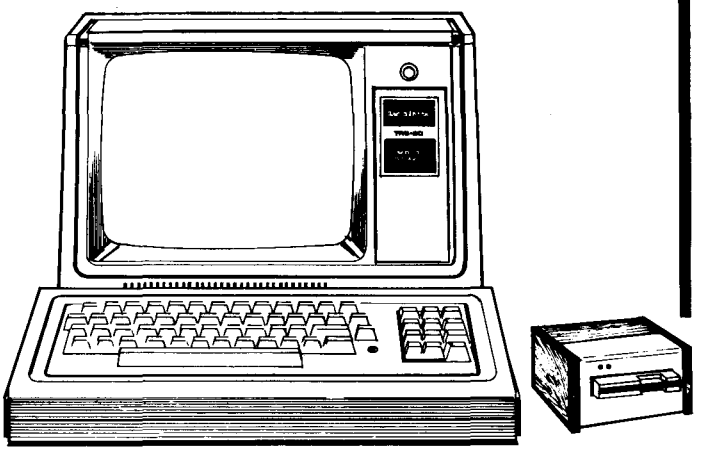


ESF for the TRS-80
users manual

exatron

USERS MANUAL

EXATRON STRINGY FLOPPY

FOR THE TRS-80

REVISED MAY 1980

exatron

excellence in electronics

Exatron Corporation, 181 Commercial St., Sunnyvale, CA 94086

TRS-80 is a trademark of Tandy Corporation
Stringy Floppy is a trademark of Exatron Corporation

TABLE OF CONTENTS

TABLE OF CONTENTS (Continued)

SECTION I

INTRODUCTION

1.1 GENERAL

Congratulations on being the new owner of an Exatron Stringy Floppy for your TRS-80! You will be pleased and perhaps quite surprised at what it will do for you. We will ensure that it serves you well.

This is your Users Manual. In it you will find what you need to know to install the Stringy Floppy (ESF for short), to check it out for proper operation, to use it to its full capability, to take care of it, to resolve any problem that might arise, and generally to answer all the questions we thought you might ask.

NOTICE — STUDY SECTION II BEFORE INSTALLING YOUR ESF

Use this manual to learn all about your Stringy Floppy. Be sure you understand what it will do for you, and be sure it does it well. The 30-day unconditional moneyback guarantee is for the purpose of making sure that if you are an ESF owner, you are a satisfied owner. If you are not, take advantage of the guarantee.

On the other hand, if you are satisfied, tell your fellow TRS-80 owners about it—or better yet, show them how it works. Take it to your TRS-80 Users Group meeting, and give a demonstration!

Your name will be kept in our computerized Owner File to keep you informed on product improvements, on new products, on ESF Owners Association activities, and to ensure good warranty service when needed.

We welcome your suggestions. If you have ideas on how this Users Manual can be more effective, or have questions without answers, let us know.

1.2 THE EXATRON STRINGY FLOPPY

The ESF is a mass storage subsystem for use with a microcomputer. It consists of a Drive Module, a flat cable for connection to your TRS-80, and a small sealed-unit power supply.

The power supply is already connected to the Drive Module, and is simply plugged into a wall receptacle or other 110 volt AC outlet. The flat cable, already hooked up, terminates in a 40-pin connector which mates with the card edge at the left rear of your TRS-80 chassis. The Drive Module contains the tape drive with slot for inserting wafers; two red light-emitting diodes (LEDs) to signal drive running and write operation; the drive electronics to process signals and to control the drive motor; the controller electronics to adapt signals to the specific microprocessor and bus; and a read-only memory (ROM) holding all the programs that operate your Stringy Floppy.

The medium on which programs and data are stored is a custom digital quality magnetic tape, wound as a continuous loop within a miniature cartridge called a wafer. Wafers are available with variable tape lengths, from 5 feet to 75 feet. With single-density packing, and normal drive speeds, a wafer will save or load 4K bytes of program on 5 feet of tape in 6 seconds. So a 20-foot wafer, for example, will hold 16K bytes and will cycle around the continuous loop of tape in 24 seconds.

The End-Of-Tape/Beginning-Of-Tape (EOT/BOT) splice reflects light, and is detected optically and electronically for proper loop-cycle control. Another optical sensor detects the presence of the write-enable sticker, a silvery reflector on the front of the wafer. If this silver circle is removed from the wafer, the sensor will disable the write circuits, and the wafer is write-protected.

The two most outstanding features of the ESF are its speed and its reliability. The baud rate of 7200 can be compared to the standard of 300 for a cassette recorder, or occasionally, 500 or 1200. As for reliability, you may never see some of the error messages in the system! Error detection is built-in and automatic, so you always know whether you have a correct save or load. Since the Stringy Floppy was designed from the ground up to digital standards, for use with industrial quality equipment, you are not hampered in any way by the

adaptation of audio equipment, audio materials, or audio standards for your microcomputer. You have no buttons, knobs, or switches to adjust when you save or load programs. The operations are all controlled by the software, and are highly reliable.

The programs in the firmware of your ESF ensure that it will work properly with either Level II or Level III BASIC in your TRS-80.

1.3 CLOCK SPEED-UP KITS

If your TRS-80 CPU clock circuit has been modified to run at a higher or lower frequency, the data density and storage capacity of the wafer will be affected. A double-speed modification will double the storage capacity of the wafers. Wafers can only be read back at the same clock speed used to save the program or data.

SECTION II

INSTALLATION AND CHECKOUT

2.1 INSTALLATION

Step 1 – Do your own receiving inspection by verifying receipt of all items on the packing list, and inspecting for physical damage. Report any irregularities immediately.

Step 2 – Make sure that your TRS-80 with supplementary equipment to date is working properly. It is impossible to evaluate the performance of a new subsystem when added to a main system working erratically. Turn on the TRS-80 WITHOUT the ESF and verify its operation. Then TURN IT OFF before connecting them up.

> NOTICE – THE CIRCUIT BOARDS USED BY RADIO SHACK ARE THICKER THAN NORMAL, AND THE CARD EDGES ARE NOT BEVELED. BEFORE YOU TRY TO CONNECT THE ESF TO THE TRS-80, READ THE INSTRUCTIONS JUST BELOW.

SPECIAL INSTRUCTIONS: It may be necessary to bevel slightly the top and bottom edges and the corners of the TRS-80 card edge, so as to be able to affix the ESF female connector. If this is done, be sure to: (1) use a fine file, such as a fingernail file; (2) draw the file only away from the card edge, not toward it or sideways, so as not to damage the metal fingers; and (3) don't permit any of the filed particles to get inside the computer box—use a vacuum cleaner.

It is recommended that you use an Exatron BUS EXTENDER on the card edge. This way you make the card edge connection only once, and you can make easy disconnects as often as needed between the BUS EXTENDER and the Stringy Floppy.

Step 3 – With the TRS-80 OFF, join the female 40-pin connector on the ESF flat cable to the card edge at the left rear of the TRS-80 chassis, or connect an Exatron BUS EXTENDER to the card edge and connect the ESF connector to the BUS EXTENDER. Observe the marking on the connector: THIS SIDE UP. If you use the Expansion Interface, you should use a BUS EXTENDER to connect the ESF close to the TRS-80 chassis, and not through the EI. We cannot guarantee that signals going through the EI are unaffected. Make all connections snug.

Step 4 – Make sure there is NO WAFER IN THE SLOT, then plug the sealed-unit power supply module into a convenient 110 volt AC outlet.

YOUR EXATRON STRINGY FLOPPY IS NOW PHYSICALLY INTEGRATED INTO YOUR COMPUTER.

2.2 SUBSYSTEM CHECKOUT

Step 5 – Now power up your computer. After answering "MEMORY SIZE" with the ENTER key, and getting back the TRS-80 sign-on message, type in the word "SYSTEM" (don't enter the quotes) and ENTER. Next type in "/12345" (no quotes), and ENTER. At this point you should see the title "EXATRON STRINGY FLOPPY" on the display. You should also see the Level II prompt sign ">", showing that program control has been returned to your Level II BASIC.

