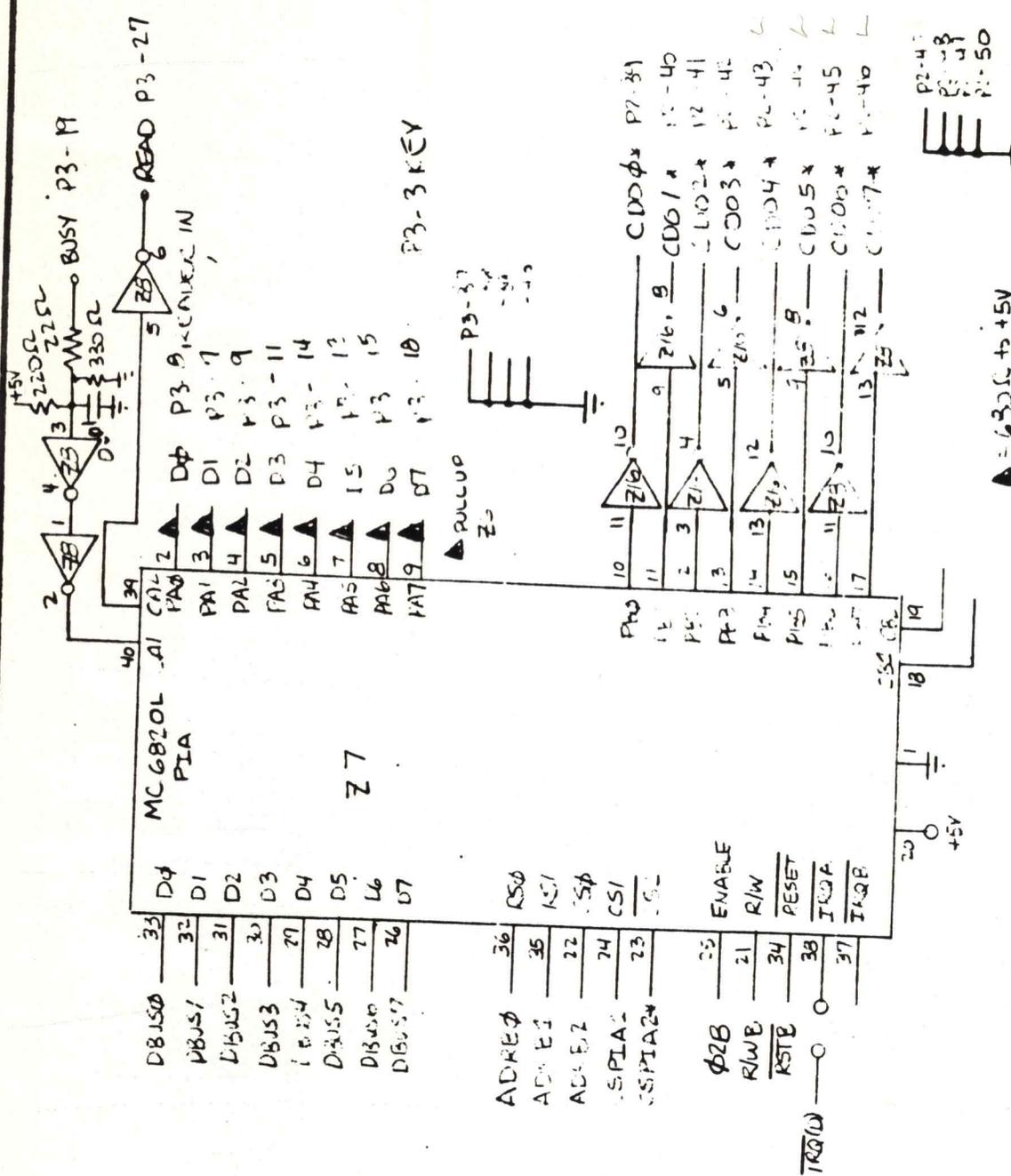


PIA COMMAND OUT-DATA/STATUS IN

| | | |
|-------------------|--------------|--------------|
| SCALE: NONE | APPROVED BY: | DRAWN BY JDC |
| DATE: 7-11-75 | | REVISED |
| -68 INTERFACE PCB | | |
| DRAWING NUMBER | | |

| | | | | | |
|-----|---|---|---|---|---|
| CLA | 1 | X | 1 | 0 | 1 |
| CPE | 1 | 0 | 1 | X | X |

VARIED FOR ...
= 1 FOR ...