(Step X) – Now HALT and REVIEW. If you have followed all instructions to this point carefully, including Installation Steps 1 through 4, and you cannot successfully complete Step 5, go no further. Try again, starting with Step 1. If nothing changes, disconnect the ESF and call or write Exatron, DO NOT DISASSEMBLE ANYTHING. Your 30-day moneyback guarantee and your one-year full warranty are there to support the Exatron policy: "Every customer fully satisfied with our product."

Step 6 – Insert an ESF wafer in the slot at the front of the drive module, label side up, and seat firmly. Type in "@NEW" (without the quotes of course), and ENTER. You will see the righthand LED on the front of the drive module light up, showing that the motor is running, and shortly after you will see the lefthand LED light up, indicating that the tape is being written on. When the

tape completes one full cycle the write LED will go out, and the motor will keep running for another full cycle to verify the operation. Then the motor will stop, the verification message "DONE" will be displayed and control will return to Level II BASIC.

Step 7 – You have now completed the installation and checkout of your ESF. Before you get too enthusiastic however, read and understand thoroughly Sections III and V of this manual, to ensure lasting and reliable service.

NOTICE

Since there is no power switch built into the ESF, power is always on when the transformer is plugged into an active power socket. It would be a good idea to connect both the TRS-80 and the ESF to a wall socket or power strip controlled by a master power switch. In this case it would be desirable to leave the TRS-80 power switch on and turn power on and off to the TRS-80/ESF system with the master power switch. The ESF will be quite warm when power has been turned on for more than a few minutes.

SECTION III

ESF OPERATIONS WITH BASIC

3.1 GENERAL

The operation and control of your Exatron Stringy Floppy lies almost entirely in the software supplied in ROM or RAM, and the various functions are exercised simply and easily. One thing is important: you must read the instructions below, and follow them precisely. As with any other operation on your TRS-80, it is sometimes possible to make a misstep and lose program material and the time it takes to enter it.

3.2 SUBSYSTEM ENABLE

When you power up your TRS-80 with the ESF properly connected, you enter as usual Level II BASIC. In order to activate the ESF, you use the SYSTEM command to execute a program in the ESF ROM, which adds ESF central control and most functions and commands to your system, and also adds a KEYBOARD DEBOUNCE subroutine to your TRS-80. The program then returns control to Level II BASIC. From this point onward, you operate with the added capability as though it were designed into the original system.

3.2.1 Procedure

a. Make sure your ESF is properly connected, and then power up your TRS-80. After "MEMORY SIZE?", reserve memory if needed, and touch the ENTER key. BASIC will sign on and give a READY prompt.

b. Type the word "SYSTEM", and ENTER.

c. Type "/12345", and ENTER. You now see the ESF sign-on message, confirming subsystem activation. At the same time, control has been returned to Level II BASIC. At this point you have the @NEW, @SAVE, and @LOAD commands and functions available (see details further in this section). If you don't need data files now, skip subparagraph e. following.

d. In addition to the system entry command "/12345", you have several other entry address options, all of which initialize the ESF firmware. These may take you directly to the functions most often needed at the time of Subsystem Enable, thereby saving additional steps. Each can be entered immediately after typing the word "SYSTEM" and touching ENTER; each is explained in detail further on in this section. Here they are:

/12340 — load the next program file from Drive #0—same as @LOAD

/12341 — load File #1 from Drive #0—same as @LOAD1

/12342 — load File #2 from Drive #0—same as @LOAD2

/12343 — load File #3 from Drive #0—same as @LOAD3

/12344 — load File #4 from Drive #0—same as @LOAD4

/12345 — initialize ESF and return to BASIC (includes initializing KEYBOARD DEBOUNCE)

/12346 — initialize ESF *without* KEYBOARD DEBOUNCE and return to BASIC. This is useful if you already have a debounce routine, an upper-lower case routine, or an auto-repeat routine.

e. To add data file functions to your system, insert the Data I/O wafer into the Drive Module, type "@LOAD1", and ENTER. (The LOAD operation is described in detail below.) When you see the word "DONE", the tape has been successfully loaded. You now have the additional commands and functions needed to read and write data files.

> NOTE — THE DATA I/O PROGRAM IS INITIALLY LOADED AT 6C00 TO 6F91 HEX (27648 TO 28611 DECIMAL) BEFORE RELOCATING ITSELF TO TOP MEMORY. IF YOU HAVE A PROGRAM OR DATA IN THIS AREA, IT WILL BE WIPED OUT BY LOADING THE DATA I/O PROGRAM. IF YOUR STACK WAS IN THIS AREA, THE LOADING WILL CRASH. IF YOU HAVE RESERVED THE TOP OF MEMORY, THE DATA I/O PROGRAM WILL BE RELOCATED BELOW THE RESERVED MEMORY AREA. REFER TO APPENDIX C FOR A MEMORY MAP.

3.2.2 Argument And Punctuation

Following are the conventions, labels, and symbols used in the commands and procedures for the Stringy Floppy. Follow them exactly. If you find that a command doesn't operate, or operates improperly, it may be that you have departed from one of these standards.

n	File number, decimal, from 1 to 99, or any expression with this range of values
m	Number of 256-byte buffers to be used
d	Drive number — needed only in multi-drive systems; may be an expression or variable; default is always Drive 0
addr	Starting memory address, a decimal number
lnth	Length of file in bytes, a decimal number — this equals the end address minus the starting address plus one
aust	Autostart address, a decimal number
lexp	List of expressions: one or more expressions, separated by commas. (An expression is a combination of one or more constants, variables, and operations.)
lvar	List of variables: one or more variables, separated by commas. (A variable is a quantity that can take on any of a given set of values.)
[ ]	Indicates an optional argument in a command
ENTER	Touch the ENTER key — required after every command, but not shown

Following are the ESF functions and commands using BASIC: tape preparation, save, and load.

3.3 CERTIFY WAFER

a. Command — @[#d]NEW[n]

b. Function — With a new ESF tape wafer in the Drive Module, this operation causes a specific digital pattern to be written on the tape, and verifies the physical and electronic quality of the tape. Use this command with all new tapes before using them for program or data storage. Insert the wafer carefully in the slot, and type "@NEW". First you see the righthand LED light up, while the drive takes the tape back to the end-of-tape/beginning-of-tape (EOT/BOT) splice. The message "ERASING.." shows on the screen. Then you see the other LED light up, showing that the WRITE operation is under way. Finally the lefthand LED goes out while the righthand one stays on, for the VERIFY cycle, when the pattern on the tape is read back to certify the wafer. At the end you will see something like "ERASING..4202 BYTES.. DONE". If an error occurs, you will see the words "PARITY ERROR" or "CHECKSUM ERROR" instead of "DONE". If this happens, repeat the operation several times when working with a new wafer.

c. Alternate Function — When you want to save new programs or data on a wafer that has already been used, and you're finished with the previous material, use the @NEW command to clear up the tape before saving the new material. Use the command with a file number to wipe out that file and ALL FOLLOWING FILES on the wafer. The commands @NEW and @NEW1 both write over the entire wafer. The command @NEW4, for example, writes over File 4 and all following files, but not Files 1 through 3. Use the appropriate command whenever you want to delete former programs or data.

d. Wafer Storage Capacity — Use @NEWn when n is the next unused file number to measure the usable storage space remaining in the wafer.

3.4 SAVE PROGRAMS ON WAFER

3.4.1 SAVE BASIC Programs

a. Command — @[#d]SAVEn (file number 1 through 99)

b. Function — If you have a BASIC program in your system, the command "@SAVE1" will cause the program to be written on the tape, starting at the first position at the beginning of the tape (BOT). After the program is saved, the tape automatically cycles around the second time, and what's on tape is verified against the program still in memory. Successful completion is indicated by the word "DONE". This message, and the return to the Level II BASIC prompt, assure you that what you have on tape is exactly what you have in memory. Since this file is identified only by number, you must keep a note on the contents of the file for future access.

c. Examples — If you have already saved one or more files on this wafer, you use the command with the next unused numeral, for instance "@SAVE2". This causes the ESF to seek the end of the file with the next lower number, and save the new program following the old. This requires that you keep a note on how many files have already been saved on the wafer at hand, so you'll know the next number to use. Files must be saved in numerical sequence starting with File 1.

Another example. You have a 10-foot wafer (8K capacity) on which you've already saved two programs. There's a BASIC program in memory ready to save. If you key in "@SAVE3", the current program will be saved on the wafer following File 2. If you make a mistake and enter "@SAVE4", the ESF will continue to look for nonexistent File 3, and won't stop until you touch the BREAK key. If for some reason you enter "@SAVE2", the new material will be saved immediately following File 1, writing over and destroying the earlier File 2. If you use the command "@SAVE1", the new material will be saved starting at the beginning of tape, and will write over and destroy the original File 1. Also, if the new File 1 is longer than the old, the old File 2 will be affected so as to be no longer accessible. Even if the new File 1 is shorter than the old, the intervening useless data may impair your ability to access File 2 properly. If you want to start over with a new set of files, use the @NEW command, and start out with a clean tape.

REMINDER — Don't forget to note down somewhere — on an index sheet with your wafers, or on a 3x5 card near your computer — the contents of each file on a wafer, and how many files are already saved on each wafer to date.

3.4.2 SAVE Machine Language Programs

a. Command—@[#d]SAVEn,addr,lnth[,aust]

b. Function — This command will cause a machine language program with the indicated parameters to be written on tape. Use of an autostart is optional but recommended. Autostart can be disabled during LOAD when necessary. See 3.5.2 c. below. The description of the operation, the need for cataloging, and the proper choice of file numbers are all similar to those described above for saving BASIC programs. Note that this function is in effect a memory dump on tape. It doesn't have to be a machine language program: it can be any code that you want to restore unchanged to memory later.

c. Example — @#1SAVE2,17152,3800,18000 This command will save on Drive No. 1, with File Number 2, the machine language program residing in memory at 17152, which is 3800 bytes long, and which autostarts at address 18000.

3.5 LOAD A PROGRAM FROM WAFER INTO MEMORY

3.5.1 LOAD BASIC Programs

a. Command — @[#d]LOAD[n]

b. Function — With a wafer on which you've SAVEd one or more files, this command without file number will load the next file on tape from the current position. First the program will check whether there is enough memory available as determined by the "MEMORY SIZE?" chosen, without regard to variables and array already in memory. If there is enough room the program will be loaded, all variables and arrays will be cleared, and a new variable pointer established just above the top of the program. Then the Level II prompt is displayed. The LOAD operation includes a parity check for each byte, and a checksum for the entire file (two standard digital techniques for detecting errors). Successful completion, indicated by display of the word "DONE", assures you that what was loaded is exactly what was on the tape wafer. If you use this command with a file number, the ESF will load the program in the file selected, following the procedure described above.

3.5.2 LOAD Machine Language Programs

a. Commands — Same as for BASIC programs

b. Functions — Same as for BASIC programs

c. Autostart Override — To make a copy of a program which has an autostart, you need to disable the autostart function. To do this, load the program to be copied in the normal way (see 3.5.1 a and 3.5.2 a below) by typing "@LOAD[n]" and then ENTER; in addition, hold the shift key down while the program is being read, until it stops. You will see on the screen something like this:

```
@LOAD2
READING..DONE
BREAK
nnnn,mmmm,1111
FD ERROR
```

The numbers on the screen—nnnn,mmmm,1111—are in order the address, length, and autostart address. Now take the wafer out of the Drive Module, insert a fresh wafer, and type the SAVE command described in b. and c. above. The ESF will then copy your original program with autostart.

> NOTE — WE PUT A LOT OF EFFORT INTO MAKING IT EASY FOR OUR USERS TO MAKE BACKUP COPIES FOR THEIR OWN USE. WE HOPE YOU APPRECIATE THIS EFFORT AND WILL NOT USE THIS CAPABILITY TO BOOTLEG SOFTWARE.

3.5.3 Using @LOAD As A Program Statement

a. Statements — The commands @LOAD and @LOADn can also be used as statements within a BASIC program, thus serving to link two or more programs together. This technique is called chaining, or program overlay.

WARNING: Use no other BASIC statements or functions on the same line as the ESF program statement.

If you use @LOAD or @LOADn in a numbered statement within a BASIC program to call up a machine language program, created as described in Section 3.4.2 above, the operation will take place just as for a direct command. If an autostart was used, execution will start there right after the called-up program is loaded. If no autostart was used, the original BASIC program goes on as usual to the next numbered statement.

If you use either of these commands to call up a second BASIC program, the ESF firmware will check for enough memory below the variable pointer (not below the selected MEMORY SIZE). If there is enough room, the called-up program is loaded, ALL VARIABLES AND ARRAYS ARE PRESERVED, and the program is immediately executed.

This feature of saving the variables and arrays when using @LOAD[n] as a program statement gives you a powerful programming tool. If you are writing a long BASIC program, requiring more memory than you have available, you can divide the program in sections, with values of variables and arrays passed from one section to another. Or you can link related programs, so that again any common variables are passed from one program to another.

WARNING: When chaining programs as described above, you must either: (1) make the first program the longest, so that the variable pointer is established in memory high enough to leave room for later programs, OR (2) set the variable pointer yourself using a program statement, so that it will not be affected by any later program action.

b. Example — The following short BASIC program illustrates the chaining technique.

```
10 REM *** MENU TO SELECT GAMES ***
20 REM SAVE 3K FOR OVERLAY PROGRAM
30 POKE 16634, PEEK(16634)+12
40 REM ADDEND ABOVE IS 4* (NBR OF K)
50 REM SAVE 20 BYTES FOR STRING
60 CLEAR 20
70 PRINT "WHICH GAME DO YOU WANT?"
80 PRINT "1.NIM"
90 PRINT "2. STARTREK"
100 PRINT "3....."
.....
150 INPUT N
160 @LOAD N+1
170 REM THIS MENU IS IN FILE 1
180 REM NIM IS IN FILE 2, ETC.
```

3.6 WRITE DATA FILES

NOTE: These functions are available only after loading the Data I/O program from wafer.

a. Commands — (1) @[#d]OPEN[n]
(2) @[#d]PRINT lexp
(3) @[#d]CLOSE
(4) @CLEAR[m]

b. Functions — In order to WRITE a data file, you must first OPEN the file. This operation sets up the appropriate file identifiers before the PRINT operation starts. The OPEN command refers to the current drive: if there is only one drive, no designation is necessary. If you have more than one drive, and want to WRITE on a drive other than the current one, you must use the drive number option. Remember also that there can be only one file OPEN at a time on any one drive. After OPENing the file, use the PRINT command to write the file on tape. Again use the drive number only when you have more than one and want to change away from the current drive. The list consists of one or more expressions, whether numeric, algebraic, or string. Don't forget that the expressions must be separated by commas. After the data file is written on tape, using the PRINT command, the file must be CLOSEd. The use of a drive number is the same as before.

An additional command, useful for multi-drive systems, is "@CLEAR[m]". This command aborts all files not closed, clears I/O buffers in memory, and makes Drive 0 the current drive. The form with the optional argument, "@CLEARm", also clears all variables and strings, and reserves the specified number of buffers for tape I/O. The number of buffers, m, dictates the maximum number of files that you can open at the same time.

c. Examples (not consecutive or related)

@OPEN2 — Open File 2 on the current drive

@#1OPEN3 — Set current drive to #1 and open File 3

@PRINT 3+4,A(10)+3.1,A#,B#—PRINT (write) on the current drive the values of the expressions listed

@#1PRINT lexp — Set current drive to #1 and PRINT (write) the values of the expressions listed

@CLOSE — Close the file on the current drive

@#1CLOSE — Close the file on Drive #1

WARNING — IF YOU TRY TO WRITE MORE DATA ON TAPE THAN THERE IS ROOM FOR, THERE'S A CHANCE THAT YOU'LL COAST PAST THE END-OF-TAPE MARKER. THIS WILL DESTROY THE HEADER FOR FILE 1, WHICH IN TURN CAUSES ALL DATA ON THE WAFER TO BE LOST. BE SURE THAT THE WAFER YOU USE IS LONG ENOUGH TO HOLD THE AMOUNT OF DATA YOU WANT TO SAVE.