PIA DATA OUT - READ - IN

SCALE: N/A

DATE: 7-11-75

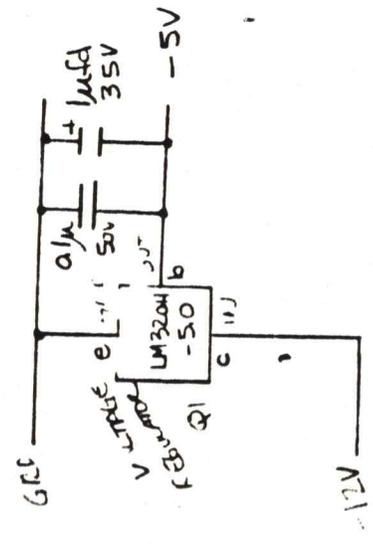
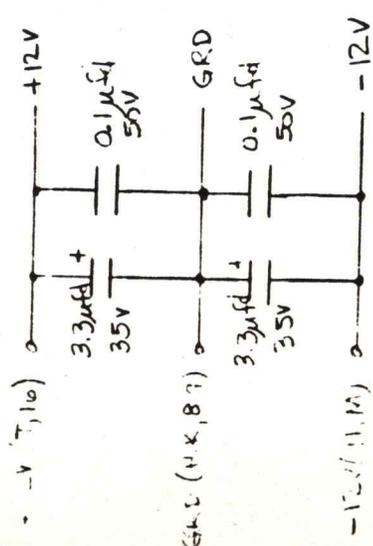
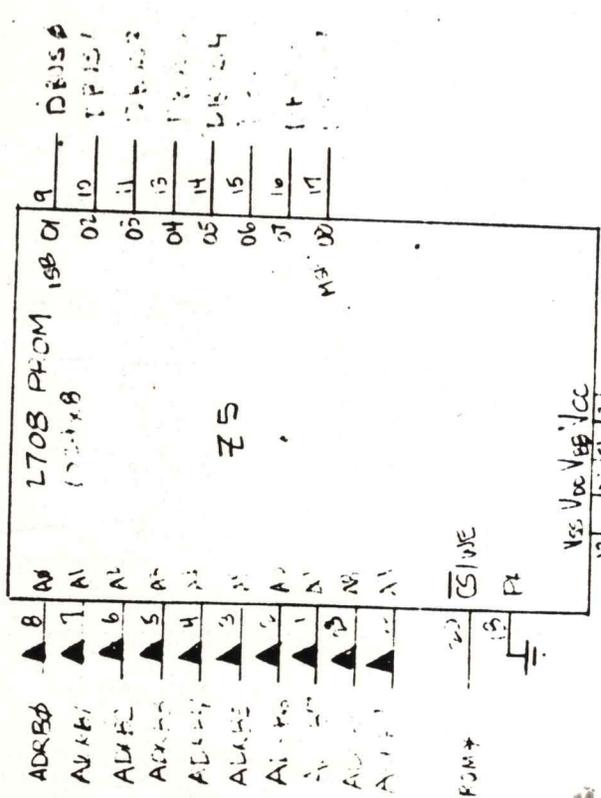
APPROVED BY:

DRAWN BY J.L.

REVISED

-68 INTERFACE PCB

DRAWING NUMBER



PROM BOARD

| | | |
|-------------------|--------------|--------------|
| SCALE: NONE | APPROVED BY: | DRAWN BY JJC |
| DATE: 7-11-75 | REVISED | |
| -68 INTERFACE PCB | | |
| JJC | | |
| DRAWING NUMBER | | 3000 1-10 |