NOTE 1. The print operation must always be preceded by OPENing a file and followed by a CLOSE. Between OPEN and CLOSE, don't remove the wafer from the drive, and don't use it to SAVE or LOAD programs.

NOTE 2. @OPEN is used before either @PRINT or @INPUT (see below). Once the operation called up by either of these second commands has started, you can't change the operation back to the other type; you must terminate with @CLOSE and re-initiate the sequence.

NOTE 3. When you have an application program using data files, it's a good idea to make File 1 on that wafer the Data I/O Program. Then SAVE the application program as File 2. To copy the Data I/O Program, first clear memory in your TRS-80, and then follow the procedure in 3.5.2 c. above.

NOTE 4. In planning for the tape length necessary to hold your data files, allow for the fact that data is written in 256-byte records, with a 256-byte gaps between records. So only 50% of the tape length is actually used. If you do go past the end of tape, be sure to use @NEW to re-certify the wafer.

NOTE 5. Use @CLEAR at the beginning of a program using data files to ensure that all files that were opened by previous programs are forgotten.

3.7 READ DATA FILES

NOTE: These functions are available only after loading the Data I/O program from wafer.

a. Commands — (1) @[#d]OPENn
(2) @[#d]INPUT lvar
(3) @[#d]CLOSE
(4) @CLEAR[m]

b. Functions — In order to read a data file from tape into memory, you must first OPEN the file. The optional drive number is needed only when you have more than one drive, AND you want to change the current drive. Use the OPEN command, and then the INPUT command above. The variables must be separated by commas, and their types must match the types on the tape. After the values of the variables are assigned from the data on the tape, you must CLOSE the file. The @CLEAR commands function exactly as described in Section 3.6.

c. Examples

@OPEN3 — Open File 3 on the current drive

@#1INPUT C,A1,P2,D$,E$ — Set current drive to #1, and assign the values to the variables

@CLOSE — Close the file on the current drive

3.8 ERROR MESSAGES

3.8.1 Errors During @NEW, @SAVE, or @LOAD

As stated above, whenever you use the @NEW, @SAVE, or @LOAD commands, you are assured that the operation was completed without error when you see the message "DONE". If for any reason an error occurs, a message is displayed identifying the occurrence and the type. Here is a list of error messages.

a. Using @LOAD

PARITY ERROR — one or more bits did not load correctly; repeat @LOAD

CHECKSUM ERROR — two or more bits did not load correctly; repeat @LOAD

b. Using @SAVE

WRITE-PROTECTED — the reflecting sticker has been removed, so as to prevent the SAVE operation; you must replace the sticker

TAPE TOO SHORT — you ran out; save this on another wafer

NOTE 1. After any of the messages above is displayed, the program returns to BASIC, and displays either FD ERROR in Level II or BAD FILE DATA in Level III.

NOTE 2. One other error type without displayed message — if you try to load a file which does not exist on the wafer, your ESF will keep looking for it. Similarly if you try to save a program with a file number too high (the next lower number has not yet been used). If the drive keeps running without finding anything, touch the BREAK key; that will stop the drive motor.

3.8.2 Error During @[#d] OPEN[n]

Error Code	Error Message	Caused By
2	SN ERROR	Data I/O program not loaded, missing argument, etc.
5	FC ERROR	Drive d not there, file already OPENed on that drive, file nbr not 1-99, too many files
22	FD ERROR	Tape error, or BREAK key

3.8.3 Errors During @[#d] PRINT lexp

Error Code	Error Message	Caused By
2	SN ERROR	Data I/O program not loaded, syntax error in expressions, etc.
5	FC ERROR	Drive d not there, file not open, file was used for @INPUT
22	FD ERROR	ESF wafer WRITE-PROTECTED, tape too short, BREAK during tape operation

ALL OTHERS: SAME AS IN APPENDIX B, LEVEL II REFERENCE MANUAL.

3.8.4 Errors During @[#d] INPUT lvar

Error Code	Error Message	Caused By
2	SN ERROR	Data I/O program not loaded, syntax error in variable list, etc.
4	OD ERROR	End of file reached
5	FC ERROR	Drive d not there, file not open, file was used for @PRINT
22	FD ERROR	ESF tape error, or BREAK while reading

ALL OTHERS: SAME AS IN APPENDIX B, LEVEL II REFERENCE MANUAL.

3.8.5 Errors During @[#d] CLOSE

Error Code	Error Message	Caused By
2	SN ERROR	Data I/O program not loaded, etc.
5	FC ERROR	Drive d not there
22	FD ERROR	ESF wafer WRITE-PROTECTED, tape too short, BREAK during operation to CLOSE @PRINT file

3.8.6 Errors During @CLEAR[m]

Error Code	Error Message	Caused By
2	SN ERROR	Data I/O program not loaded, etc.
7	OM ERROR	Not enough memory for m buffers
22	FD ERROR	ESF wafer WRITE-PROTECTED, tape too short, BREAK during operation to CLOSE @PRINT file

SECTION IV

ASSEMBLY LANGUAGE OPERATIONS

4.1 LOAD AN ASSEMBLY LANGUAGE PROGRAM

It is possible that some Stringy Floppy owners without assembly language programming skills may have occasion to LOAD ESF wafers with assembly language programs or machine code written by others. You can load them by using exactly the same commands and procedure as for loading BASIC programs. Review Section 3.5.1 for details.

> NOTICE — THE REST OF SECTION IV IS FOR SKILLED ASSEMBLY LANGUAGE PROGRAMMERS ONLY. IF YOU USE BASIC ONLY, AND DO NOT WRITE YOUR OWN MACHINE CODE OR ASSEMBLY LANGUAGE PROGRAMS, OMIT THE REST OF THIS SECTION; YOU ARE NOT EXPECTED TO UNDERSTAND IT, AND YOU DON'T NEED IT NOW.

4.2 ASSEMBLY LANGUAGE SUBROUTINES

There are a number of distinct functions that make up the ESF firmware, and many of these are subroutines that can be used in other contexts. Each requires parameters in specified CPU registers before being called, and each ends with coded reports of successful completion or error. These subroutines are extremely useful once you have worked with assembly language, and can write the programs which invoke them.

NOTE 1. These are subroutines which must be CALLed at the proper address; they end with the RETURN opcode.

NOTE 2. Whenever these subroutines are used, a file must be OPENed and later CLOSEd, as for data files.

4.3 ERROR CODES

Each of the subroutines described below reports completion without error by returning with the Zero Flag SET (true, or binary 1). If an error has occurred, the Zero Flag will be RESET (false, binary 0, or NZ). Further, the type of error is identified by a status word in the accumulator, or A-Register. Each bit of the byte returned in A reports uniquely an error by the presence of a binary 1. Errors reported by each bit, starting with the least significant, are as follows:

D0 — Wafer is WRITE-PROTECTED (Readout is 00000001B or 01H)
D1 — BREAK key was hit by operator (00000010B or 02H, etc. below)
D2 — Not enough tape to SAVE program or to WRITE data identified
D3 — PARITY ERROR detected
D4 — CHECKSUM ERROR detected
D5 — Not enough memory to LOAD program or to READ data identified
D6 — File does not VERIFY
D7 — End-of-file mark on a data tape detected (80H)

If an error is reported, you can use a short routine to look at the A-Register and report which bit is SET.

4.4 FIND BEGINNING OF TAPE

a. Function — WIND tape to EOT/BOT marker and stop
b. Address — 3000H (hexadecimal)
c. Parameters — None required
d. Error reported — BREAK key used

This function positions the tape in the wafer for the start of appropriate operations. Use the @NEW command in BASIC to certify a new wafer or clean up superseded programs or data.

4.5 READ A DATA RECORD INTO MEMORY

a. Function — READ the next data record from tape into memory
b. Address — 3003H
c. Entry Parameters — HL register must point to the load address: the hex memory location where you want to load the material. BC register must

contain the hex number of bytes to be read into memory.

d. Return Parameters — BC register will hold the actual number of bytes read. The zero flag set shows no error.

e. Errors Reported — BREAK key used, parity error, checksum error, not enough memory, or end-of-file mark detected.

f. Note: If the record loaded is shorter than indicated by the entry in BC, BC will report the actual length of the record. If the entry was correct, BC will not change.

4.6 WRITE A DATA RECORD ON A WAFER

a. Function — Write a data record onto tape

b. Address — 3006H

c. Entry Parameters — HL must point to the memory location of the first byte to be written. BC must contain the hex number of bytes to be saved on wafer.

d. Return Parameters — The zero flag set indicates save with no error.

e. Errors Reported—Write-protected, BREAK, and not enough tape.

4.7 WRITE END-OF-DATA-FILE MARK

a. Function — Write end-of-file mark and file number at end of data file

b. Address — 3021H

c. Entry Parameter — File number, 1 to 99 binary, in Register A.

d. Errors Reported—Write-protected, BREAK, and not enough tape.

4.8 WRITE ASSEMBLY LANGUAGE PROGRAM ON A WAFER

a. Function — Save assembly language program material

b. Address — 300CH

c. Entry Parameters — HL must point to the memory location of the program to be saved. BC must contain the length of the program in hex bytes. DE must contain the starting address of the program: where it must reside when run. Register A, the accumulator, must hold the file number of the program — binary 1 through 99.

d. Errors Reported—Write-protected, BREAK, not enough tape, and does not verify.

4.9 MOVE TAPE TO BEGINNING OF FILE N

a. Function — Position tape to beginning of File n.

b. Address — 300FH

c. Entry Parameter — File number, 1 to 99 binary, in Register A.

d. Errors Reported — Parity error, BREAK

4.10 SELECT DRIVE

a. Function — Select drive number in multi-drive system.

b. Address — 3012H

c. Entry Parameter — Drive number, 1 to 7 decimal, in Register A.

d. Return — Zero flag set, done; zero flag reset, drive not there.

4.11 RETURN TO BASIC

a. Function — Returns program to BASIC and prompt.

b. Address — 3015H

c. Use — All files not having a normal auto-start address should use this address for autostart.

4.12 VERIFY TAPE

a. Function — Compare a file on tape against memory. Same as the second cycle in the BASIC @SAVEn command.

b. Address — 3024H

c. Entry Parameter — File number n in H.

d. Return Parameters — Number of bytes in HL, error code in A.

4.13 WRITE RECORD AND END-OF-FILE MARK

a. Function — Combination of 4.6 and 4.7 above, i.e., write data buffer and EOF.

b. Address — 3027H

c. Entry Condition — If BC = 0 only EOF is written.

4.14 DISPLAY ERROR MESSAGE

a. Function — Decodes error message in A (if any) and displays it on screen

b. Address — 302AH

SECTION V

CARE AND MAINTENANCE

5.1 GENERAL

The design components, materials, and packaging of the Stringy Floppy result in a product that is rugged, reliable, and difficult to abuse. If you operate your ESF properly, and exercise the proper care and maintenance, the life of the drive unit will be more than 2000 hours, and the life of the tape wafers will be more than 2000 passes.

5.2 DRIVE MODULE

Probably the most important point is this: DO NOT DISASSEMBLE. Aside from curiosity, there is no reason for you to go into the drive module, unless you are a manufacturer with an OEM application.

Next most important is to adopt working rules that avoid the possibility of dust accumulation inside the drive slot. If where you work (or play!) with your computer is kept clean, you will have no problems. Otherwise an improvised dust cover might be advisable. A practice which will be satisfactory in most cases is to leave a wafer—say one you use most often—in the drive slot, not quite all the way in. This will keep dust off both the tape head and the exposed tape in the wafer.

Be sure however not to have a wafer all the way in the slot when you turn on either the TRS-80 or the ESF. Carelessness here could cause you to accidentally erase a tape you didn't want to lose.

If your ESF power unit is plugged into an extension or wall outlet with a switch, it will be turned off when your other equipment is turned off. If not, and you're going to leave your system for a while, unplug your power unit.

Make a practice of keeping your Stringy Floppy Drive Unit on the left side of your TRS-80 keyboard, well forward of the CRT monitor or the Expansion Interface. If it sits too close to the monitor it may be affected by electromagnetic fields, and report errors not otherwise present.

There is no incompatibility between simultaneous hookup of both your cassette recorder and your Stringy Floppy. Both may be connected at the same time, and you can load a program from cassette, for instance, and then save it on an ESF wafer, without changing any connections.

5.3 WAFERS

The ESF tape wafer is an almost completely enclosed cartridge, with a 2.5 mm slot for the drive capstan, and a 14 mm slot for the tape head. Be sure to handle your wafers so as never to touch the tape with your fingers, or with any projection that could misalign it or pull out a loop.

Don't store your tapes near an electromagnetic field; you may lose the data on them. Keep them away from motors, transformers, power supplies, or any other field sources. In particular, don't read them, write on them, or lay them down, near the TRS-80 CRT monitor, with its high voltages and transformers.

KEEP THE DUST AND DIRT OUT. There are several ways. For a while you may want to slip them back into the plastic envelope they came in. Or you may find—or make—a covered plastic, metal, or wooden box to keep them in. Or, your local stationery store has plastic pages for three-ring binders, each page with about 10 pockets for business cards. These pockets are just the right size for your ESF wafers.

DO NOT INSERT OR REMOVE WAFERS from the Drive Module while the motor is running. Either instance can cause the tape loop to pull free from the case: once that has occurred, you'll not be able to use that wafer again. Insert wafers carefully, making sure that they are properly seated, past the detent. Remove them carefully also, so as not to damage the short section of exposed tape.

Most of the tape in ESF wafers is black, and does not create deposits on the read and write heads in the Drive Module. A few wafers were manufactured

with a brown tape, similar in appearance to audio tape. This is also a high-quality digital tape, but it can under some conditions leave a deposit on the heads. You can clean the tape heads by using a cotton swab and alcohol, normally available for this purpose commercially. Use the fluid sparingly, so as not to flood or dampen anything other than the tape heads. At the same time wipe the drive capstan with the swab and alcohol. To rotate the capstan to wipe it all around, remove the swab and turn on the motor briefly. DO NOT RUN MOTOR WHILE USING THE SWAB.

You may find occasionally while you are certifying a new wafer using the @NEW command, that you get an unexpected "WRITE-PROTECTED" message. This can be caused by a slight misplacement of the silver write-protect reflector circle. If the optical sensor doesn't find a strong enough reflection of light it disables the write function. Check the reflector location against that on your other wafers—a small shift may fix it. Other remedies may include using a small piece of shiny aluminum wrap; or painting around the edge of the circle with stenographic white-out; or even pasting a piece of white paper over the right locations.

A possible alternative to removing the silver circle when you want to write-protect a wafer, is to cover the circle with a piece of black paper, using mending tape. It works, and can be undone by simple removal when the time comes.

Remember to mark your wafers in some way to avoid mixing them up. If the manufacturer's label on the wafer will not readily accept marking, or if any mark is likely to be permanent, you can stick on a small pressure-sensitive label, and keep track of the contents of the wafer with an index. It is frustrating to end up a session with your microcomputer and ESF, and have a batch of unlabeled wafers that need to be loaded and sorted out. Use labels and avoid this.

If you follow all the suggestions above, you'll find that your Stringy Floppy will serve you well for a long time.

SECTION VI

THEORY OF OPERATION

The Exatron Stringy Floppy uses a miniature digital tape drive and endless loop magnetic tape wafers. Saturated recording techniques are used, and data is recorded on tape using a self-clocking bi-phase method of data encoding. This is the same technique as that used by floppy disks. When new information is recorded on the tape, all traces of the old information are obliterated. It's not necessary to erase tapes before recording new data over old data. However it's a good idea to use the @NEW command to remove old data and programs from a tape. This is accomplished by writing dummy information over the full length of the tape.

Utility programs to save and load programs and data are in a 2048-byte read only memory (ROM), one of the integrated circuits in the ESF for the TRS-80. This ROM occupies memory space beginning at hexadecimal address 3000. The top 128 locations in the ROM, 3780 to 37FF hex, are not used because this memory space is reserved. (Refer to page D/1 of the Level II BASIC Reference Manual.)

Eight input/output ports are used by the Exatron Stringy Floppy for reading and writing. The utility ROM uses port number F0 hex for input and output. Up to seven ESF drive units can be added, using port numbers F1 through F7 hex. A decoder must be used in the added units. Software to support extra drives is included in read/write memory.

APPENDIX A

GUARANTEE

Purchasers of the Exatron Stringy Floppy may return their units in good condition within 30 days of receipt and get a full refund of the purchase price.

WARRANTY

Exatron, 181 Commercial Street, Sunnyvale, CA 94086, warrants the electrical and mechanical parts and workmanship of the Stringy Floppy to be free from defects for a period of one year from date of purchase. This warranty EXCLUDES the magnetic tape wafers. Should the Stringy Floppy prove defective, it will be repaired free of charge. Purchasers should return the Stringy Floppy directly to Exatron at the above address. Exatron will pay shipping costs and will make repairs as soon as possible but in any event within 30 days.

This warranty does not cover damage resulting from accidents or abuse of the Stringy Floppy. This warranty will be null and void if there are any modifications to the design of the Stringy Floppy. Exatron shall not be responsible for any incidental or consequential damages. Some states do not allow the exclusion or limitation of incidental or consequential damages, so this limitation or exclusion may not apply to purchaser.

This warranty limits any implied warranty to one year from date of purchase. Some states do not allow limitations on how long an implied warranty lasts, so this limitation may not apply to purchaser.

This warranty gives purchaser specific legal rights and purchaser may also have other rights which vary from state to state.

RETURNS

Save the shipping carton for your Stringy Floppy. If you need warranty repair within the one-year period, write a detailed statement of what's wrong with your unit and enclose it with your ESF in the carton. If you need repair work after the warranty period has expired, follow the same procedure. You will receive the same service, with a reasonable charge for parts and labor.

APPENDIX B

SAVING MACHINE LANGUAGE PROGRAMS

Section 3.4.2 of this manual has the information you need to know in order to SAVE a machine language program, for the cases when you know all the relevant parameters: the starting address, the length in bytes, and the autostart or entry address. This works well when you wrote the assembly language program or object code yourself. What do you do when you have commercial software you bought on the cassette, and you want to transfer it to ESF wafer? Or when you want to make a backup copy of a program you have on hand and use regularly? (Making backup copies is important. Protect yourself this way against the possibility of inadvertent error.)

There are several answers to these questions. Let's start by charting the parameters for a number of popular programs written for the TRS-80. The information in this table has been furnished by Stringy Floppy owners, TRS-80 version, who were good enough programmers to find out for themselves. Additions to this table for other programs of general interest are welcome.

REMEMBER THE SAVE COMMAND:

@[#d]SAVEn,addr,lnth[,aust] (nbrs in decimal)

Program	Starting Addr	Length	Autostart
ESF Data I/O Program	27648	914	28416
ESF-80 Monitor	17152	2173	17152
Level III BASIC	17152	5376	17152
TRS-80 Editor/Assmblr	17152	6721	18058
Electric Pencil	17232	4144	17232
TBUG	17280	2560	17312
TSHORT	17131	562	17621

To SAVE these programs on ESF wafer, load the cassette without executing the program, and then use the @SAVE command. Do not start execution of the program before making the wafer copy: some of them contain self-modifying code which changes the program once executed. Wherever a "/" is called for to start execution, do not RESET: type "/6681" to return to BASIC, ready to use the ESF @SAVE command.

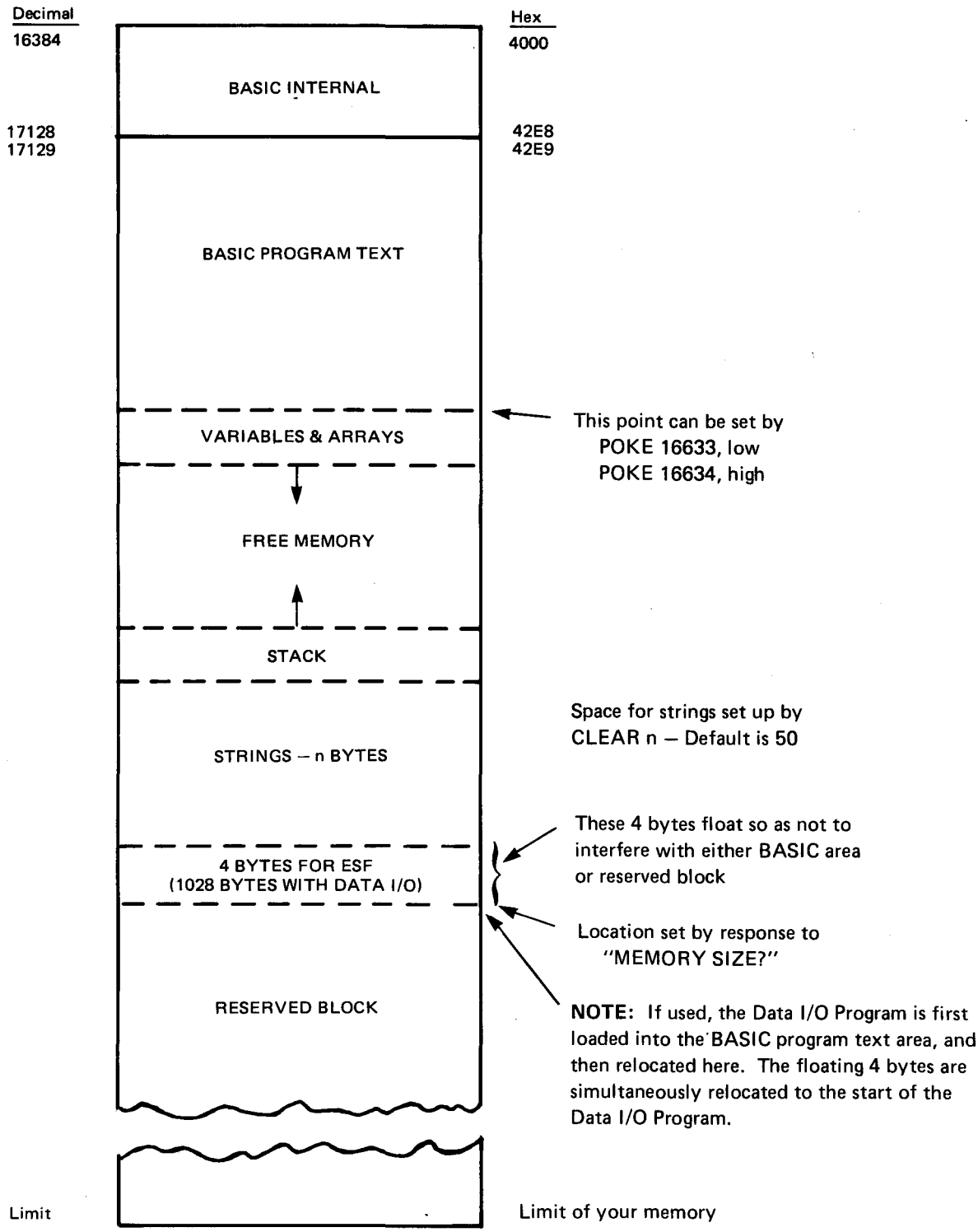
If you have programs not listed above which you'd like to transfer to ESF wafer, use the services of the ESF Owners Association to get information you don't have.

Another way to get the parameters you need on a program you bought on cassette is to use the ESF-80 Monitor. This monitor includes a command LOAD CASSETTE which includes identification and printout of the parameters you need. The monitor is available from Exatron Corporation.

APPENDIX C

MEMORY MAP

(Supplements pp. D/1, 2 in Level II BASIC Reference Manual.)



APPENDIX D

SUMMARY – COMMANDS & SYNTAX

DEFINITIONS

[ ] – Indicates optional parameters
d – Drive number (0-7)
n – File number (1-99)
m – # of I/O buffers (0-8)
ALL PARAMETERS ARE ENTERED IN DECIMAL

INITIALIZATION COMMANDS (After "SYSTEM")

Command	Syntax/Function	Example
/12340	Initialize firmware and load next file on wafer	[Same as @LOAD below]
/12341	Initialize firmware and load File 1	[Same as @LOAD1 below]
/12342–/12344	Initialize firmware and load Files 2, 3, or 4	[Same as @LOADn below]
/12345	Initialize firmware and return to BASIC	
/12346	Initialize firmware and return to BASIC without debounce subroutine—useful when your application program has debounce included	

GENERAL COMMANDS

@#d	@#d Select drive d and also change default drive to d. Default drive is 0 initially.	@#2
@NEW	@[#d] NEW[n] Certify wafer starting at file n to end of tape. Absence of n certifies whole tape.	@NEW2
@SAVE	@[#d] SAVEn Write BASIC program out to wafer at file n.	@#2SAVE1
@SAVE	@[#d] SAVE n,start addr,#bytes[,autostart addr] Write a machine language program to wafer at file n. If the optional autostart address if omitted, the subsequent @LOAD will return to BASIC after the file is loaded.	@SAVE1,17152,5380,17152
@LOAD	@[#d] LOAD[n] Load file n from ESF. Absence of n causes next file to be loaded. The same command is used for BASIC and machine language programs.	@LOAD2

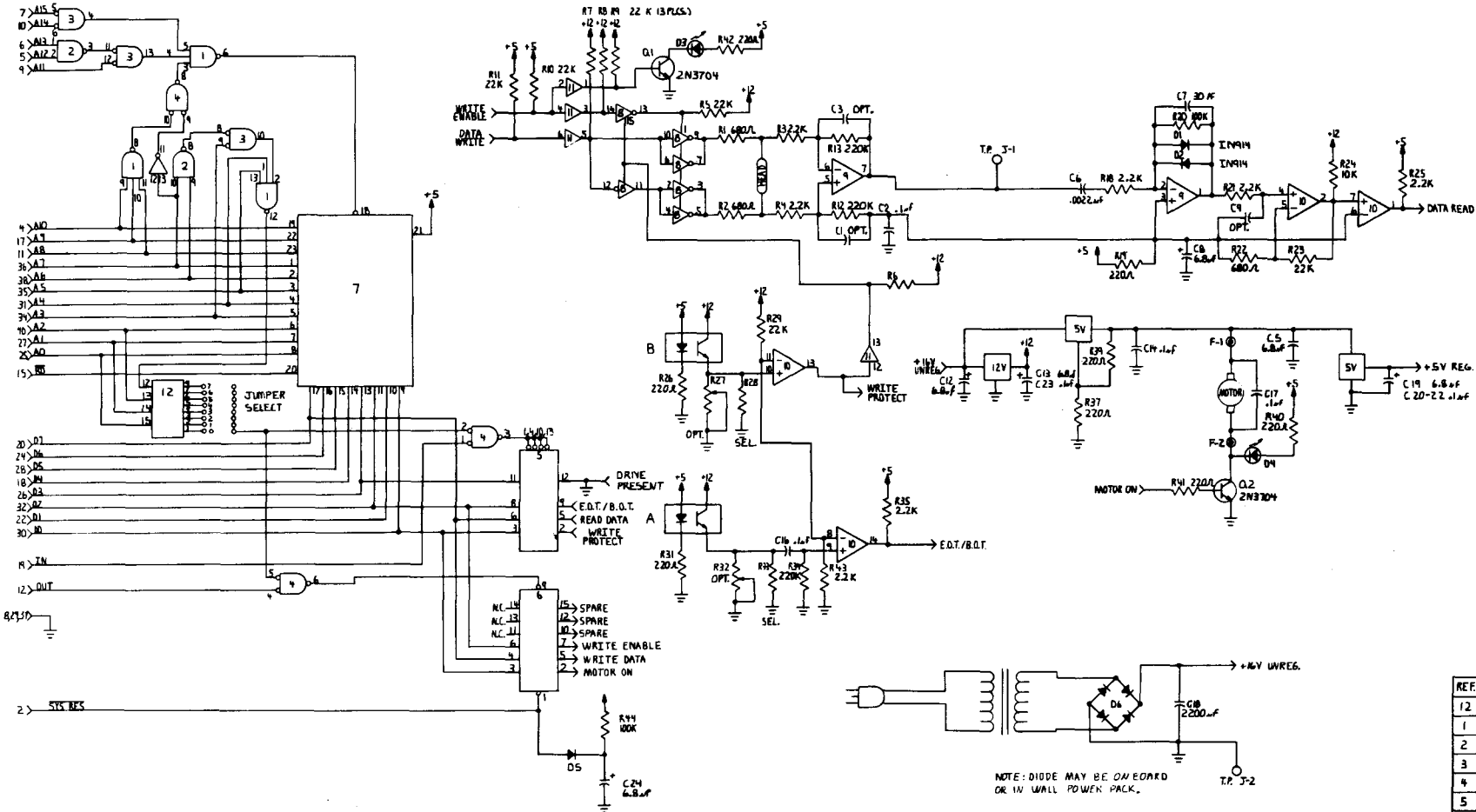
Command	Syntax/Function	Example
DATA FILE COMMANDS – Data I/O Program must be loaded		
@CLEAR	@CLEAR[m] Abort all files that are open. If m is specified, then m buffers are reserved for I/O and all variables are cleared. (Thus, this should be used at the beginning of the program only.) # of I/O buffers is set to 1 when the Data I/O program is loaded. @CLEAR will not change the # of buffers nor clear variables and can be used anywhere in the program to abort all opened files.	@CLEAR4
@OPEN	@[#d] OPENn Open data file n. The next @INPUT or @PRINT will dictate either the read or write mode. To change modes, @CLOSE the file and reopen first. THERE MAY BE ONLY *ONE* OPEN FILE PER ESF DRIVE – FOR READ OR WRITE AND NOT BOTH.	@OPEN1
@INPUT	@[#d] INPUT list-of-variables Read the list of variables from the opened file on the specified drive.	@#1INPUT A,B,C$
@PRINT	@[#d] PRINT list-of-expressions Write the values associated with the list of expressions out to the opened file on the specified drive.	@PRINT A,D$,A+B
@CLOSE	@[#d] CLOSE Close the file on the default drive, unless drive is specified.	@#2CLOSE

APPENDIX E

MULTIPLE DRIVE SYSTEMS

All original ESFs are shipped as a drive zero. If you order additional drives please specify which sequential drive number your new unit will be.

You may alter your new drive 1, for instance, back to a drive 0 by moving the jumper on IC12 down to zero and adding the ROM to the new drive. Then rejumper your old drive 0 to a drive 1. All additional drives will have a unique jumper and no ROM. Note: You must have one drive 0 on line at all times. You must not have two drives of the same number on line at the same time.



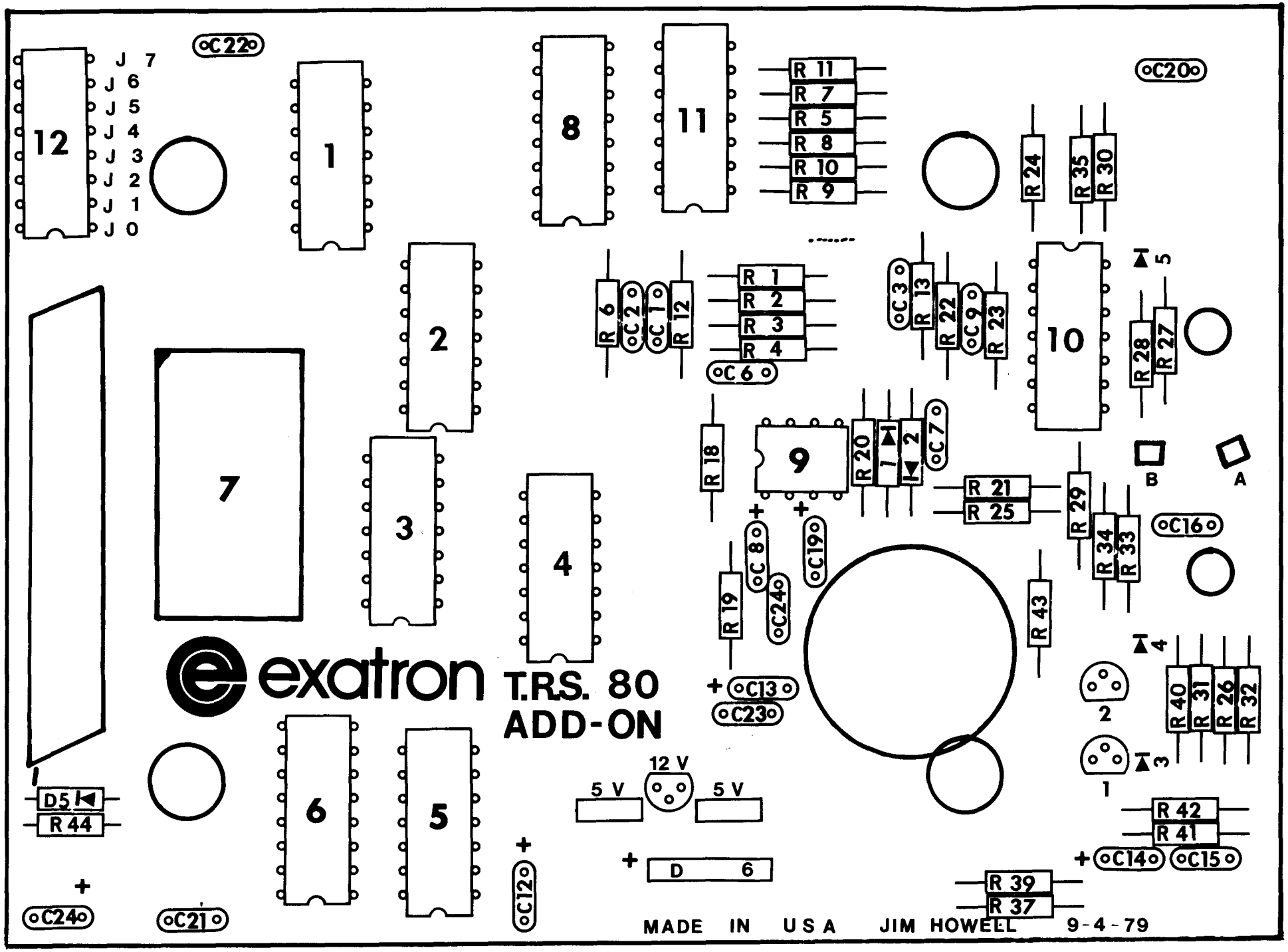
REF.	PART NO.	+5	+12	GND
12	74LS42	16		8
1	74LS10	14		7
2	74LS00	14		7
3	74LS02	14		7
4	74LS32	14		7
5	LM8093	14		7
6	74LS174	16		8
7	2716	24		12
8	80C98		16	8
9	LF353		8	4
10	LM339	2		12
11	74C906	14		7

exatron

NONE		JHH
4-9-79		

T.R.S. 80 ADD-ON

1000-031


C22
J 7
J 6
J 5
J 4
J 3
J 2
J 1
J 0
12
1
8
11
R 11
R 7
R 5
R 8
R 10
R 9
C20
R 24
R 35
R 30
2
R 6
C 2
C 1
R 12
R 1
R 2
R 3
R 4
C 6
C 3
R 13
R 22
C 9
R 23
10
5
R 28
R 27
7
3
4
9
R 18
R 20
C 7
B
A
R 21
R 25
R 29
C16
C 8
C24
C19
R 34
R 33
R 19
R 43
exatron
T.R.S. 80
ADD-ON
4
C13
C23
2
R 40
R 31
R 26
R 32
3
1
12 V
5 V
5 V
6
5
D5
R 44
R 42
R 41
C14
C15
C12
D 6
R 39
R 37
C24
C21
MADE IN USA JIM HOWELL 9-4-79

ESFOA

EXATRON STRINGY FLOPPY OWNERS ASSOCIATION

Secretary, Fred Waters

WHAT IS IT?

ESFOA is a voluntary association of all those who own the Exatron Stringy Floppy mass storage subsystem, and of those who are interested in the ESF to the point of taking part in ESFOA activities. There is no formal organization, no requirements, no dues, and the degree of participation is entirely up to the individual.

THE PURPOSE

The purpose of ESFOA is to facilitate the interchange of ideas, information, software, and hardware design, to encourage mutual assistance and cooperative projects among members with specific common interests, and to furnish the administrative backup to support these goals.

HOW DOES IT WORK?

1. To start off, there is a permanent secretary for the administrative backup.
2. For those within reach, there is a workshop every Saturday morning at the Exatron plant in Santa Clara from 9 to 11. These have been continuous since January 1978. There is always coffee on hand, a round table for discussion of whatever comes up, and friendly help from others members when needed.
3. We put out a monthly newsletter. Originally it was published locally and distributed with a mailing list. Now it is a page in Microcomputing, a leading national magazine. This is the medium for the broadest coverage to accomplish the goals of ESFOA.
4. Each interested member fills out a sheet, his "Activity/Interest Record", and we put them all together in a binder. You find that another member is about to write a program you just finished, or vice versa. Or you find that there are two other members who like you want to use their microcomputer and Stringy Floppy, for instance, to keep track of commodity futures. It's available to all members.
5. For those interested in writing programs for possible commercial sale and distribution, ESFOA musters the assistance, the motivation, and practical support in the way of field test and limited commercial distribution among members.

WHY AN OWNERS ASSOCIATION?

Looking back a short four years ago, personal computing was limited to a tight little circle of dedicated and highly skilled enthusiasts. Most of those involved were already professionals in the industry–electronic engineers, or computer programmers. In this short span there took place a most astounding growth: in concepts, in the manufacture of the systems and components for microcomputers, in systems software to make it all work, and in program languages and applications programming for us all to do interesting things with. So most of the people who fool around with microcomputers today are people who are learning, who are starting out with the more popular games and utility programs, and who have all sorts of opportunities ahead of them for wider and more intensive use of their systems.

During this time there has been one overriding limitation on progress–one factor which operates like a governor to slow down our ability to do all we want with our systems. That factor, that limitation, is the constant and pressing need for more and better software. Once we have begun to appreciate what we can do with a microcomputer, and once we've had a chance to satiate the initial high enthusiasm for games, we find that we still need more and better software-systems software and practical applications software. Well, to some degree we're all becoming programmers, even the beginners among us. And we certainly have a good sprinkling of fine programmers and professionals. So we can all benefit from each others' efforts to some degree.

Here are some of the types of material we can write, improve on, field-test, exchange, develop for commercial sale, or otherwise work with: expanded ways to use the ESF hardware, input/output interfaces, utility routines, techniques, methods and tricks, monitors, debugging ideas, adaptations for specific hardware configurations, new and useful home applications, high-utility small business applications, and even assemblers and tiny languages.

ESFOA operates on a policy of giving software credit where due, and will not sanction the copying and exchange of proprietary software. We encourage the writing of software among our members, and where the author wants to market his ideas, we afford the maximum protection. Other authors believe in donating smaller works to the public domain. In either case the author gets the credit and whatever else he's entitled to, and we follow the same policy on any software from any source.

So these are some of the reasons why we have an owners association, and some of the ways that all members can benefit if they choose.

HOW HAS IT WORKED SO FAR?

The answer to this question is what convinces us that an owners association can and will be successful. When the S-100 version of the Exatron Stringy Floppy was getting started, all it had in the way of software was a rudimentary utility routine, just enough to make the hardware operate. That was a long way from having a highly useful mass storage subsystem. One of the first members of ESFOA was willing to put his talents to work to develop a much more sophisticated and efficient utility routine, and then greatly extend the usefulness of the ESF by writing a monitor. That monitor incorporates the necessary operating routines to exercise the ESF fully, and with subsequent improvements and add-ons, it is still the basis for the great success of the S-100 ESF. Another member wrote a tape operating system for the Stringy Floppy, which gives it many of the operating features of a disk subsystem. Other projects undertaken by members are a relocator, a linking loader, a disk-compatible operating system, augmentations and expansions to existing assemblers and interpreters, a scheme for position-independent code, and a disassembler. On a more day-to-day level, the experienced programmers have always been extremely helpful to others, with answers to knotty questions, short pieces of code to solve a particular problem, or general guidance in programming. WE KNOW THIS IDEA WORKS!

So there you have it. The Exatron Stringy Floppy Owners Association will continue to encourage all kinds of cooperation among the members, and provide services that members find useful.